



PENGEMBANGAN MODEL KLASIFIKASI
MORPHOLOGICAL NEURAL NETWORK UNTUK
SISTEM PENGENALAN EKSPRESI WAJAH

KUALIFIKASI

Robert
99223138

PROGRAM DOKTOR TEKNOLOGI INFORMASI
UNIVERSITAS GUNADARMA
Juni 2024

DAFTAR ISI

Bab 1	3
1.1 Latar Belakang	3
1.2 Rumusan Masalah Penelitian	6
1.3 Batasan Masalah Penelitian.....	6
1.4 Tujuan Penelitian.....	6
1.5 Kontribusi dan Manfaat Penelitian.....	7
Bab 2	8
2.1 Pengolahan Citra	8
2.1.1 Definisi Citra.....	8
2.1.2 Citra Berwarna	9
2.1.3 Citra <i>Grayscale</i>	10
2.1.4 Citra Biner.....	12
2.1.5 Konvolusi.....	13
2.1.6 Morfologi Matematika	15
2.1.7 <i>Viola Jones Algorithm</i>	23
2.1.8 Perubahan ukuran citra.....	26
2.2 Ekstraksi Fitur	26
2.2.1 <i>Local Monotonic Pattern</i>	26
2.2.2 <i>Gabor Filter</i>	29
2.3 Artificial Intelligence	31
2.3.1 <i>Machine Learning</i>	33
2.3.2 Deep Learning.....	36
2.3.3 Metrik model klasifikasi	42

2.4	Peta Jalan Penelitian.....	45
Bab 3	47
3.1	Alur Penelitian.....	47
3.2	Pengumpulan Citra Ekspresi Wajah	48
3.3	Pembentukan <i>Dataset</i>	51
3.3.1	Deteksi wajah.....	53
3.3.2	Image Resize	58
3.3.3	Color conversion.....	61
3.3.4	Gabor Filter.....	62
3.3.5	Ekstraksi fitur LMP.....	64
3.4	Pembentukan Model.....	66
3.4.1	Pembentukan Model SVM.....	66
3.4.2	Pembentukan Model CNN.....	66
3.4.3	Pembentukan Model MNN	68
3.5	Time Table.....	85
Bibliografi	86

Bab 1

Pendahuluan

1.1 Latar Belakang

Penelitian tentang robot cerdas terus meningkat dari tahun ke tahun (Mukherjee, Gupta, Chang, & Najjaran, 2022). Robot cerdas merupakan robot yang dapat melakukan tugas dengan berinteraksi dengan lingkungan sekitarnya. Pada umumnya robot cerdas dikembangkan untuk bidang industri (Garcia, Jimenez, De Santos, & Armada, 2007). Salah satu contoh robot cerdas yang digunakan pada bidang industri adalah *welding robot*. *Welding robot* merupakan robot yang dapat melakukan proses pengelasan pada sebuah objek tanpa campur tangan manusia (Coman, Rontescu, Bogatu & Cicic, 2021). Namun, selain robot dalam bidang industri, penelitian tentang robot cerdas bidang sosial menjadi salah satu penelitian terbanyak dalam beberapa tahun terakhir (Johal, 2020). Robot sosial merupakan robot yang didisain untuk berinteraksi dengan manusia (Rawal & Stock-Homburg, 2022; Retto, 2017). Robot sosial tersebut didisain agar dapat berinteraksi layaknya seperti manusia, di mana robot harus memiliki kemampuan berkomunikasi dalam bahasa tingkat tinggi (Hegel, Muhl, Wrede, Fastabend & Sagerer, 2009). Salah satu contoh robot sosial adalah Sophia, Sophia merupakan robot yang manusia dan dapat melakukan percakapan dengan manusia dalam bahasa tingkat tinggi, mengenal wajah seseorang, dan mengeluarkan ekspresi, dan, namun tidak memiliki kemampuan untuk mengenal ekspresi wajah seseorang (Retto, 2017). Ekspresi wajah merupakan salah satu komponen penting agar robot dapat berkomunikasi sesuai dengan keadaan (Frith, 2009). Selain bidang robotika, ekspresi wajah juga berperan penting dalam bidang psikologi, di mana emosi seseorang dapat mempengaruhi pengambilan keputusan seseorang (George & Dane, 2016), berdasarkan penelitian (Rawal & Stock-Homburg, 2022), 55% informasi afektif didapatkan melalui ekspresi wajah.

Artificial Intelligence (AI) adalah bidang yang digunakan agar mesin dapat berpikir, dan mengambil keputusan layaknya seperti manusia (Khanzode, 2020). AI merupakan bidang yang sangat luas, didalam AI terdapat *Machine Learning* (ML), di mana ML memungkinkan untuk mesin untuk belajar dari data yang diberikan serta berpikir dan mengambil keputusan (Awad & Khanna, 2015; Khanzode, 2020). Walaupun demikian, performa dari ML sangat bergantung pada proses ekstrasi fitur (Alzubaidi, Zhang, Humaidi, Al-Dujaili, Duan & Al-Shamma, 2021; Shinde & Shah, 2018). Oleh karena itu terdapat pembelajaran yang lebih dalam dari ML yaitu *Deep Learning* (DL). DL memungkinkan mesin tidak hanya belajar, berpikir, dan mengambil keputusan, namun dapat memilih data mana yang penting dan yang tidak penting dalam mengambil keputusan, sehingga membuat mesin tidak terlalu terpengaruh oleh *noise* (data tidak penting) (Janiesch, Zschech & Heinrich, 2021).

Banyak penelitian yang memanfaatkan ML dan DL untuk mengenal ekspresi wajah. Secara garis besar penelitian-penelitian tersebut menggunakan model *Support Vector Machine* (SVM) dari model ML yang digabung dengan metode ekstrasi fitur atau menggunakan model *Convolutional Neural Network* (CNN) dari model DL. Pada penelitian (Shan, Gong & McOwan, 2009), digunakan *Local Binary Pattern* (LBP) untuk ekstrasi fitur dan SVM sebagai model untuk mengenal ekspresi wajah berdasarkan dari hasil ekstrasi fitur LBP. Pada penelitian (Mohammad & Ali, 2011), digunakan *Local Monotonic Pattern* (LMP) untuk ekstrasi fitur dan SVM. Pada penelitian (Deshmukh, Patwardhan & Mahajan, 2016), digunakan *Viola-Jones Algorithm* (VJA) untuk deteksi wajah, LBP, dan SVM untuk membuat sistem pengenalan ekspresi wajah secara real-time. Pada penelitian (Sawardekar & Naik, 2018), digunakan LBP, CNN, dan *K-Nearest Neighbor* (KNN) model ML untuk pengenalan ekspresi wajah. Pada penelitian (Turabzadeh, Meng, Swash, Pleva & Juhar, 2018), digunakan LBP dan KNN untuk membangun sistem pengenalan ekspresi wajah secara *real-time*. Penelitian (Eleyan, 2023), dilakukan analisis berbagai *histogram-based feature extraction algorithm* dan KNN. Selain dari model-model yang sudah disebutkan di atas, terdapat penelitian yang mengembangkan model DL, yang memanfaatkan morfologi matematika untuk melakukan ekstrasi fitur yaitu *Morphological Neural Network*

(MNN). Hasil penelitian (Shen, Shih, Zhong, & Chang, 2019) menunjukkan bahwa MNN memiliki performa yang setara dengan CNN namun dengan kompleksitas yang rendah.

Secara garis besar model CNN dan MNN memiliki arsitektur yang mirip, perbedaan dari kedua model tersebut adalah CNN menggunakan konvolusi untuk ekstraksi fitur, sedangkan MNN menggunakan morfologi untuk ekstraksi fitur, kedua model kemudian menghubungkan lapisan ekstraksi fitur ke Fully Connected *Layer* (FC-Layer) yang dapat dikatakan sebuah Artificial Neural Network (ANN) untuk melakukan klasifikasi objek.

Jenis morfologi yang digunakan untuk model MNN pada penelitian (Shen et al., 2019) adalah *opening* dengan *structure element* (SE) yang digunakan adalah *disk*. SE merupakan komponen penting dalam operasi morfologi. SE digunakan untuk penentuan pengambilan nilai dalam sebuah area pada citra. Hasil dari operasi morfologi *opening* tidak mengekstraksi fitur secara utuh. Berdasarkan uraian diatas, maka dalam penelitian ini diusulkan “*pengembangan model klasifikasi Morphological Neural Network untuk sistem pengenalan ekspresi wajah*”. Dalam Penelitian ini diusulkan metode ekstraksi fitur menggunakan operasi morfologi yang meminimalisir hilangnya fitur akibat operasi. Operasi yang diusulkan kemudian dikembangkan menjadi sebuah model MNN yang dapat mengekstraksi dan juga mengenali ekspresi wajah manusia. Selain itu, penelitian ini juga melakukan pengenalan ekspresi wajah menggunakan model SVM dan CNN. SVM terdapat metode tambahan yang digunakan untuk meng-ekstraksi fitur ekspresi wajah. Dalam penelitian ini model-model tersebut (MNN, SVM, dan CNN) akan dilatih menggunakan *dataset* gabungan baik *dataset* primer maupun sekunder. Pengenalan ekspresi wajah menggunakan model SVM dan CNN juga merupakan lanjutan dari penelitian (Robert, 2023), di mana pada penelitian tersebut terdapat kekurangan yang dapat dilakukan dalam penelitian ini seperti keterbatasan *dataset* dan kurangnya variasi etnis.

1.2 Rumusan Masalah Penelitian

Berdasarkan dari uraian latar belakang masalah dan usulan topik penelitian diatas, terdapat beberapa masalah penelitian yang perlu dicari solusinya adalah:

1. Metode dan algoritma morfologi matematika mana yang paling tepat digunakan untuk mengekstrasi fitur ekspresi wajah.
2. Bagaimana membangun model pengenalan ekspresi wajah berdasarkan gabungan usulan morfologi matematika dan jaringan saraf tiruan (usulan model MNN).
3. Bagaimana membangun *prototype* sistem pengenalan ekspresi wajah menggunakan usulan model MNN.

1.3 Batasan Masalah Penelitian

Adapun batasan masalah dari penelitian ini adalah:

1. Fitur yang digunakan dalam penelitian ini adalah bentuk komponen wajah yang menggambarkan ekspresi.
2. *Dataset* yang digunakan dalam penelitian ini terdiri dari 7 ekspresi, yaitu: netral, marah, sedih, kaget, senang, jijik, dan takut.

1.4 Tujuan Penelitian

Tujuan dari penelitian ini adalah:

1. Menghasilkan metode dan algoritma ekstrasi fitur ekspresi wajah berbasis morfologi matematika.
2. Menghasilkan model klasifikasi MNN yang dapat digunakan untuk pengenalan ekspresi wajah.
3. Menghasilkan *prototype* perangkat lunak sistem pengenalan ekspresi wajah.

1.5 Kontribusi dan Manfaat Penelitian

Penelitian ini bermanfaat dalam bidang keilmuan, karena penelitian ini mengembangkan algoritma ekstraksi fitur ekspresi wajah berdasarkan morfologi matematika untuk membangun model klasifikasi, dan menghasilkan model MNN untuk pengenalan ekspresi wajah.

Selain itu, penelitian ini juga bermanfaat dalam bidang teknologi, karena salah satu dari hasil penelitian ini adalah sebuah *prototype* sistem pengenalan ekspresi wajah, dan pengenalan ekspresi wajah juga dapat menjadi kecerdasan tambahan untuk robot. Penelitian ini bermanfaat untuk robot sosial, agar robot dapat berinteraksi dengan manusia secara tepat dengan keadaan yang ada.

Terakhir, penelitian ini bermanfaat untuk bidang psikologi, karena ekspresi wajah dapat menggambarkan keadaan seseorang, yaitu *mood* seseorang, di mana *mood* mempengaruhi berbagai aktivitas pada seseorang seperti pengambilan keputusan, nafsu makan, dan lain-lain.

Bab 2

Telaah Pustaka

2.1 Pengolahan Citra

2.1.1 Definisi Citra

Secara visual, citra adalah sebuah representasi informasi yang dapat dianalisis oleh manusia. Terdapat dua jenis informasi pada citra yaitu: informasi dasar, dan informasi abstrak. Informasi dasar (*low level image analysis*) adalah informasi yang dapat dianalisa secara langsung tanpa bantuan pengetahuan tambahan. Contohnya adalah warna, bentuk, dan tekstur. Sedangkan informasi abstrak adalah informasi yang tidak dapat dianalisa secara langsung atau membutuhkan pengetahuan tambahan. Contohnya adalah ekspresi wajah, sebuah kejadian. Kedua informasi ini tidak dapat dianalisa oleh komputer tanpa bantuan pengetahuan (Gonzalez, Woods, & Masters, 2009; Madenda, 2015; Poynton, 2003).

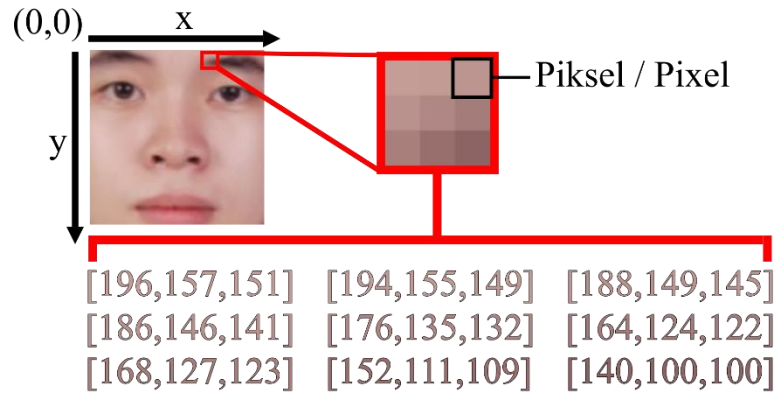
Secara matematis, sebuah citra dapat dinyatakan dalam sebuah fungsi dua dimensi $f(x, y)$, x dan y representasi koordinat spasial, dan f merupakan nilai intensitas warna pada koordinat (x, y) . f dapat dikatakan nilai intensitas warna dari sebuah citra yang dapat dikodekan dalam 24-bit (citra berwarna) dan pada umumnya format citra berwarna memiliki 3 nilai yaitu Red (R), Green (G), Blue (B) (dinamakan format RGB), tiap nilai RGB berkisaran antara 0 hingga 255 (Gonzalez et al., 2009; Madenda, 2015). Selain itu, citra dapat dikodekan dalam 8-bit (citra *grayscale*) yang hanya terdapat 1 nilai yang berkisaran antara 0 hingga 255 (*luminance*) (Poynton, 2003). Terakhir, citra dapat juga dikodekan dalam 1-bit (citra biner), yang terdapat dua kemungkinan nilai, yaitu 0 atau 1 (hitam/putih) (Bovik, 2009; Madenda, 2015). Sebuah citra $f(x, y)$ dapat direpresentasikan dalam bentuk matriks berukuran $M \times N$ seperti pada persamaan (2.1). Setiap elemen dari matriks pada persamaan (2.1) adalah piksel dari sebuah citra digital.

$$f(x, y) = \begin{bmatrix} f(0,0) & f(0,1) & f(0,2) & \dots & f(0, N-1) \\ f(1,0) & f(1,1) & f(1,2) & \dots & f(1, N-1) \\ f(2,0) & f(2,1) & f(2,2) & \dots & f(2, N-1) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ f(M-1,0) & f(M-1,1) & f(M-1,2) & \dots & f(M-1, N-1) \end{bmatrix} \quad (2.1)$$

2.1.2 Citra Berwarna

Citra berwarna adalah citra yang memiliki informasi warna secara visual, pada umumnya warna citra digital direpresentasikan menggunakan 3 warna dasar, yaitu: *Red* (R), *Green* (G), dan *Blue* (B) (disingkat RGB). Tiap warna dasar dikodekan dalam 8-bit (nilai berada diantara 0 hingga 255 dalam desimal) dan dapat dibentuk 2^{24} jenis warna berbeda. RGB merupakan warna aditif yang artinya warna dimulai dari hitam yang kemudian ditambahkan ketiga warna dasar sehingga membentuk dapat variasi warna (Gonzalez et al., 2009; Madenda, 2015).

Selain ruang warna RGB terdapat ruang warna lain yang dapat digunakan seperti HSV. HSV salah satu ruang warna yang merepresentasikan bagaimana manusia melihat warna. HSV memiliki 3 komponen, yaitu: *luminance*, *hue*, dan *saturation*. *Luminance* merepresentasi tingkat kecerahan warna. *Hue* merepresentasi warna dari piksel itu sendiri. *Hue* direpresentasikan dalam bentuk derajat dari 0° hingga 360° . 0° merupakan warna merah, 120° merupakan warna hijau, 240° merupakan warna biru, dan 360° kembali lagi ke warna merah. *Saturation* adalah tingkat tinggi rendahnya kandungan cahaya putih dalam suatu warna. Jika nilai *chrominance* semakin rendah maka warna terakit akan semakin memucat (ke arah putih). Jika sebaliknya, warna akan mendekati warna saturasi (*pure color*). (Gonzalez et al., 2009; Madenda, 2015; Poynton, 2003). Gambar 2.1 merupakan contoh citra wajah manusia yang memiliki warna dalam bentuk RGB.



Gambar 2.1. Citra berwarna (RGB)

2.1.3 Citra *Grayscale*

Citra *grayscale* adalah citra yang direpresentasikan hanya menggunakan 1 komponen, yaitu *luminance*. Standar pengkodean citra *grayscale* adalah 8-bit yang artinya nilai skala keabuan berkisar diantara 0 hingga 255 ($8_2 = 256_{10}$). Citra *grayscale* dapat diperoleh melalui konversi ruang warna RGB ke ruang warna lain seperti YC_bC_r atau HSV, di mana komponen Y dari YC_bC_r , dan V dari HSV merepresentasikan warna *grayscale* pada citra (Ahmad, Moon, & Shin, 2018; Madenda, 2015; Poynton, 2003). Persamaan (2.2) dan (2.3) menunjukkan metode konversi citra berwarna ke citra *grayscale*. Y-*grayscale* menggunakan persamaan (2.2), dan V-*grayscale* menggunakan persamaan (2.3).

$$Y = b_r R + b_g G + b_b B \quad (2.2)$$

$$V = \text{Maks}(R, G, B) \quad (2.3)$$

b_r , b_g , dan b_b merupakan bobot tiap *channel* yang menurut ITU-BT.601 (*International Telecommunication Union*) bobot yang direkomendasikan untuk tiap warna adalah $b_r = 0,299$, $b_g = 0,587$, dan $b_b = 0,114$ (Madenda, 2015; Poynton, 2003). Gambar 2.2 menunjukkan contoh konversi citra berwarna ke dua jenis *grayscale*, yaitu Y-*grayscale* dan V-*grayscale*.




2.1.4 Citra Biner

Citra biner merupakan citra yang mirip dengan citra *grayscale*, namun citra biner hanya memiliki dua tingkat keabu-abuan, yaitu 0 (hitam), dan 1 (putih). Citra biner dikodekan dalam 1-bit (Bovik, 2009; Gonzalez et al., 2009; Madenda, 2015). Pada umumnya citra biner didapatkan dari hasil konversi citra *grayscale*. Persamaan (2.4) dan (2.5) adalah *thresholding* yang dapat digunakan untuk konversi citra *grayscale* ke citra biner (Bovik, 2009). B merupakan citra biner. (x, y) menyatakan koordinat piksel. C merupakan citra *grayscale* yang digunakan untuk mendapatkan nilai citra biner. $f(C(x, y))$ merupakan fungsi logika yang membandingkan nilai piksel pada $C(x, y)$ dengan *threshold* (dalam persamaan disebut TH), *threshold* merupakan nilai yang ditentukan secara *manual* antara 0 hingga 255.

$$B(x, y) = f(C(x, y)) \quad (2.4)$$

$$f(x) = \begin{cases} 0 & \text{jika } x \geq TH \\ 1 & \text{jika } x < TH \end{cases} \quad (2.5)$$

Gambar 2.3 merupakan contoh citra biner yang didapatkan dari hasil konversi citra *grayscale* menggunakan *thresholding*. Terdapat dua contoh citra biner, yang pertama menggunakan *thresholding* 140, yang kedua menggunakan *thresholding* 50. Jika nilai *luminance* piksel pada citra *grayscale* dibawah atau sama dengan *threshold* maka nilai piksel pada citra biner adalah 0 (seperti contoh yang menggunakan *threshold* 140). Jika sebaliknya (nilai lebih dari *threshold*) maka nilainya adalah 1 (seperti contoh yang menggunakan *threshold* 50).

Contoh citra	Contoh Piksel			Format
	[168] [158] [139]	[166] [148] [124]	[161] [137] [113]	Y-Grayscale
	[1] [1] [0]	[1] [1] [0]	[1] [0] [0]	Biner dengan threshold 140 Cek batas: $113 \geq 140$ Karena dibawah batas (False) maka nilanya 0
	[1] [1] [1]	[1] [1] [1]	[1] [1] [1]	Biner dengan threshold 50 Cek batas: $113 \geq 50$ Karena diatas batas (True) maka nilanya 1

Gambar 2.3 Contoh citra biner

2.1.5 Konvolusi

Konvolusi merupakan teknik filter dalam pengolahan citra, konvolusi adalah sebuah proses integral atau penjumlahan dari sebuah dua fungsi komponen. Terdapat komponen penting dalam melakukan konvolusi, yaitu *kernel*. *Kernel* adalah sebuah matriks yang elementnya memiliki nilai baik positif maupun negatif, pada umumnya kernel memiliki jumlah baris dan kolom yang sama (seperti 3×3 , 5×5 , dst) (Kim, 2023). Gambar 2.4 menunjukkan contoh sebuah kernel 3×3 .

0	-1	0	-1	-1	-1	1/16	2/16	1/16
-1	4	-1	0	0	0	2/16	4/16	2/16
0	-1	0	1	1	1	1/16	2/16	1/16

Gambar 2.4. Contoh Kernel

Operasi konvolusi dinyatakan dalam simbol “*” atau \otimes . Operasi konvolusi antara matriks A dengan matriks B yang berukuran 3×3 dapat dinyatakan dalam persamaan (2.6), untuk lebih detail dapat melihat Gambar 2.5 yang menunjukkan contoh perhitungan secara visual.

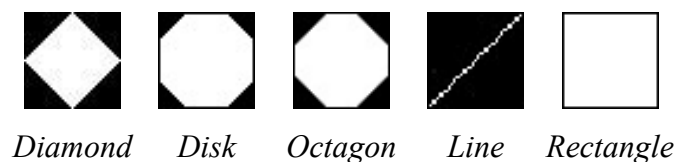
$$C = A * B = \sum_{n=-1}^1 \sum_{m=-1}^1 A(n, m) \times B(n, m) \quad (2.6)$$

Matriks A	Matriks B																			
<table border="1" style="border-collapse: collapse; text-align: center; width: 100px;"> <tr><td>168</td><td>166</td><td>161</td></tr> <tr><td>158</td><td>148</td><td>137</td></tr> <tr><td>139</td><td>124</td><td>113</td></tr> </table>	168	166	161	158	148	137	139	124	113	*	<table border="1" style="border-collapse: collapse; text-align: center; width: 100px;"> <tr><td>1</td><td>3</td><td>2</td></tr> <tr><td>4</td><td>6</td><td>5</td></tr> <tr><td>8</td><td>7</td><td>9</td></tr> </table>	1	3	2	4	6	5	8	7	9
168	166	161																		
158	148	137																		
139	124	113																		
1	3	2																		
4	6	5																		
8	7	9																		
		$C = A * B = 168 \times 1 + 166 \times 3 + 161 \times 2 + 158 \times 4$ $+ 148 \times 6 + 137 \times 5 + 139 \times 8$ $+ 124 \times 7 + 113 \times 9$																		

Gambar 2.5. Contoh operasi konvolusi

2.1.6 Morfologi Matematika

Morfologi matematika adalah teori untuk menganalisis bentuk dari sebuah citra. Dalam *image processing* pada komputer, morfologi dapat diaplikasikan untuk *image filtering* seperti menghilangkan *noise* pada citra, peningkatan tepi, memperbaiki intensitas cahaya yang tidak seimbang. Selain *image filtering*, morfologi dapat digunakan untuk *image segmentation* (Soille, 2004). Morfologi memiliki empat operasi yaitu dilasi (*dilation*), erosi (*erosion*), *opening*, dan *closing*. Operasi morfologi butuh komponen yang disebut *Structure Element* (SE). SE dapat direpresentasikan dalam bentuk matriks yang memiliki nilai antara 0 dan 1, di mana SE memiliki variasi bentuk dan ukuran (Yuqian Zhao, Weihua Gui, & Zhencheng Chen, 2006). Gambar 2.6 Contoh *Structure Element* menunjukkan contoh bentuk SE. Tujuan dari pengaplikasian morfologi matematika adalah untuk ekstraksi struktur yang relebant (fitur), dan menghilangkan struktur yang tidak relevant (*noise* atau objek yang tidak diinginkan) dari citra (Soille, 2004). Pada umumnya pengaplikasian morfologi dilakukan pada citra biner atau citra *grayscale* (Madenda, 2015). SE akan melintasi semua piksel yang ada pada citra berdasarkan dari bentuk SE (Madenda, 2015).



Gambar 2.6 Contoh *Structure Element*

2.1.6.1 Dilasi

Dilasi adalah salah satu operasi morfologi matematika yang digunakan untuk memperluas sebuah objek. Persamaan (2.7) menunjukkan persamaan dilasi (Goyal, 2011; Madenda, 2015; Soille, 2004).

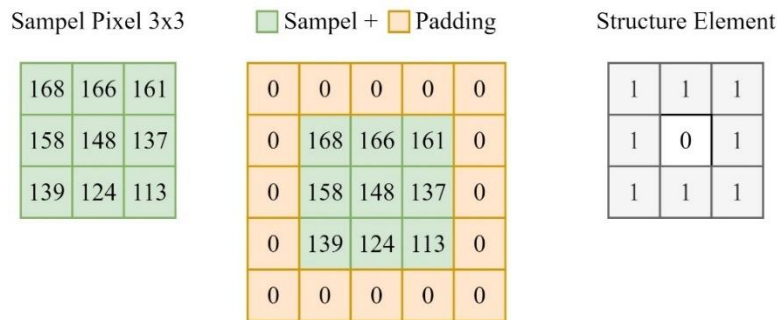
$$f \oplus B = \{ s \mid (B)_s \cap f \neq \emptyset \} \quad (2.7)$$

Berdasarkan persamaan 2.6. f adalah citra yang akan di-dilasi, B Merupakan SE yang bergeser sebesar s . Operasi dilasi pada citra biner berbeda dengan citra *grayscale*, di mana untuk mendapatkan nilai pada citra biner menggunakan persamaan (2.8) sedangkan citra *grayscale* dilakukan pada persamaan (2.9) (Madenda, 2015).

$$f \oplus B = \begin{cases} 1 & \text{jika } B_s = f = 1 \\ 0 & \text{jika } B_s = f = 0 \text{ atau } B_s \neq f \end{cases} \quad (2.8)$$

$$f \oplus B = \max_{i,j \in B} (B(i,j) * f(x+i, y+j)) \quad (2.9)$$

Gambar 2.7 menunjukkan contoh operasi dilasi, di mana digunakan contoh sampel piksel *grayscale* dari Gambar 2.2. Sampel piksel tersebut kemudian ditambahkan padding agar citra memiliki luas yang sama setelah proses dilasi. Terdapat sebuah SE 3×3 yang memiliki nilai 0 pada tengah (lihat Gambar 2.7 *structure element*) yang akan digunakan untuk operasi dilasi. Operasi dilakukan per piksel, dimulai dari koordinat (0,0) hingga (2,2). Tiap operasi pada piksel melibatkan area sekitarnya (seperti Gambar 2.7 kotak merah) dan area yang dilibatkan bergantung pada luas SE, kemudian dilakukan pengambilan nilai maksimum diantara nilai piksel yang berada pada SE yang bernilai 1. Gambar 2.8 menunjukkan hasil akhir dilasi, di mana citra sebelah kiri (a) adalah citra original dan sebelah kanan (b) adalah citra setelah dilasi. Jika dilihat proses dilasi membuat citra menjadi lebih tebal (terlihat jelas pada garis yang mengbung antar kotak dan lubang pada kotak mengecil).

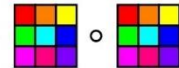


Perhitungan Dilasi

Perhitungan nilai piksel untuk koordinat (0,0) melibatkan area 3x3 (tergantung dari luas/dimensi Structure Element)

0	0	0	0	0
0	168	166	161	0
0	158	148	137	0
0	139	124	113	0
0	0	0	0	0

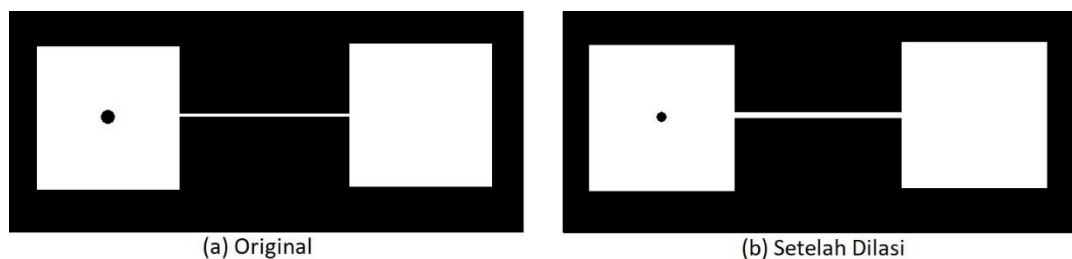
Dilakukan pengambilan nilai maksimum diantara nilai piksel citra yang berada pada SE yang bernilai 1.



$\text{maks}(166, 158, 148)$

→ 166 Nilai piksel yang baru

Gambar 2.7. Contoh perhitungan dilasi



(a) Original

(b) Setelah Dilasi

Gambar 2.8. (a) Sebelum dilasi dan (b) setelah dilasi

2.1.6.2 Erosi

Erosi adalah operasi morfologi matematika yang digunakan untuk mengikis sebuah objek. Operasi erosi berbanding terbalik dengan operasi dilasi, operasi erosi dapat menggunakan persamaan (2.10) (Goyal, 2011; Madenda, 2015; Soille, 2004).

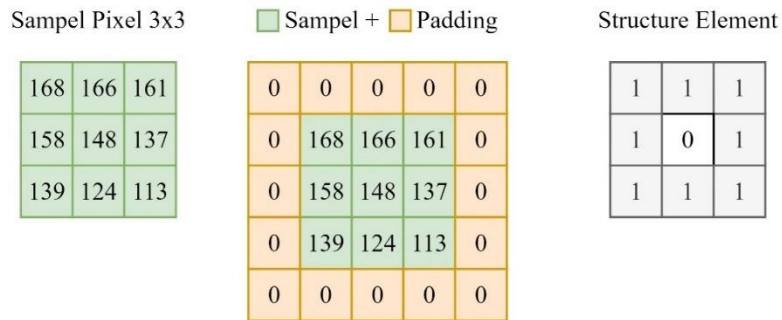
$$f \ominus B = \{s \mid (B)_s \subseteq f\} \quad (2.10)$$

Sama seperti dilasi, terdapat perbedaan operasi erosi antara citra biner dengan citra *grayscale*, di mana persamaan (2.11) digunakan untuk citra biner sedangkan persamaan (2.12) digunakan untuk citra *grayscale* (Madenda, 2015).

$$f \ominus B = \begin{cases} 1 & \text{jika semua nilai pada } B_s = 1 \\ 0 & \text{jika semua nilai pada } B_s \neq 1 \end{cases} \quad (2.11)$$

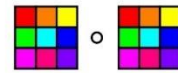
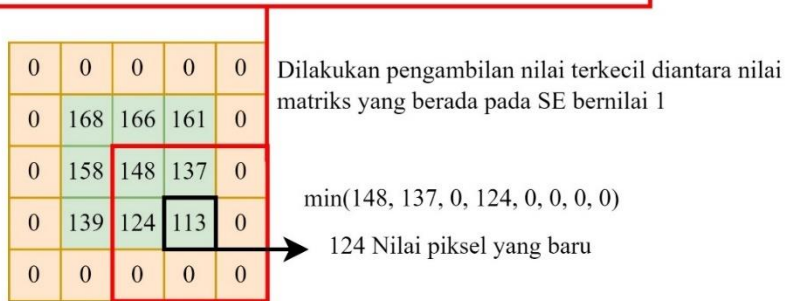
$$f \ominus B = \min_{i,j \in B} (B(i,j) * f(x+i, y+i)) \quad (2.12)$$

Gambar 2.9 menunjukkan contoh perhitungan erosi, di mana digunakan sampel piksel 3×3 sebagai contoh citra yang akan diproses. Citra kemudian ditambahkan *padding* agar ukuran citra tidak berkurang setelah proses. Pada proses erosi ini menggunakan *circular* SE seperti pada Gambar 2.9, di mana proses dilakukan per piksel mulai dari koordinat (0,0) hingga (2,2). Pada contoh dilakukan proses untuk piksel (2,2), di mana proses erosi melibatkan piksel tentangnya dan dilakukan pengambilan nilai terkecil diantara nilai piksel matriks yang berada pada SE yang bernilai 1. Gambar 2.10 menunjukan hasil akhir erosi, di mana citra sebelah kiri (a) adalah citra original dan sebelah kanan (b) adalah citra setelah erosi. Jika dilihat proses erosi mengikis citra (terlihat jelas pada garis yang menghubungkan antar kotak dan lubang pada kotak membesar).

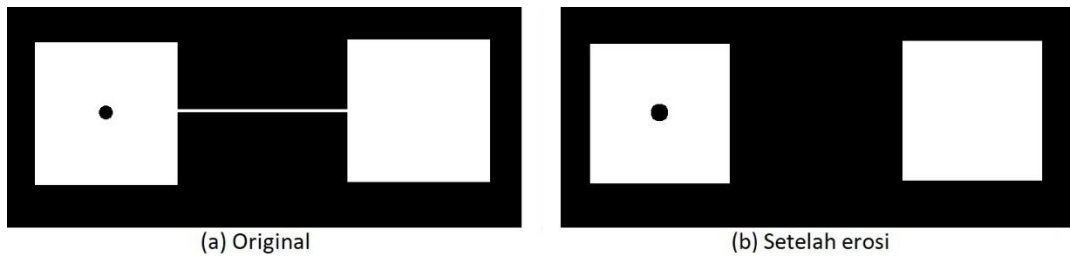


Perhitungan Erosi

Perhitungan nilai piksel untuk koordinat (0,0) melibatkan area 3x3 (tergantung dari luas/dimensi Structure Element)



Gambar 2.9. Contoh perhitungan erosi



(a) Original

(b) Setelah erosi

Gambar 2.10 Sebelum erosi (a) dan setelah erosi (b)

2.1.6.3 Closing

Closing adalah operasi morfologi matematika yang pada umumnya digunakan untuk menutup celah antara objek atau menutup lubang pada objek. Operasi *closing* memiliki dua tahap yaitu dilasi yang kemudian dilakukan erosi. Persamaan (2.13) menunjukkan persamaan untuk *closing* (Goyal, 2011; Madenda, 2015; Soille, 2004).

$$f \diamond B = (f \oplus B) \ominus B \quad (2.13)$$

Operasi *closing* pada umumnya digunakan pada citra biner, namun dapat juga digunakan pada citra *grayscale*. Gambar 2.11 menunjukkan contoh pengaplikasian *closing* pada sebuah citra, di mana objek berwarna putih merupakan piksel yang bernilai 1 sedangkan hitam adalah piksel yang bernilai 0. Jika dilihat sebelum dan sesudah operasi *closing*, operasi *closing* dapat menutup lubang pada kotak dan mempertahankan kotak dan jembatan antar kotak.



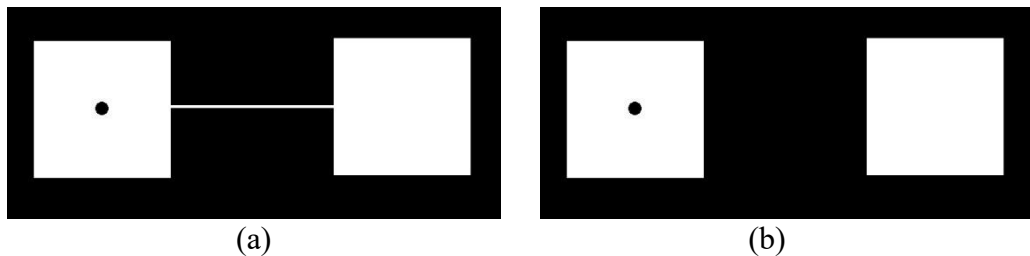
Gambar 2.11. Contoh citra (a) sebelum dan (b) sesudah *closing*

2.1.6.4 Opening

Opening adalah operasi morfologi matematika yang pada umumnya digunakan untuk menghilangkan objek-objek kecil (atau *noise*) pada citra. Sama seperti *closing* operasi *opening* ini memiliki dua tahap yaitu erosi kemudian dilasi. Persamaan (2.14) menunjukkan persamaan untuk *opening* (Goyal, 2011; Madenda, 2015; Soille, 2004).

$$f \circ B = (f \ominus B) \oplus B \quad (2.14)$$

Operasi *opening* pada umumnya digunakan pada citra biner, namun dapat digunakan juga untuk citra *grayscale*. Gambar 2.12 menunjukkan contoh pengaplikasian *opening* pada sebuah citra, di mana objek berwarna putih merupakan piksel yang bernilai 1 sedangkan hitam adalah piksel yang bernilai 0. Jika dilihat dari hasil, operasi *opening* dapat digunakan untuk memutus jembatan antar kotak dan mempertahankan bentuk kotak (dan juga lubang pada kotak)



Gambar 2.12. Contoh citra (a) sebelum dan (b) sesudah *opening*

2.1.6.5 Pendeteksian Tepi Objek Menggunakan Morfologi Matematika

Selain digunakan untuk pengikisan/penebalan objek, operator morfologi dapat digunakan untuk pendeteksian tepi objek (disebut Gradien Morfologi). Tepi dari sebuah objek dapat menggunakan selisih antara citra dengan operasi morfologi (seperti dilasi, erosi, *opening*) (Madenda, 2015; Shen et al., 2019). Persamaan (2.15), (2.16), (2.17) adalah selisih antara citra asli dengan dilasi (Gradien Morfologi Dilasi), citra asli dengan erosi (Gradien Morfologi Erosi), dan citra asli dengan *opening* (Gradien Morfologi *Opening*) secara berurutan.


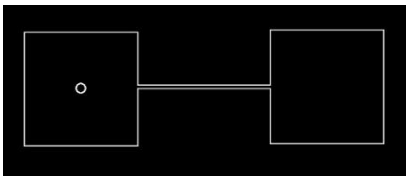
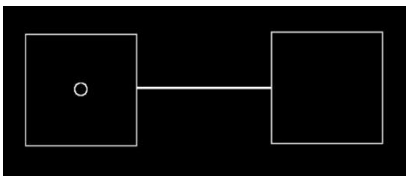

$$GMD = |(I \oplus B) - I| \quad (2.15)$$

$$GME = |I - (I \ominus B)| \quad (2.16)$$

$$GMO = |I - ((I \ominus B) \oplus B)| \quad (2.17)$$

Pada persamaan (2.15) hingga (2.17). I adalah citra asli, B adalah SE, GMD adalah Gradien Morfologi Dilasi, GME adalah Gradien Morfologi Erosi, dan GMO adalah Gradien Morfologi *Opening*. Tabel 2.1 menunjukan hasil dari tiap operasi gradien morfologi, di mana operasi gradien morfologi menggunakan persamaan (2.15), (2.16), (2.17) diimplementasikan pada sebuah citra biner yang memiliki objek dua kotak yang terhubung dengan garis (objek dan garis berwarna putih).

Tabel 2.1. Hasil gradien morfologi pada sebuah objek

Gradien Morfologi	Variable	Citra Sesudah Operasi									
Dilasi	<div>Citra original sebelum operasi</div> 										
Erosi	<div>Structure Element</div> <table><tr><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	1	1	1	1	1	1	1	1	1	
1	1	1									
1	1	1									
1	1	1									
Opening											

Berdasarkan dari hasil operasi pada Tabel 2.1, terdapat perbedaan hasil yang dapat dilihat secara kasat mata. Operasi GMD didapat tepi objek kotak dan juga garis yang menghubungkan antara dua kotak. Operasi GME tepi yang didapat hanya objek kotak, sedangkan untuk garis tetap sama seperti original, hal ini dikarenakan garis hilang saat proses erosi. *Opening* tepi tidak didapatkan dan hanya tersisa objek garis, hal ini dikarenakan *opening* melakukan proses erosi terlebih dahulu yang menyebabkan garis menghilang kemudian proses dilasi yang menebalkan objek ke ukuran sebelumnya, sehingga ketika proses pengurangan yang tersisa hanya objek garis.

2.1.7 Viola Jones Algorithm

Viola Jones Algorithm (VJA) adalah metode deteksi wajah yang dibuat oleh Paul Viola dan Michael Jones (Viola & Jones, 2001), *Viola Jones Algorithm* memiliki 2 komponen yang digunakan untuk mendeteksi wajah, pertama adalah *integral image*, kedua *Haar Basis Function*. *Integral image* adalah sebuah citra berisi total nilai warna piksel yang dimulai dari kiri atas hingga kanan bawah citra, di mana tiap piksel pada koordinat (x, y) merupakan nilai total warna piksel pada koordinat $(0,0)$ hingga (x, y) , perhitungan *integral image* pada koordinat (x, y) dilakukan menggunakan Persamaan (2.18) (Viola & Jones, 2001, 2004; Y.-Q. Wang, 2014).

$$I_{int}(x, y) = \sum_{0,0}^{x,y} I_{org}(x, y) \quad (2.18)$$

Berdasarkan persamaan (2.18), $I_{int}(x, y)$ merupakan nilai *integral image* yang pada koordinat (x, y) . Nilai pada koordinat tersebut didapatkan dengan cara penjumlahan nilai piksel pada citra original (I_{org}), mulai dari koordinat awal yaitu $(0,0)$, di mana koordinat tersebut dalam dunia *computer vision* menunjukkan ujung atas kiri citra hingga koordinat hingga (x, y) .

Gambar 2.13. merupakan contoh perhitungan *integral image* pada citra 4×4 , di mana nilai piksel *integral image* pada koordinat (x, y) merupakan jumlah nilai piksel citra mulai dari kiri atas citra (nilai pada koordinat (x, y) juga masuk dalam perhitungan) (Viola & Jones, 2001, 2004).

1	5	3	10
15	6	42	50
5	55	30	60
65	70	78	80

Original Image

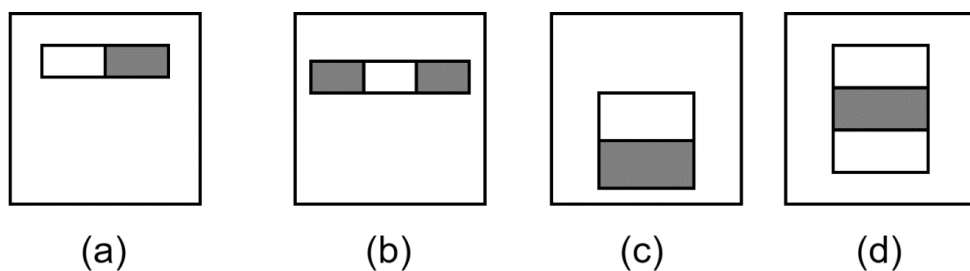
▶

1	6	9	19
16	27	72	132
21	87	162	222
86	222	375	575

Integral Image

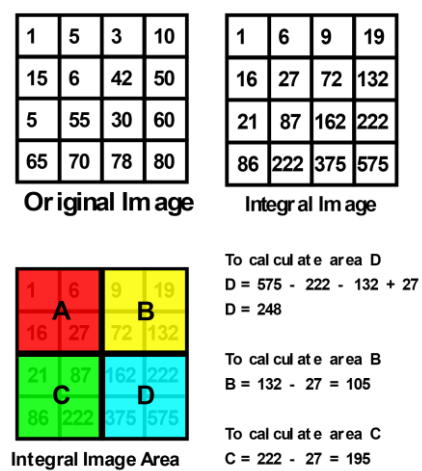
Gambar 2.13. Konversi citra ke citra integral

Komponen kedua adalah *Haar Basis Function*. *Haar Basis Function* digunakan untuk mendeteksi fitur wajah (seperti mata) pada sebuah *integral image* (Viola & Jones, 2004). Gambar 2.14 merupakan contoh *Haar Basis Function*, di mana nilai area putih dikurang dengan nilai area garis-garis (Viola & Jones, 2001, 2004; Y.-Q. Wang, 2014).



Gambar 2.14. Contoh *Haar Basis Function*

Perhitungan area *Haar Basis Function* dapat dilakukan seperti pada Gambar 2.15. Pada Gambar 2.15, untuk menghitung area D dilakukan dengan cara mengambil nilai piksel (4,4) dikurang nilai piksel (4,2) dikurang nilai piksel (2,4) ditambah (1,1). Jika dilakukan perbandingan hasil dengan menghitung nilai area D pada *integral image* dengan menghitung nilai area D menggunakan citra original memberikan hasil yang sama (Viola & Jones, 2001).



Gambar 2.15. Contoh perhitungan *Haar Basis Function*

2.1.8 Perubahan ukuran citra

Perubahan ukuran citra (*image resizing*) adalah salah satu metode yang digunakan untuk memperbesar atau memperkecil ukuran citra. Terdapat beberapa metode yang dapat digunakan seperti *nearest neighbor*, *bilinear*, dan *bicubic interpolation* (Gonzalez et al., 2009). Dalam penelitian ini digunakan method *bicubic interpolation* untuk merubah ukuran citra. *Bicubic interpolation* menggunakan persamaan (2.19).

$$h(X) = \begin{cases} (a + 2)|x|^3 - (a + 3)|x|^2 + 10 & -1 \leq |x| < 1 \\ a|x|^3 - 5a|x|^2 + 8a|x| - 4a & 1 \leq |x| < 2 \\ 0 & 2 \leq |x| \end{cases} \quad (2.19)$$

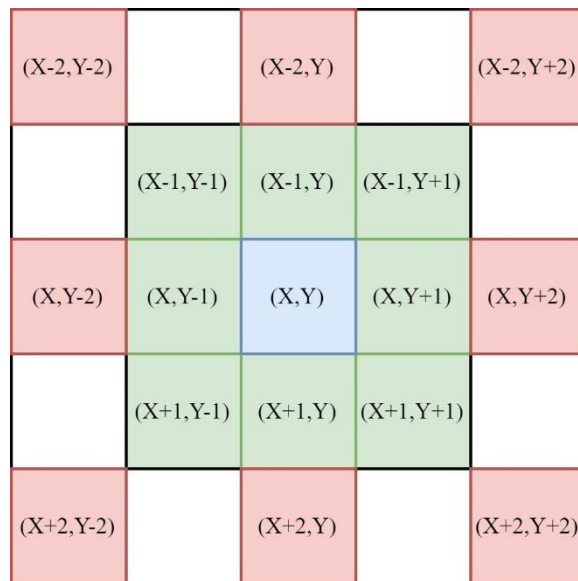
Berdasarkan dari persamaan (2.19), a merupakan koefisien yang berada diantara -0.5 dan -0.75. x merupakan nilai koordinat piksel tetangga.

2.2 Ekstrasi Fitur

2.2.1 Local Monotonic Pattern

Local Monotonic Pattern (disingkat LMP) merupakan salah satu algoritma ekstrasi fitur *histogram-based* (Eleyan, 2023). LMP merupakan pengembangan dari algoritma LBP (Mohammad & Ali, 2011). Algoritma LMP bekerja dengan cara membandingkan piksel secara monoton (sesuai dengan arah perbandingan). Terdapat tiga jenis piksel yang digunakan untuk perbandingan, yaitu: piksel tengah, piksel tetangga pertama, dan piksel tetangga kedua. Algoritma LMP memiliki dua perbandingan yang dilakukan bersamaan pada tiap perbandingan tetangganya (Babu & Rao, 2023; Mohammad & Ali, 2011).

Gambar 2.16. menunjukkan visualisasi tetangga pada proses suatu piksel LMP. Kotak berwarna biru merupakan piksel utama. Kotak berwarna hijau merupakan piksel tetangga radius pertama. Kotak berwarna merah merupakan piksel tetangga radius kedua.



Gambar 2.16. Visualisasi tetangga LMP

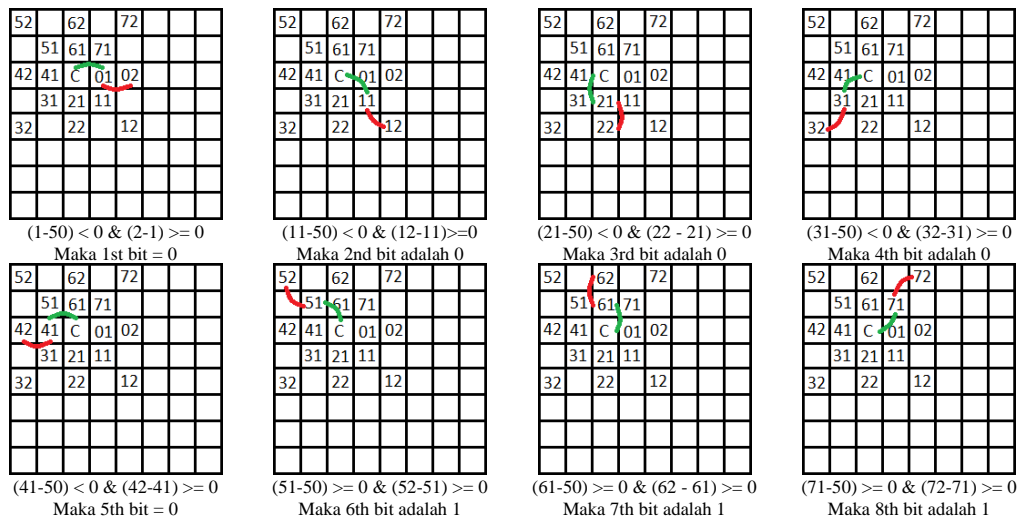
Berdasarkan Gambar 2.16 perbandingan pertama dilakukan pada piksel utama (kotak hijau) dengan piksel tetangga radius pertama, dan piksel radius pertama dengan piksel radius kedua. Sebagai contoh, perbandingan pertama terdapat pada piksel (x, y) dengan $(x, y + 1)$, dan $(x, y + 1)$ dengan $(x, y + 2)$. Perbandingan proses LMP beserta perbandingan nilai piksel dapat menggunakan persamaan (2.20), (2.21), dan (2.22) (Eleyan, 2023; Mohammad & Ali, 2011).

$$LMP_{P,R1,R2}(X,Y) = \sum_{n=0}^{P-1} S(I_{nR1} - I_c) * S(I_{nR2} - I_{nR1}) * 2^p \quad (2.20)$$

$$S(I_{nR2} - I_{nR1}) = \begin{cases} 1, x \geq 0 \\ 0, x < 0 \end{cases} \quad (2.21)$$

$$s(I_n - I_c) = \begin{cases} 1, x \geq 0 \\ 0, x < 0 \end{cases} \quad (2.22)$$

Berdasarkan persamaan (2.20), P merupakan piksel jumlah piksel tetangga, $R1$ merupakan radius untuk piksel tetangga pertama, $R2$ merupakan radius untuk piksel tetangga kedua, di mana jumlah tetangga dari piksel utama ditentukan oleh radius yang digunakan. I_c merupakan nilai dari piksel tengah/utama (*center*), I_{n1} merupakan nilai dari piksel tetangga pertama, I_{n2} merupakan nilai dari piksel tetangga kedua (Babu & Rao, 2023). Gambar 2.17 adalah contoh visualisasi proses LMP, di mana setelah melakukan perbandingan piksel tengah dengan tetangga pertama (ditandai dengan garis lengkung warna hijau) dan piksel tetangga pertama dengan tetangga kedua (ditandai dengan garis lengkung warna merah), didapatkan nilai biner dengan pola yaitu 11100000, dan jika dikonversi ke bilangan desimal menjadi 224.



Gambar 2.17. Contoh perhitungan LMP

2.2.2 Gabor Filter

Gabor *Filter* adalah salah satu metode ekstraksi fitur yang dapat digunakan untuk mendeteksi tepi atau mengekstraksi tekstur (Adak, 2013; Munawar et al., 2021). Gabor *Filter* terdiri dari komponen *real* dan *imaginary* yang ditunjukkan pada persamaan (2.23) dan (2.24) secara berurutan. Kedua komponen tersebut merepresentasikan arah secara ortogonal. Pada umumnya, dapat menggunakan salah satu dari kedua persamaan tersebut atau digabung kedalam *complex form* seperti pada persamaan (2.27) (Adak, 2013).

$$g(x, y, \lambda, \theta, \psi, \sigma, \gamma) = \exp\left(-\frac{x'^2 + \gamma y'^2}{2\sigma^2}\right) \cos\left(2\pi \frac{x'}{\lambda} + \psi\right) \quad (2.23)$$

$$g(x, y, \lambda, \theta, \psi, \sigma, \gamma) = \exp\left(-\frac{x'^2 + \gamma y'^2}{2\sigma^2}\right) \sin\left(2\pi \frac{x'}{\lambda} + \psi\right) \quad (2.24)$$



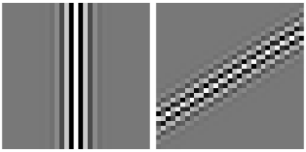

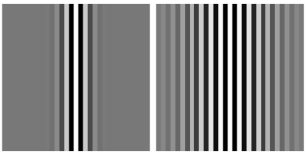
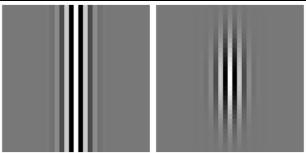
$$g(x, y, \lambda, \theta, \psi, \sigma, \gamma) = \exp\left(-\frac{x'^2 + \gamma y'^2}{2\sigma^2}\right) \exp\left(i\left(2\pi \frac{x'}{\lambda} + \psi\right)\right) \quad (2.25)$$

$$x' = x \cos \theta + y \sin \theta \quad (2.26)$$

$$y' = -x \sin \theta + y \cos \theta \quad (2.27)$$

Berdasarkan persamaan (2.21) hingga (2.23). λ panjang gelombang sinusoidal. θ orientasi garis normal ke paralel. ψ offset. σ adalah *gaussian envelope*. γ adalah rasio aspek spasial. Tabel 2.1 menunjukkan pengaruh parameter-parameter yang digunakan pada persamaan (2.22).

Tabel 2.2. Pengaruh parameter *Gabor Filter*

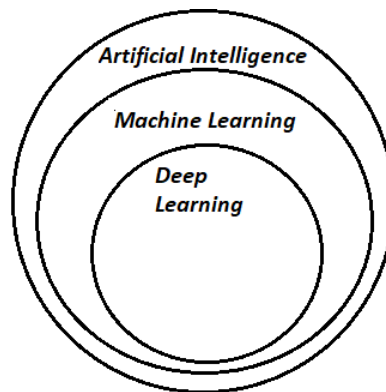
Nilai Parameter	Visualisasi Filter
(Nilai Awal) $\lambda = 2, \theta = 0, \psi = 0, \sigma = 2, \gamma = 0$	
$\lambda = 4$	 (Sebelum) (Sesudah)
$\theta = 45$	 (Sebelum) (Sesudah)
$\psi = 0$	 (Sebelum) (Sesudah)
$\sigma = 6$	 (Sebelum) (Sesudah)
$\gamma = 0.35$	 (Sebelum) (Sesudah)

Berdasarkan

Tabel 2.2 dapat dilihat bahwa parameter mempengaruhi bentuk filter. λ (*lambda*) mempengaruhi pola dari garis yang berada pada filter. θ (*theta*) mempengaruhi orientasi dari garis pada filter. ψ (*psi*) mempengaruhi urutan garis pada filter. σ (*sigma*) mempengaruhi jumlah pola garis pada filter. γ (*gamma*) mempengaruhi nilai intensitas piksel pada area tertentu pada filter.

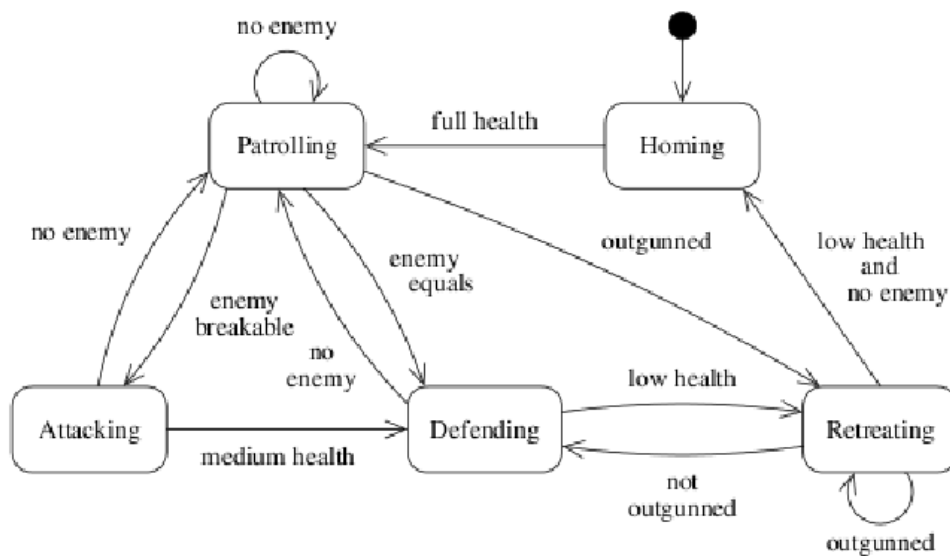
2.3 Artificial Intelligence

Artificial Intelligence (AI) atau bahasa Indonesia disebut kecerdasan buatan merupakan sebuah bidang umum mengenai kecerdasan buatan untuk mesin. Dalam membangun AI dibutuhkan pengetahuan, pengetahuan ini dapat dibangun dengan pembelajaran mendalam dengan mesin yang disebut *Machine Learning* (ML). Pembelajaran yang lebih mendalam dibandingkan disebut *Deep Learning* (DL) (Abiodun et al., 2018). Hubungan antara AI, ML, dan DL ditunjukkan pada Gambar 2.18 (Alzubaidi et al., 2021).



Gambar 2.18 Bidang kecerdasan buatan (Deepthi. B, Gupta, Rai, & Arora, 2022)

AI adalah kecerdasan yang paling sederhana. Salah satu contoh implementasi AI adalah *Tactical Decision Making* (TDM), di mana keputusan yang diambil TDM dilakukan secara sederhana menggunakan teknik seperti *finite state machines* seperti Gambar 2.19. TDM umumnya digunakan pada permainan strategi (Janiesch et al., 2021; Robertson & Watson, 2014; Shinde & Shah, 2018).



Gambar 2.19 Contoh *Finite state machine* (Madsen, Lucca, Daniel, & Adamatti, 2012)

Di dalam AI terdapat ML, di mana ML dapat mempelajari data yang diberikan, salah satu teknik dalam ML adalah *Support Vector Machine* (SVM). SVM dapat digunakan untuk mengklasifikasi suatu data dengan mencari pola dari suatu data (Awad & Khanna, 2015)

Di dalam ML terdapat DL, di mana DL dapat mempelajari data yang lebih kompleks dibandingkan ML. Perbedaan antara ML dan DL adalah DL dapat mempelajari fitur secara otomatis. Salah satu metode DL yang dapat digunakan adalah *Artificial Neural Network* (ANN). ANN merupakan kecerdasan buatan yang mempelajari data lebih dalam (disebut *deep learning*) (O'Shea & Nash, 2015). Terdapat beberapa kategori pada ANN seperti *Deep Supervised Learning* (dalam *Deep Supervised Learning* terdapat *Convolutional Neural Network* (CNN), *Recurrent Neural Network* (RNN)), *Deep Semi-supervised Learning* (RNN, *Gate Recurrent Unit* (GRU)), *Deep Unsupervised Learning* (*Long-Short Term Memory Neural Network* (LSTM), dan RNN juga dapat termasuk dalam *unsupervised*), dan *Deep Reinforced Learning* (Alom et al., 2019).

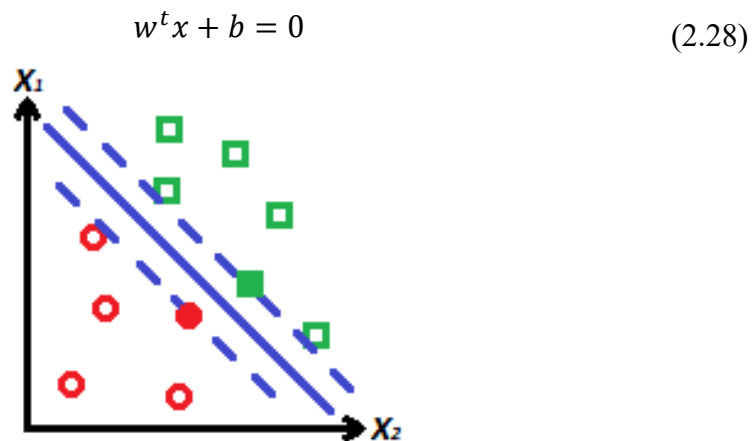
2.3.1 Machine Learning

Machine Learning (ML) adalah salah satu cabang dari AI. ML merupakan sebuah proses komputasi data untuk melakukan tugas tertentu. Proses komputasi tersebut merupakan sebuah proses pembelajaran secara berulang yang membuat mesin menjadi lebih baik dalam melakukan tugas yang diberikan (El Naqa & Murphy, 2015). Performa dari model ML sangat berpengaruh terhadap kualitas *dataset* yang digunakan, di mana kualitas yang dimaksud antara lain *noise* yang ada pada *dataset*, atau tingkat kompleksitas *dataset* (Janiesch et al., 2021). Oleh karena itu perlu dilakukan ekstraksi fitur.

2.3.1.1 Support Vector Machine

Support Vector Machine (SVM) adalah salah satu ML yang digunakan untuk mencari pola dari data dan mengklasifikasi data. SVM bekerja dengan cara membuat *hyperplane* yang memisahkan dua kategori data (Awad & Khanna, 2015). SVM dapat digunakan untuk klasifikasi ekspresi wajah dengan menggunakan teknik *regional histogram* seperti penelitian (Deshmukh et al., 2016; Eleyan, 2023; Lmp, 2011). Secara garis besar berdasarkan kompleksitas kasus, terdapat tiga jenis kasus pada SVM yaitu: *linear separable*, *non-linear separable*, dan *non-separable* (H. Wang, Xiong, Yao, Lin, & Ren, 2017).

Case of linear separable merupakan salah satu jenis kasus yang memiliki kompleksitas paling sederhana, di mana dalam kasus ini kategori data dapat dipisahkan secara *linear*. Optimal *hyperplane* dapat didefinisikan dengan persamaan (2.28), di mana w merupakan bobot vektor. x merupakan nilai *input* data. b merupakan *threshold*, bobot vektor dilakukan transpos agar dapat dilakukan perkalian dengan x . Gambar 2.20 merupakan contoh kasus di mana data dapat dipisahkan secara *linear*.



Gambar 2.20. Contoh SVM dengan dua kategori data

Berdasarkan Gambar 2.20, terdapat dua kategori data yang berbeda, kategori data pertama berbentuk bulat dan berwarna merah, untuk data kedua berbentuk kotak dan berwarna hijau. Data tersebut dipisahkan oleh sebuah *hyperplane* (garis biru), di mana arah dari *hyperplane* bergantung pada *margin level* (garis putus-putus biru dan biasa disebut *support vector* atau *C* dalam pemrograman) (Awad & Khanna, 2015; H. Wang et al., 2017).

Case of non-linearly separable merupakan salah satu jenis kasus di mana data tidak dapat dipisahkan dengan *linear* (Bhavsar & Panchal, 2012). Jika kasus data tidak dapat dipisahkan secara *linear*, dapat menggunakan *kernel* yang dapat mentransformasi data ke dimensi yang lebih tinggi (disebut *kernel space*). Terdapat beberapa kernel yang dapat digunakan dalam SVM, yaitu (Awad & Khanna, 2015):

- *Linear kernel*

$$K(x, y) = x \cdot y \quad (2.29)$$

- *Gaussian Radial Basis Function (RBF)*

$$K(x, y) = \exp \frac{||x - y||^2}{\sigma^2} \quad (2.30)$$

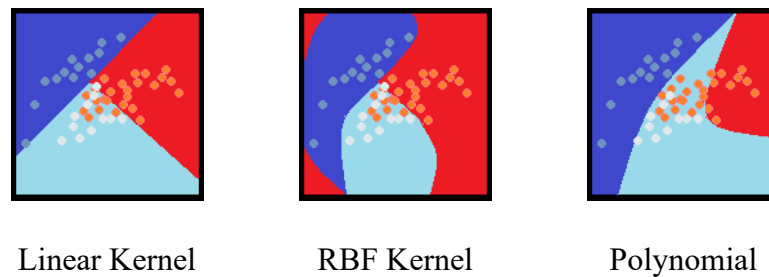
- *Hyperbolic Tangent (Sigmoid)*

$$K(x, y) = \tanh(\beta xy + \gamma) \quad (2.31)$$

- *Polynomial Kernel*

$$K(x, y) = (xy + c)^d, d > 0 \quad (2.32)$$

Berdasarkan persamaan (2.29) hingga (2.32). x, y merupakan *input* data point. σ adalah parameter yang mengatur lebar dari gaussian. β adalah bobot yang mengatur hubungan antara data point pada dimensi tinggi. c adalah konstan yang mengatur margin (disebut juga soft-margin SVM). d adalah derajat dari polynomial yang berpengaruh terhadap curve dari decision boundary (Awad & Khanna, 2015; Ben-Hur, Ong, Sonnenburg, Schölkopf, & Rätsch, 2008). Gambar 2.21 menunjukkan contoh visualisasi 3 kernel SVM pada sebuah *dataset* yang memiliki 3 kategori.



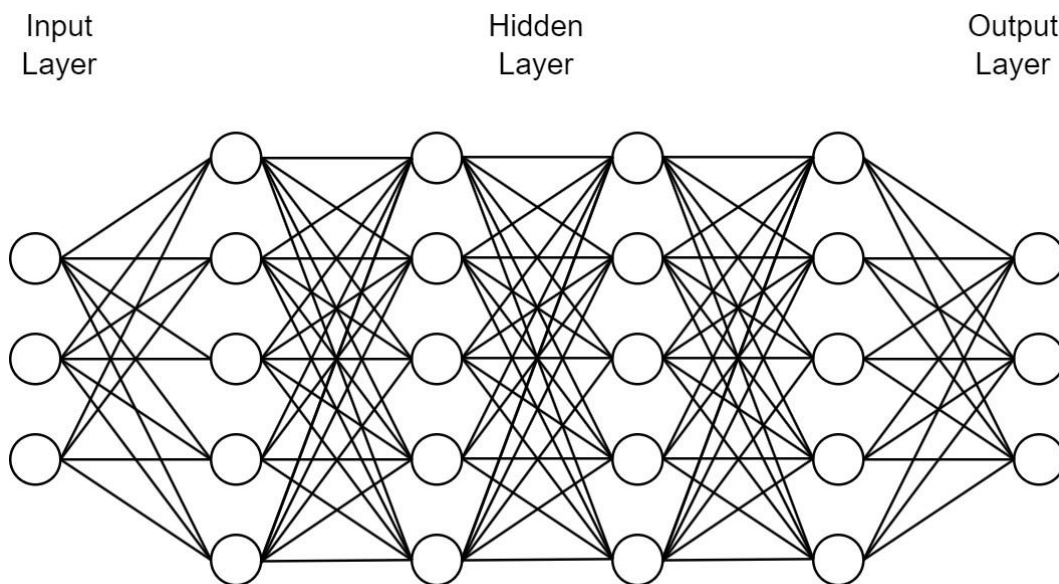
Gambar 2.21. Contoh visualisasi SVM dengan kernel berbeda (Pedregosa et al., 2011)

2.3.2 Deep Learning

Deep Learning (DL) bagian dari ML. DL dapat mengatasi kelemahan yang ada pada ML, yaitu ketergantungan pada kualitas *dataset* yang digunakan. DL dapat mengatasi *noise* dan juga mempelajari data yang lebih kompleks dibandingkan ML, oleh karena itu DL mengungguli ML (Janiesch et al., 2021).

2.3.2.1 Artificial Neural Network

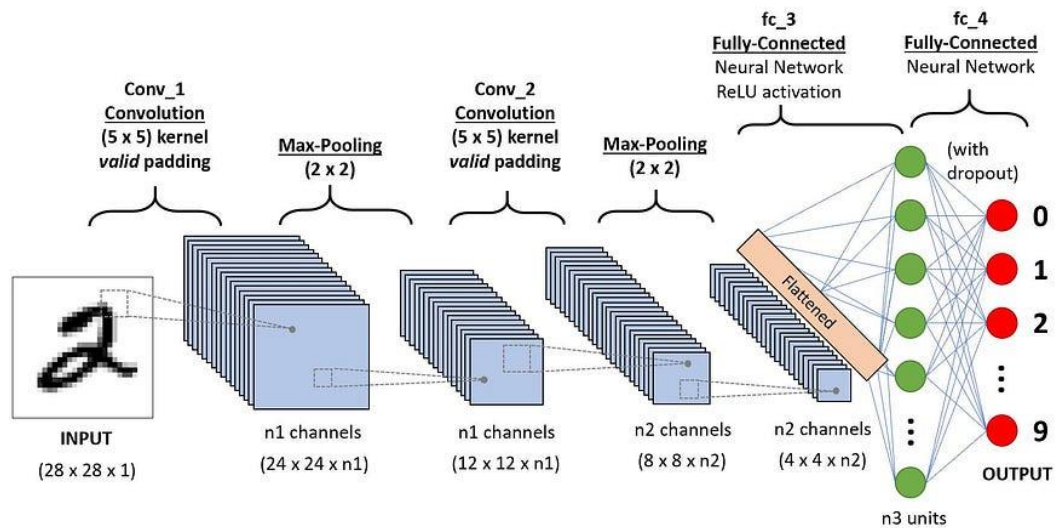
Artificial *Neural Network* (ANN) adalah salah satu model DL, di mana model dari ANN itu sendiri mirip dengan jaringan saraf yang ada pada manusia. ANN terdiri dari lapisan-lapisan yang memiliki tugas masing-masing dan tiap lapisan memiliki saraf tiruan (artificial *neuron*, dan sering disebut *neuron* saja). Tiap lapisan terhubung dengan lapisan sebelumnya, lapisan ini sering disebut dengan *hidden layer* atau *Fully-Connected Layer* (FC Layer). Gambar 2.22 menunjukkan sebuah model ANN (Maind & Wankar, n.d.).



Gambar 2.22. ANN (Bre, Gimenez, & Fachinotti, 2018)

2.3.2.2 Convolutional Neural Network

Convolutional Neural Network (CNN) adalah salah satu model *Deep Learning* yang pada umumnya dimanfaatkan pada bidang *computer vision* (Janiesch et al., 2021). Arsitektur dari CNN tidak berbeda jauh dari ANN. Perbedaan arsitektur dari kedua model adalah pada CNN memiliki *Convolution Layer* yang dapat melakukan ekstraksi fitur citra (Li, Liu, Yang, Peng, & Zhou, 2022). Gambar 2.23 menunjukkan contoh arsitektur CNN secara garis besar.



Gambar 2.23. Arsitektur CNN (Montero, Gundul, Frayco, & Jr, 2023)

2.3.2.2.1 Lapisan CNN

CNN memiliki beberapa jenis *layer* yang dapat digunakan dan masing-masing *layer* memiliki tugas atau fungsi masing-masing, yaitu:

1. *Input Layer*

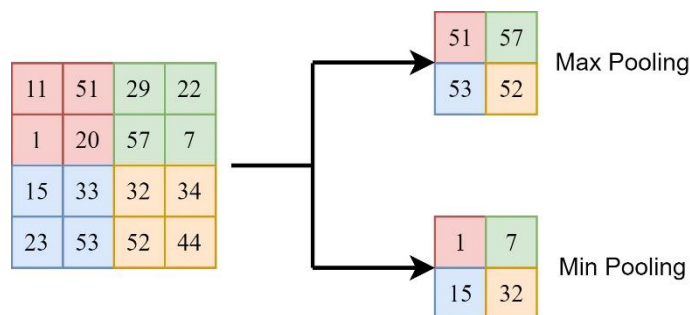
Lapisan pertama pada CNN yang bertugas untuk menerima *input* data (pada CNN *input* berupa citra).

2. *Convolution Layer*

Lapisan yang bertugas untuk melakukan ekstraksi fitur. Pada umumnya *convolution layer* merupakan paling pertama dari *hidden layer*, di mana lapisan ini menerima citra dari *input layer* yang kemudian dilakukan ekstraksi fitur menggunakan operasi konvolusi (Alzubaidi et al., 2021). Terdapat beberapa parameter (disebut *hyperparameter*) yang dapat dimodifikasi untuk mengoptimalkan model seperti *depth*, *stride*, *zero-padding* (O'Shea & Nash, 2015).

3. *Pooling Layer*

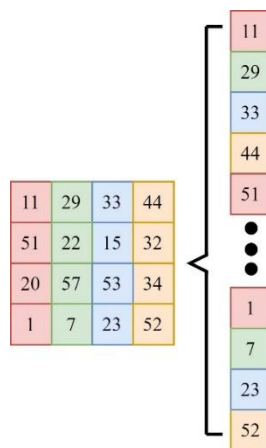
Lapisan yang bertugas untuk melakukan *downsampling*. *Downsampling* adalah teknik untuk mengurangi dimensi citra namun tetap mempertahankan fitur yang ada pada citra (Alzubaidi et al., 2021; O'Shea & Nash, 2015). Terdapat tiga jenis *pooling* yang dapat digunakan yaitu: *Max Pooling* (mengambil nilai tertinggi), *Average Pooling* (mengambil nilai rata-rata), dan *Minimum Pooling* (mengambil nilai terendah). Gambar 2.24 menunjukkan contoh operasi *pooling*.



Gambar 2.24. Contoh operasi *pooling*

4. *Flatten Layer*

Lapisan yang bertugas untuk mengubah citra menjadi *feature vector* (berupa 1 dimensi). Lapisan ini berada diantara *convolution/pooling layer* dan *fully-connected layer*. Tujuan dari lapisan ini adalah agar dapat terhubung dengan *layer* selanjutnya (Alzubaidi et al., 2021) (yaitu *Fully-Connected Layer* atau *Output Layer* jika tidak menggunakan *Fully-Connected Layer*). Gambar 2.25 menunjukkan contoh *flatten* sebuah matriks.



Gambar 2.25. Contoh *Flatten*

5. *Fully-Connected Layer*

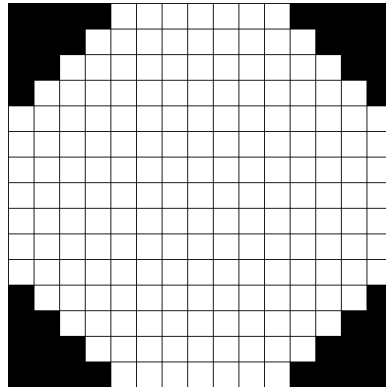
Lapisan yang bertugas untuk mempelajari fitur yang ada untuk menentukan (dan mempelajari) kelas dari *input*. Lapisan ini memiliki *neuron*, di mana tiap *neuron* terhubung dengan *layer* sebelumnya (yaitu *flatten layer*) (Alzubaidi et al., 2021).

6. *Output Layer*

Lapisan akhir yang memberikan kelas pada objek yang telah dipelajari (Alzubaidi et al., 2021).

2.3.2.3 Morphological Neural Network

Morphological Neural Network (MNN) merupakan bagian dari metode *deep learning*, konsep dari MNN mirip dengan CNN, perbedaan MNN dengan CNN adalah MNN memanfaatkan operasi morfologi untuk ekstraksi fitur citra (Shen et al., 2019). Penelitian tentang konsep dari MNN sendiri sudah ada sejak 1990 (Davidson & Ritter, 1990). Walaupun demikian, penelitian dari MNN masih sedikit dilakukan dan dapat dikembangkan. Pada penelitian 2022 oleh Shen et al. Dibatasi sebuah model MNN yang memanfaatkan *opening* dari morfologi matematika. Gambar 2.26 menunjukkan visualisasi dari *disk SE*, dan Gambar 2.27 menunjukkan pengaplikasian *opening* pada sebuah citra biner menggunakan *disk SE*. Menurut penelitian (Shen et al., 2019), morfologi cocok digunakan untuk menganalisis bentuk.

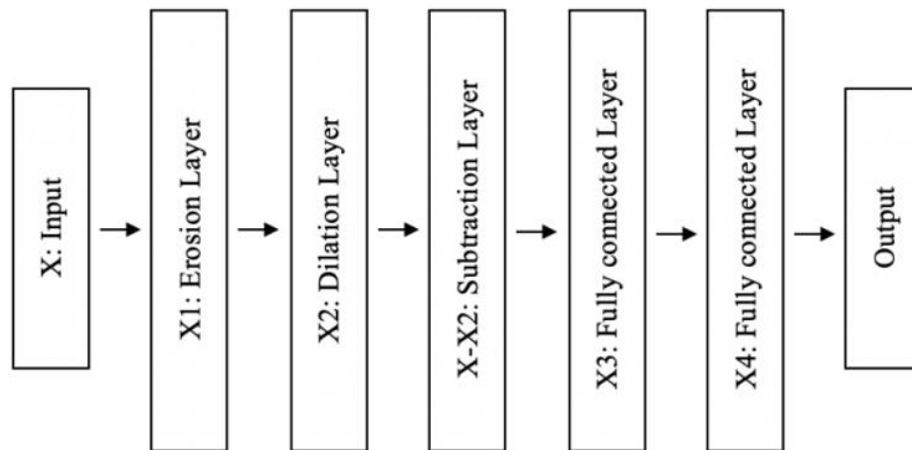


Gambar 2.26. *Disk Structure Element*



Gambar 2.27 Contoh aplikasi *opening* menggunakan *disk SE* (Shen et al., 2019)

Kemudian dibuat sebuah model MNN menggunakan teknik *opening* (dan disk SE), Gambar 2.28 menunjukkan model MNN yang dibuat pada penelitian (Shen et al., 2019), di mana lapisan pertama adalah *input*, yang diikuti dengan lapisan erosi, dilasi, *subtraction*, FC-layer, dan terakhir *output layer*. Pada *subtraction layer* dilakukan pengurangan antara citra asli dengan citra *opening*.



Gambar 2.28 Model MNN pada penelitian 2019 (Shen et al., 2019)

2.3.3 Metrik model klasifikasi

Dalam penelitian ini digunakan sebuah model untuk melakukan klasifikasi beberapa kelas (*multi-class classification*). Terdapat konsep yang dapat digunakan untuk mengukur performa dari sebuah model yang melakukan *multi-class classification*, yaitu *Confusion Matrix* (CM). CM merupakan tabel silang yang menyimpan jumlah kemunculan dua nilai, yaitu nilai klasifikasi aktual dan nilai klasifikasi prediksi.

Gambar 2.29 menunjukkan contoh *confusion matrix* untuk 3 kelas prediksi (Grandini, Bagli, & Visani, 2020).

	<i>Predicted</i>			
	Kelas	A	B	C
	A	6	2	1
	B	3	7	1
	C	1	1	8

Gambar 2.29. Contoh *Confusion Matrix*

Berdasarkan Gambar 2.29, terdapat 3 kelas yaitu A, B, dan C. Sel yang berwarna hijau merupakan prediksi yang benar, yang umumnya disebut *True Positive* (TP) dan *True Negative* (TN) tergantung dari sudut pandang. Sel yang berwarna orange merupakan prediksi yang salah, umumnya disebut *False Positive* (FP) dan *False Negative* (FN) tergantung dari sudut pandang. Gambar 2.30 menunjukkan sudut pandang yang dimaksud, di mana jika kelas A sebagai kelas pusat, maka nilai prediksi berada diantara kategori TP atau FP, dan untuk kelas B dan C berada pada kategori TN atau FN. Selain itu, jika B sebagai kelas pusat, maka kategori prediksi berada diantara TP atau FP, selain kelas itu kategori berada pada TN atau FN.

	<i>Predicted</i>			
	Kelas	A	B	C
	A	TP	FN	FN
	B	FP	TN	FN
	C	FP	FN	TN

(a)

	<i>Predicted</i>			
	Kelas	A	B	C
	A	TN	FP	FN
	B	FN	TP	FN
	C	FN	FP	TN

(b)

Gambar 2.30. Contoh TP TN FP FN

CM dapat dihitung beberapa nilai yang dapat digunakan untuk mengetahui performa dari sebuah model. Nilai-nilai tersebut yaitu (Grandini et al., 2020):

1. *Precision*

Presision atau presisi adalah rasio antara prediksi benar kelas tersebut (TP) yang dibagi dengan total prediksi benar. Perhitungan presisi menggunakan persamaan (2.33)

$$Precision = \frac{TP}{TP + FP} \quad (2.33)$$

2. *Recall*

Recall adalah nilai prediksi benar kelas tersebut (TP) yang dibagi dengan TP dan prediksi salah yang seharusnya positif (FN), perhitungan recall menggunakan persamaan (2.34)

$$Precision = \frac{TP}{TP + FN} \quad (2.34)$$

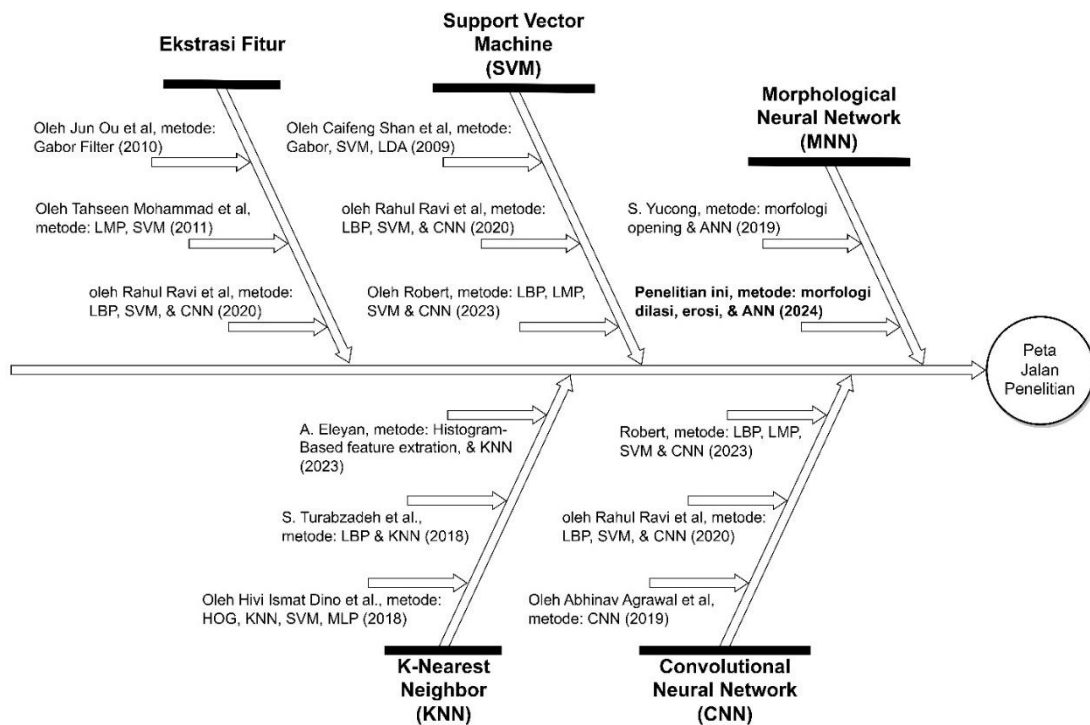
3. *Accuracy*

Accuracy Merepakan rasio antara total prediksi benar dengan total keseluruhan prediksi (baik benar atau salah). Perhitungan akurasi dapat menggunakan persamaan (2.35)

$$Precision = \frac{TP + TN}{TP + TN + FP + FN} \quad (2.35)$$

2.4 Peta Jalan Penelitian

Gambar 2.31 menunjukkan peta jalan penelitian yang terkait dengan topik penelitian yang sudah dilakukan oleh anggota peneliti dalam periode tahun 2019 – 2023.



Gambar 2.31. Peta jalan penelitian

Pada tahun 2010, dilakukan penelitian tentang pengenalan enam jenis ekspresi wajah otomatis menggunakan Gabor *filter* dan K-Nearest Neighbor (KNN) (Ou, Bai, Pei, Ma, & Liu, 2010). Pada tahun 2009, dilakukan pembelajaran komprehensif tentang pengenalan ekspresi wajah menggunakan *Local Binary Pattern* (LBP) yang digabung *AdaBoost* untuk melakukan ekstrasi fitur. Hasil dari LBP+*Adaboost* digunakan sebagai *input* model SVM yang digunakan untuk mengenali ekspresi wajah (Shan et al., 2009). Pada tahun 2011, dilakukan pengembangan metode ekstrasi fitur *Local Monotonic Pattern* (LMP) untuk pengenalan ekspresi wajah. Hasil dari LMP kemudian digunakan sebagai *input*

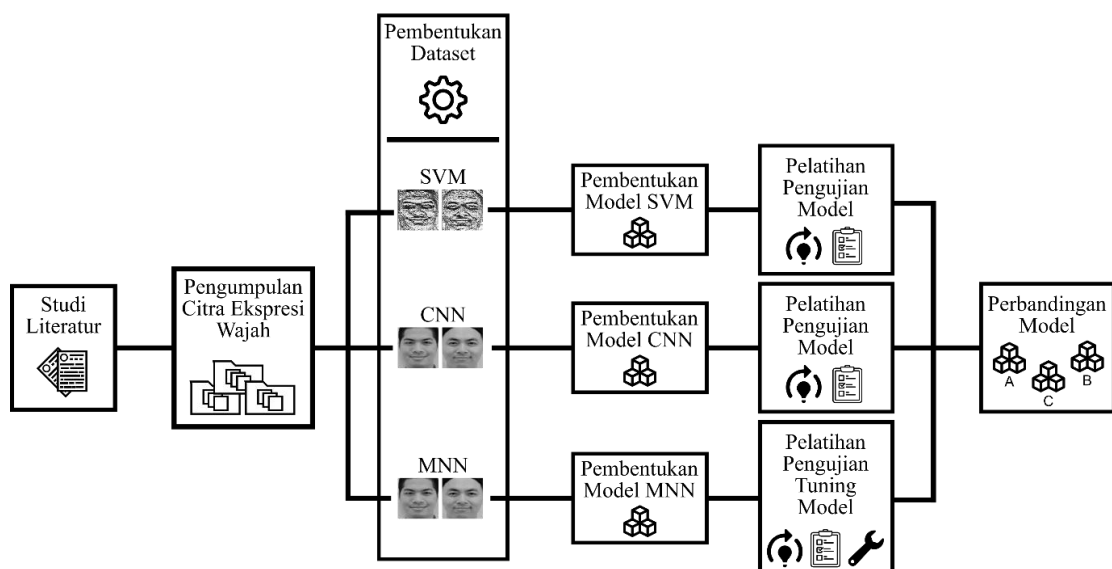
model SVM. Pada tahun 2019, dikembangkan sebuah model DL MNN yang memanfaatkan *opening* dari morfologi matematika untuk mengklasifikasi angka dan rambu lalu lintas (Shen et al., 2019). Pada tahun 2020, dilakukan penelitian pengenalan ekspresi wajah menggunakan *Local Binary Pattern* (LBP) sebagai ekstraksi fitur, CNN dan SVM sebagai model *Machine Learning* (Ravi, Yadhukrishna, & Prithviraj, 2020). Pada tahun 2023, dilakukan pengenalan ekspresi wajah menggunakan *histogram-based feature descriptor* (untuk ekstraksi fitur), dan KNN sebagai model pengenalan wajah (Eleyan, 2023). Pada tahun yang sama (2023), dilakukan penelitian pembentukan skenario *dataset* menggunakan LBP, *Local Monotonic Pattern*, dan Viola Jones (VJA) sebagai ekstraksi fitur, dan CNN, SVM sebagai model pengenalan ekspresi wajah. Pada tahun ini 2024, akan diajukan penelitian “PENGEMBANGAN MODEL KLASIFIKASI *MORPHOLOGICAL NEURAL NETWORK* UNTUK SISITEM PENGENALAN EKSPRESI WAJAH”.

Bab 3

Metode Penelitian

3.1 Alur Penelitian

Gambar 3.1 menunjukkan metode penelitian. Terdapat 5 tahap utama yang akan dilakukan, yang pertama adalah studi literatur untuk menyusun bab 1 dan bab 2. Tahap kedua adalah pengumpulan citra ekspresi wajah (data citra berupa data primer dan data sekunder). Tahap ketiga adalah pembentukan *dataset* untuk tiap model (SVM, CNN, dan MNN), skenario pembentukan *dataset* dilakukan berdasarkan pada penelitian (Robert, 2023). Pada tahap keempat dilakukan pembentukan model, khusus untuk SVM dan CNN menggunakan model pada penelitian (Robert, 2023), sedangkan MNN menggunakan usulan pada penelitian ini. Tahap terakhir adalah pelatihan dan pengujian untuk semua model (SVM, CNN, MNN), terdapat tahap *parameter tuning* untuk tiap model. Kemudian semua performa dari tiap model akan dibandingkan satu sama lain, dan juga dianalisis pada bab 4.



Gambar 3.1. Metode penelitian

3.2 Pengumpulan Citra Ekspresi Wajah

Citra ekspresi wajah dikumpulkan secara langsung oleh peneliti (data primer) dan juga menggunakan data yang dikumpulkan oleh peneliti lain (data sekunder). Terdapat 7 ekspresi wajah yang akan digunakan dalam penelitian ini, yaitu: marah, jijik, menghina, senang, sedih, kaget, dan netral (tanpa ekspresi). Gambar 3.2 menunjukkan contoh 7 ekspresi wajah manusia yang digunakan penelitian ini.



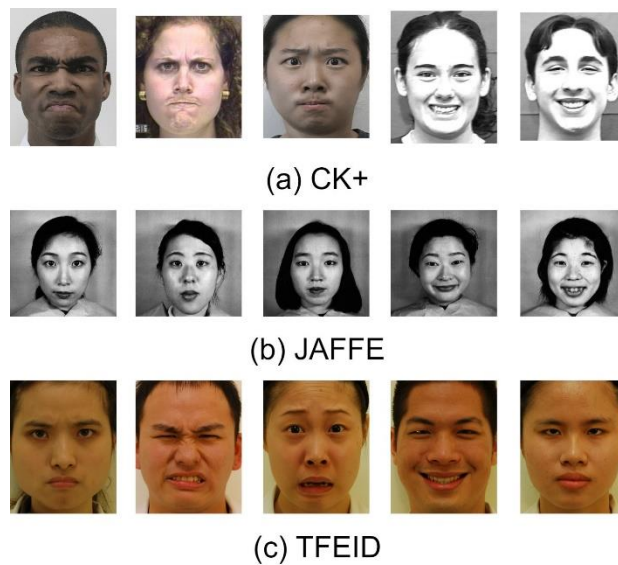
Gambar 3.2. Contoh 7 jenis ekspresi wajah

Dataset primer akan dilakukan pengambilan citra ekspresi wajah mahasiswa Universitas Gunadarma baik pria maupun wanita. Pengambilan akan dilakukan dari beberapa sudut pandang guna menambah variasi *dataset*. Gambar 3.3 menunjukan contoh *dataset* primer dari berbagai sudut pandang.



Gambar 3.3. Contoh *dataset* primer

Dataset sekunder digunakan *dataset* yang telah digunakan umum oleh peneliti lain terkait pengenalan ekspresi wajah. Terdapat beberapa *dataset* yang umum digunakan dalam penelitian ekspresi wajah. Pertama, Extended *Cohn-Kanade* (CK+) yang berisi citra ekspresi wajah pria dan wanita dari berbagai etnis dengan resolusi tinggi (Kanade, Cohn, & Tian, 2000; Lucey et al., 2010). Kedua, *Taiwanese Facial Expression Image Dataset* (TFEID) yang berisi citra ekspresi wajah pria dan wanita dari etnis Taiwan (Chen & Yen, 2007). Ketiga, *Japanese Female Facial Expression* (JAFFE) yang terdiri dari citra ekspresi wajah wanita etnis Jepang (Lyons, 2021; Lyons, Kamachi, & Gyoba, 2020). Gambar 3.4 menunjukkan contoh citra *dataset* CK+ (a), JAFFE (b), dan TFEID (c).



Gambar 3.4. Contoh *dataset* sekunder

Tabel 3.1 menunjukkan detail dari tiap *dataset*, mulai dari jumlah citra dari tiap kelas serta ruang warna dan ukuran citra. CK+ memiliki jumlah yang tidak seimbang pada kelas *neutral* dan memiliki ruang warna campur antara RGB dan Gray, dengan ukuran citra dikisaran 640×490. JAFFE *dataset* memiliki jumlah citra pada tiap kelas yang seimbang dengan perbedaan diantara 0 hingga 2 citra, ukuran citra 256×256, dan ruang warna *grayscale*. TFEID juga memiliki jumlah citra yang seimbang di tiap kelas, ukuran citra dikisaran 481×600, ruang warna RGB.

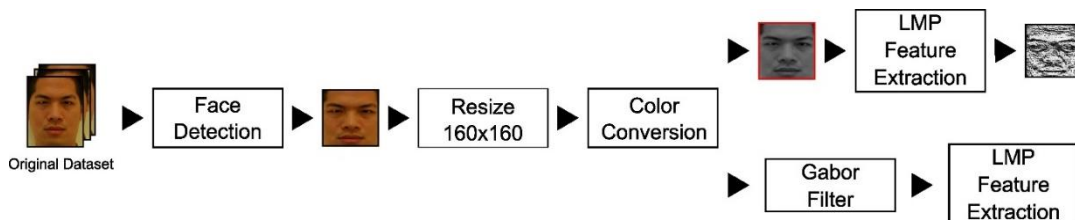
Tabel 3.1. Detail *dataset* sekunder

Ekspresi	<i>Dataset</i> CK+	<i>Dataset</i> JAFFE	<i>Dataset</i> TFEID	Total
Anger	45	30	34	109
Disgust	59	29	40	128
Fear	25	32	40	97
Happy	69	31	40	140
Neutral	107	30	39	176
Sad	28	31	39	98
Surprise	83	30	36	149
Ukuran Citra	640×490	256×256	481×600	-
Warna Citra	RGB & Gray	Gray	RGB	-

3.3 Pembentukan *Dataset*

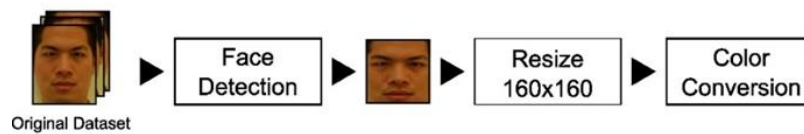
Secara garis besar, dalam pembuatan model AI (khususnya ML dan DL) terdapat proses yang berperan penting, yaitu *preprocessing dataset* seperti ekstraksi fitur, penyesuaian ukuran citra, dan augmentasi (Deshmukh et al., 2016; Franchi et al., 2020; Mohammad & Ali, 2011; Ravi et al., 2020; Sawardekar & Naik, 2018; Shan et al., 2009). Pada penelitian (Robert, 2023), dilakukan sebuah skenario pembentukan *dataset* menggunakan beberapa metode pengolahan citra (seperti konversi warna ke *grayscale*, deteksi wajah, dan ekstraksi fitur), di mana *preprocessing* mempengaruhi performa dari model ML dan DL. Selain itu, pada penelitian (Alam & Yao, 2019) juga dilakukan penelitian yang serupa, di mana *preprocessing* mempengaruhi performa model *machine learning*.

Pada tahap ketiga, dilakukan pembentukan *dataset*. Gambar 3.5 menunjukkan alur pembentukan *dataset* untuk SVM. Pertama, dilakukan pendeteksian wajah menggunakan VJA, proses ini berguna untuk mengurangi *noise* pada citra. Hasil VJA membuat ukuran citra bervariasi, oleh karena itu dilakukan *resizing* citra untuk menyamakan semua ukuran citra dan juga menyesuaikan dengan dimensi *input* model. Selanjutnya dilakukan konversi warna citra dari RGB ke *Grayscale* dikarenakan fitur warna tidak dibutuhkan dan agar dapat diekstraksi fiturnya menggunakan LMP. Terakhir, terdapat dua proses ekstraksi fitur berbeda. Proses ekstraksi fitur pertama menggunakan LMP (Robert, 2023). Proses ekstraksi fitur kedua adalah usulan dari penelitian ini, di mana pertama diaplikasikan Gabor *Filter* terlebih dahulu kemudian diekstraksi menggunakan LMP.



Gambar 3.5. Pembentukan *dataset* untuk SVM

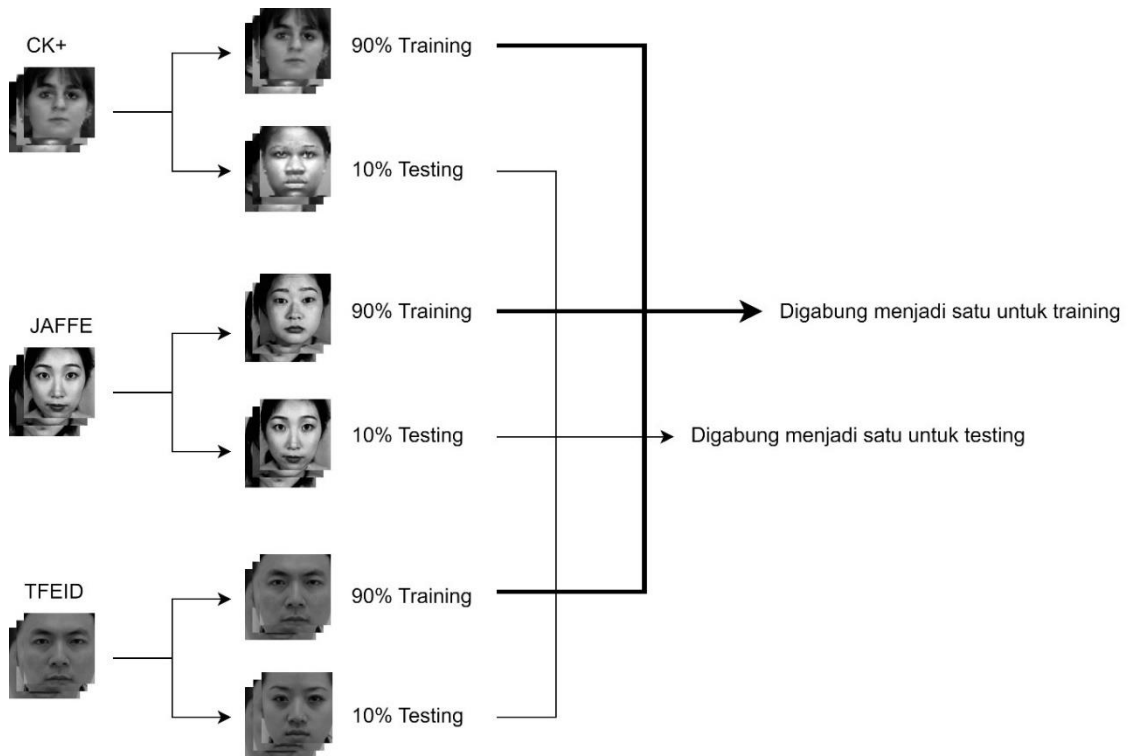
Gambar 3.6 menunjukkan alur pembentukan *dataset* untuk CNN dan MNN. Terdapat 3 proses yang akan dilakukan. Pertama, dilakukan deteksi wajah menggunakan VJA guna mengurangi *noise*. Kemudian mengubah ukuran citra untuk menyamakan semua ukuran citra dan sesuai dengan dimensi *input* model. Terakhir, dilakukan konversi warna dari RGB ke *Grayscale* karena fitur warna tidak dibutuhkan untuk mengenali ekspresi wajah. Skema pembentukan *dataset* ini berdasarkan performa terbaik dari penelitian (Robert, 2023).



Gambar 3.6. Pembentukan *dataset* untuk CNN dan MNN

Dataset yang sudah melalui pembentukan *dataset*, dilakukan augmentasi dari sisi geometris seperti membalikan *flipping* secara horizontal dan vertikal, dan rotasi dari 0° hingga 45° . Tujuan dari augmentasi *dataset* adalah memperbanyak *dataset* dan juga variasi *dataset*.

Dataset yang telah dibentuk kemudian dibagi menjadi dua jenis *dataset*. Jenis pertama adalah *training dataset* dengan jumlah 90% dari total semua *dataset*. Kedua adalah *testing dataset* yang terdiri dari 10% dari total semua *dataset*. Gambar 3.7 menunjukkan visualisasi pembagian *dataset*. SVM *training dataset* digunakan untuk melatih model, sedangkan *testing dataset* digunakan untuk menguji *dataset*. CNN dan MNN terdapat pembagian lagi pada *training*, di mana 90% dari *training dataset* digunakan untuk melatih model dan 10% dari *training dataset* digunakan untuk validasi, dan *testing dataset* digunakan untuk menguji model. Proses pembentukan *dataset* untuk SVM, CNN, dan MNN menggunakan Algoritma 3.1 hingga Algoritma 3.6.



Gambar 3.7. Visualisasi pembagian *dataset*

3.3.1 Deteksi wajah

Proses deteksi wajah menggunakan VJA, VJA memanfaatkan dua komponen yaitu *integral image* dan *Haar Basis Function*. Algoritma 3.1 menunjukkan cara menghitung dari *integral image*. *Input* merupakan citra dengan ruang warna *grayscale* (dengan nama variabel *img*) dengan ukuran $w \times h$ dan *output* berupa *integral image* (yang disimpan pada nama variabel *itg_img*). Algoritma *integral image* cukup sederhana, baris 1 dan 2 dilakukan perulangan terhadap baris dan kolom citra yang digunakan untuk menentukan koordinat *integral image* yang sedang dihitung. Baris ketiga dilakukan perhitungan *integral image* untuk koordinat (i, j) yang menghitung total nilai piksel citra asli mulai dari koordinat $(0,0)$ hingga (i, j) menggunakan Persamaan (2.18).

Algoritma 3.1. Integral image

```
Input      : citra grayscale [w,h] → img
Output     : citra integral [w,h] → itg_img
1 for i = 0 to w do
2     for j = 0 to h do
3         itg_img[i,j] = sum(img_gry[0:i,0:j])
4     end for
5 end for
```

Algoritma 3.2 menunjukkan cara kerja dari VJA. *Input* dari Algoritma 3.2 adalah citra integral dengan ukuran $w \times h$ yang diproses menggunakan Algoritma 3.1. Terdapat beberapa parameter yang digunakan pada Algoritma 3.2. Parameter pertama adalah *detection window* dengan ukuran $w2 \times h2$ yang digunakan untuk perhitungan *Haar*. Parameter kedua adalah *Haar* yang menggunakan Gambar 2.14. Parameter ketiga adalah nilai *threshold* untuk masing-masing *Haar* yang digunakan untuk menentukan apakah *Haar* tersebut merupakan fitur wajah atau bukan. *Output* dari algoritma ini adalah berupa koordinat wajah yang dimulai pada koordinat $[x1, y1]$ dengan panjang dan lebar yaitu $[w2, h2]$.

Baris pertama dan kedua dilakukan perulangan terhadap baris dan kolom citra yang digunakan untuk menentukan koordinat *detction window*, agar lebih mudah memahami dapat melihat Gambar 3.8. Setiap perulangan pada baris pertama dan kedua, dilakukan komputasi tiap area *Haar* (mulai dari jenis *Haar* (a) hingga (d)). Pada baris 4,9,13,17 terdapat tiga perhitungan. Perhitungan pertama adalah area putih, perhitungan kedua adalah area hitam, perhitungan ketiga adalah fitur (area putih – area hitam), untuk lebih mudah memahami perhitungan area putih, area hitam, dan fitur dapat melihat Gambar 3.9. Setiap perhitungan nilai fitur (a) hingga fitur (d), dilakukan pengecekan apakah nilai fitur yang dihitung merupakan fitur atau bukan dengan cara membandingkan nilai fitur dengan *threshold* masing-masing (TH_a hingga TH_d) seperti baris 6,10,14,18. Jika salah satu dari keempat nilai fitur lebih kecil dari *threshold* yang ditentukan maka *Haar* tersebut bukan merupakan fitur wajah dan algoritma akan melakukan pergeseran *detection window* ke koordinat selanjutnya. Selain itu, jika semua nilai fitur lebih besar dari *threshold* maka *detection window* tersebut merupakan wajah, dan algoritma akan memberikan

empat nilai yaitu i, j, dw, dh . i, j merupakan koordinat dari deteksi terakhir. dw, dh merupakan luas dari *detection window* itu sendiri.

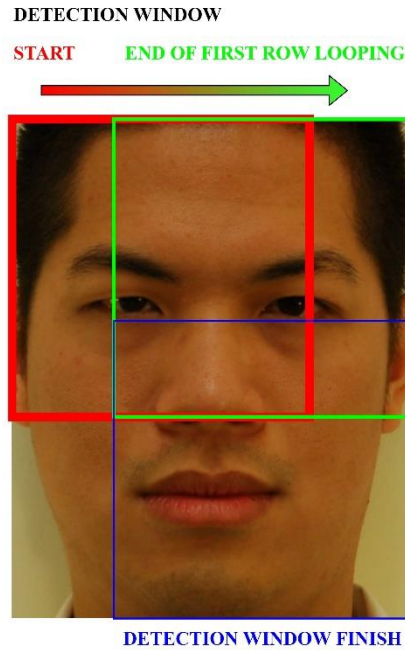
Algoritma 3.2. Algoritma Viola-Jones

```

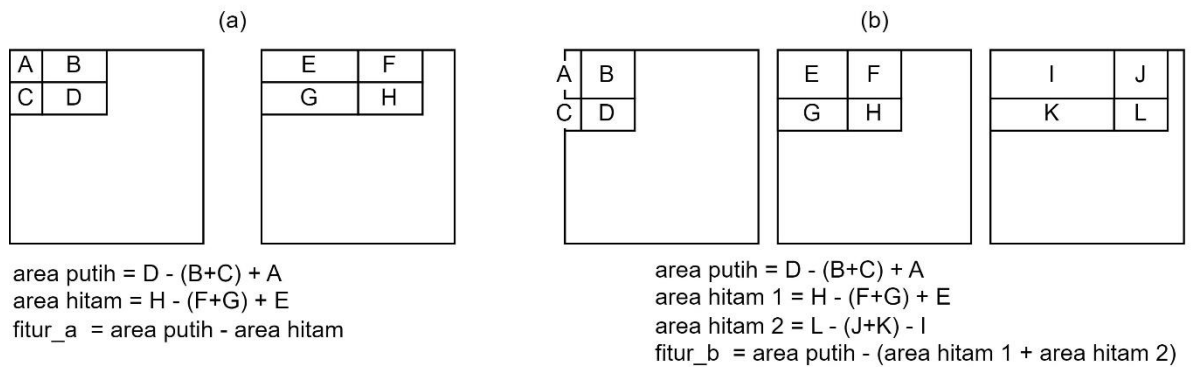
Input      : citra integral  $w \times h \rightarrow itg\_gry$ 
Parameter  : detection window dengan ukuran  $w2 \times h2 \rightarrow dw$ 
            : Haar pada Gambar 2.14 (a) hingga (d)
            : thershold  $\rightarrow TH(a)$  hingga  $TH(e)$ 
Output     : koordinat wajah  $\rightarrow [x1, y1], [w2, h2]$ 

for i = 0 to (w-w2) do
    for j = 0 to (h-h2) do
        # hitung fitur Haar (a) pada Gambar 2.14
        hitung fitur_a = area putih Gambar 2.14 (a) -
1   area hitam Gambar 2.14 (a) untuk detection window
2   (i:i+w2, j:j+h2)
3   if fitur_a < TH_a do
4       continue
5
6       # hitung fitur Haar (b) pada Gambar 2.14
7       hitung fitur_b = area putih Gambar 2.14 (b) -
8       area hitam Gambar 2.14 (b) untuk detection window
9       (i:i+w2, j:j+h2)
10      if fitur_b < TH_b do
11          continue
12
13      # hitung fitur Haar (c) pada Gambar 2.14
14      hitung fitur_c = area putih Gambar 2.14 (c) -
15      area hitam Gambar 2.14 (c) untuk detection window
16      (i:i+w2, j:j+h2)
17      if fitur_c < TH_c do
18          continue
19
20      # hitung fitur Haar (d) pada Gambar 2.14
21      hitung fitur_d = area putih Gambar 2.14 (d) -
22      area hitam Gambar 2.14 (d) untuk detection window
23      (i:i+w2, j:j+h2)
24      if fitur_d < TH_d do
25          continue
26      return i, j, (dw), (dh)

```

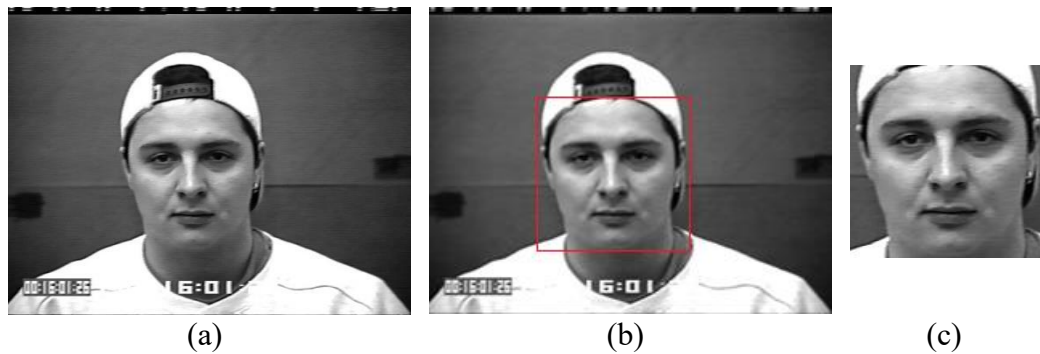
Gambar 3.8. Pergeseran Detection Window



Gambar 3.9. Contoh perhitungan *Haar*

Gambar 3.9 menunjukkan contoh perhitungan untuk fitur a dan b pada Algoritma 3.2 baris 4 dan 9. Pertama, dilakukan perhitungan pada area hitam dan putih. Untuk mendapatkan hanya luas D dilakukan pengurangan dengan luas B dan C kemudian dilakukan penambahan luas A . Perhitungan dilakukan secara demikian dikarenakan pada *integral image* luas $D = A + B + C + D$, luas $B = A + B$, dan luas $C = A + C$. Secara teknis, pengurangan dengan luas A dilakukan dua kali

dikarenakan luas B dan C , oleh karena itu dilakukan penambahan luas A setelah pengurangan dengan B dan C . Perhitungan luas H memiliki pola yang sama dengan perhitungan luas D . Kemudian dilakukan pengurangan antara area putih dengan hitam untuk mendapatkan nilai fitur. Fitur kemudian dibandingkan dengan nilai *threshold* untuk menentukan apakah Haartersebut fitur wajah atau bukan. Pola untuk perhitungan area putih, area hitam, dan fitur untuk *Haar* (a), (b), (c), (d) memiliki pola yang sama. Gambar 3.10 menunjukkan hasil dari implementasi algoritma deteksi wajah pada sebuah citra.



Gambar 3.10. (a) citra original, (b) hasil deteksi, (c) hasil *cropping*

3.3.2 Image Resize

Proses perubahan ukuran citra menggunakan metode *bicubic interpolation*. Algoritma 3.3 menunjukkan cara kerja dari *bicubic interpolation*. *Input* berupa citra (yang ingin diubah ukurannya) dan rasio (skala ukuran yang diinginkan). Parameter berupa koefisien a yang dapat mempengaruhi koordinat tetangga (umumnya nilai dikisaran -0.75 hingga -0.5). Hasil dari algoritma ini adalah citra dengan ukuran $[(H \times r), (W \times r), C]$.

Pada baris 1 dilakukan perhitungan dH , dW yang digunakan untuk menentukan ukuran citra setelah diubah ukurannya dan kemudian dilakukan perhitungan h yang digunakan untuk konstanta yang akan mempengaruhi koordinat tetangga pada piksel yang akan dihitung. Baris 2 dilakukan pembuatan matriks yang digunakan untuk menyimpan hasil. Baris 4,5,6 dilakukan perulangan pada *channel*, dH , dW (matriks yang dibuat pada baris 2), perulangan ini digunakan untuk menentukan koordinat matriks hasil yang akan dihitung. Baris 6 dan 7 dilakukan perhitungan x dan y , yang merupakan konstanta yang mempengaruhi koordinat tetangga (nilai berubah setiap perulangan berjalan pada baris 3, 4, 5). Baris 8 hingga 15 dilakukan perhitungan $x1$ hingga $x4$ dan $y1$ hingga $y4$, variabel tersebut digunakan untuk menentukan bobot (*weight*) tiap tetangga dan koordinat tetangga pada piksel yang sedang dihitung. Baris 16 dan 17 dilakukan perhitungan menggunakan persamaan (2.19), di mana baris 16 perhitungan untuk bobot horizontal, baris 17 untuk bobot vertikal yang masing-masing disimpan pada *matrix_l* dan *matrix_r*. Baris 19 dilakukan pembuatan matriks berukuran 4×4 yang digunakan untuk menyimpan nilai piksel tetangga. Baris 20 hingga 35 merupakan pengambilan nilai tetangga berdasarkan dari perhitungan sebelumnya. Baris 36 merupakan perkalian antara matriks tetangga 4×4 dengan bobot horizontal, kemudian dikalikan dengan bobot vertikal. Hasil yang didapatkan kemudian disimpan pada variabel *output*.

Algoritma 3.3. *Image Resize* (menggunakan *Bicubic Interpolation*)

```

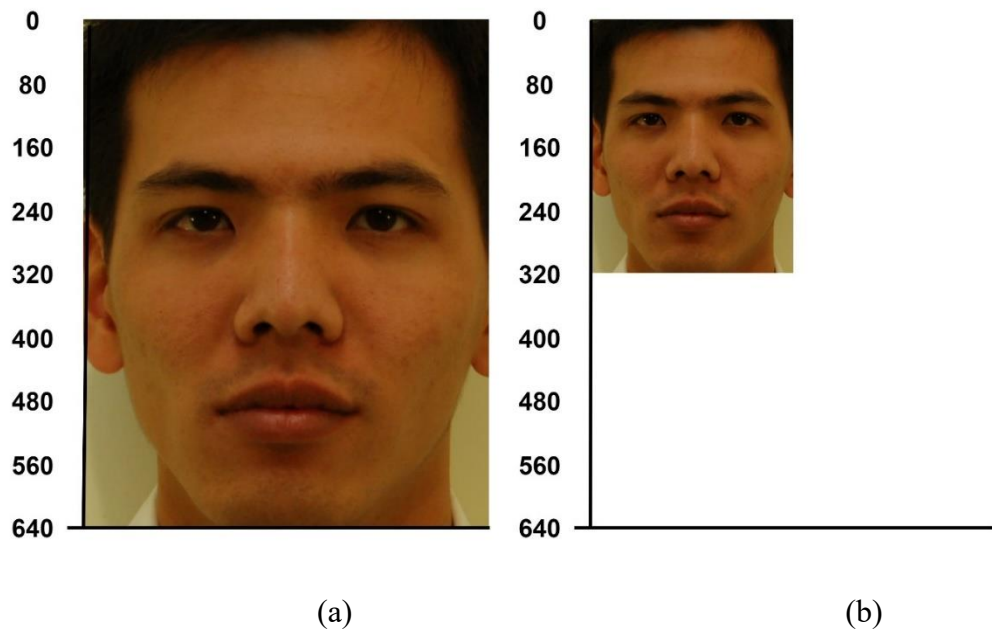
Input      : Citra ukuran  $[H,W,C] \rightarrow \text{img}$ 
              : Ratio antara 0 hingga  $\infty \rightarrow r$ 
Parameter : koefisien (a) = -0.5  $\rightarrow a$ 
Output    : Citra ukuran  $[(H*r),(W*r),C] \rightarrow \text{output}$ 

1 Calculate dH = H*r, dW = W*r, h = 1/ratio #used for
  calculation
2 Create Matrix output Dimension [dH,dW,C] #to store
  result
3 For i = 0 to C do
4     For j = 0 to dH do
5         For k = 0 to dW do
6             Calculate x = i * h + 2
7             Calculate y = j * h + 2
8             Calculate x1 = 1 + x - floor(x)
9             Calculate x2 = x - floot(x)
10            Calculate x3 = floor(x) + 1 - x
11            Calculate x4 = floot(x) + 2 - x
12            Calculate y1 = 1 + y - floor(y)
13            Calculate y2 = y - floot(y)
14            Calculate y3 = floor(y) + 1 - y
15            Calculate y4 = floot(y) + 2 - y
16            Create matrix mat_l dimension [1,4],
mat_r dimension [4,1]
              # h() menggunakan pers (2.19)
17            Calculate mat_l[0,0] = h(x1),
mat_l[0,1] = h(x2), mat_l[0,2] = h(x3), mat_l[0,3] =
h(x4) # bobot horizontal
18            Calculate mat_r[0,0] = h(y1),
mat_l[0,1] = h(y2), mat_l[0,2] = h(y3), mat_l[0,3] =
h(y4) # bobot vertical
19            Create matrix mat_m dimension [4,4]
20            mat_l[0,0] = img[y-y1, x-x1]
21            mat_l[1,0] = img[y-y2, x-x1]
22            mat_l[2,0] = img[y+y3, x-x1]
23            mat_l[3,0] = img[y+y4, x-x1]
24            mat_l[0,1] = img[y-y1, x-x2]
25            mat_l[1,1] = img[y-y2, x-x2]
26            mat_l[2,1] = img[y+y3, x-x2]
27            mat_l[3,1] = img[y+y4, x-x2]
28            mat_l[0,2] = img[y-y1, x+x3]
29            mat_l[1,2] = img[y-y2, x+x3]
30            mat_l[2,2] = img[y+y3, x+x3]
31            mat_l[3,2] = img[y+y4, x+x3]
32            mat_l[0,3] = img[y-y1, x+x4]
33            mat_l[1,3] = img[y-y2, x+x4]
34            mat_l[2,3] = img[y+y3, x+x4]
35            mat_l[3,3] = img[y+y4, x+x4]

```

36	Calculate $output = (mat_m \cdot mat_l) \cdot$
mat_r	

Gambar 3.11 menunjukkan hasil perubahan ukuran citra wajah menggunakan Algoritma 3.3. Berdasarkan Gambar 3.11 ukuran dari *input* citra adalah 600×480 dan ratio = 0,5. Ukuran menjadi 300×240 setelah melalui proses algoritma perubahan ukuran citra.



Gambar 3.11, (a) citra original, (b) citra setelah diubah ukurannya

3.3.3 Color conversion

Proses konversi warna dari RGB ke *Grayscale* menggunakan Persamaan (2.2). Algoritma 3.4 menunjukan proses konversi ruang warna citra dari RGB ke *Grayscale* dengan cara mengambil nilai Y dari YC_bC_r . *Input* dari algoritma ini adalah citra RGB yang memiliki dimensi $W \times H \times C$ yang diberi nama variabel *img_rgb*. *Output* dari algoritma ini adalah citra *grayscale* yang disimpan pada variable *img_gry*. Baris 1 dan 2 dilakukan perulangan baris dan kolom citra yang digunakan untuk menentukan koordinat citra *grayscale* yang akan dihitung. Baris 3 dilakukan konversi citra *grayscale* menggunakan Persamaan (2.2), di mana dilakukan penjumlahan nilai dari ketiga *channel* citra RGB. Tiap *channel* memiliki bobot masing-masing, untuk *channel* R memiliki bobot 0.299, *channel* G memiliki bobot 0.587, dan *channel* B memiliki bobot 0.114. Gambar 3.12 menunjukkan contoh hasil implementasi algoritma konversi warna dari RGB ke *Grayscale* (dengan menggunakan nilai Y dari YC_bC_r).

Algoritma 3.4. RGB to *Grayscale*

```
Input      : citra RGB [w,h,c] → img_rgb
Output     : citra Grayscale [w,h] → img_gry
1 for i = 0 to w do
2     for j = 0 to h do
3         img_gry[i,j] = 0.299*img_rgb[i,j,0] +
          0.587*img_rgb[i,j,1] + 0.114*img_rgb[i,j,2]
4     end for
5 end for
```



(a)

(b)

Gambar 3.12. Sebelum (a) dan sesudah (b) konversi warna

3.3.4 Gabor Filter

Pada proses SVM terdapat proses Gabor *filter* sebelum diekstraksi menggunakan LMP. Pertama dilakukan pembuatan filter terlebih dahulu. Algoritma 3.5 menunjukkan cara pembuatan Gabor filter. *Input* dari algoritma ini adalah sebuah citra *grayscale* yang memiliki ukuran $[H, W]$. Kemudian jumlah filter, luas kernel, lambda (λ), psi (φ), sigma σ , dan gamma γ yang akan digunakan untuk pembuatan Gabor *filter*. Terakhir adalah *threshold* bawah dan *threshold* atas yang digunakan untuk deteksi tepi. Pada baris 1 dilakukan perulangan untuk menentukan rotasi (nilai theta) Gabor *filter* berdasarkan jumlah filter yang digunakan. Baris 2 dilakukan pembuatan Gabor *filter* berdasarkan Persamaan (2.23) dan parameter yang di-*input*. Gambar 3.13 menunjukkan contoh Gabor *filter* dengan total 16 filter. Pada baris 4 dilakukan perulangan terhadap filter-filter, di mana filter-filter tersebut digunakan untuk konvolusi citra pada baris 5. Pada baris 7 dilakukan deteksi tepi menggunakan Canny *edge detection* dengan parameter *lower threshold* dan *upper threshold*.

Algoritma 3.5. Gabor Filter

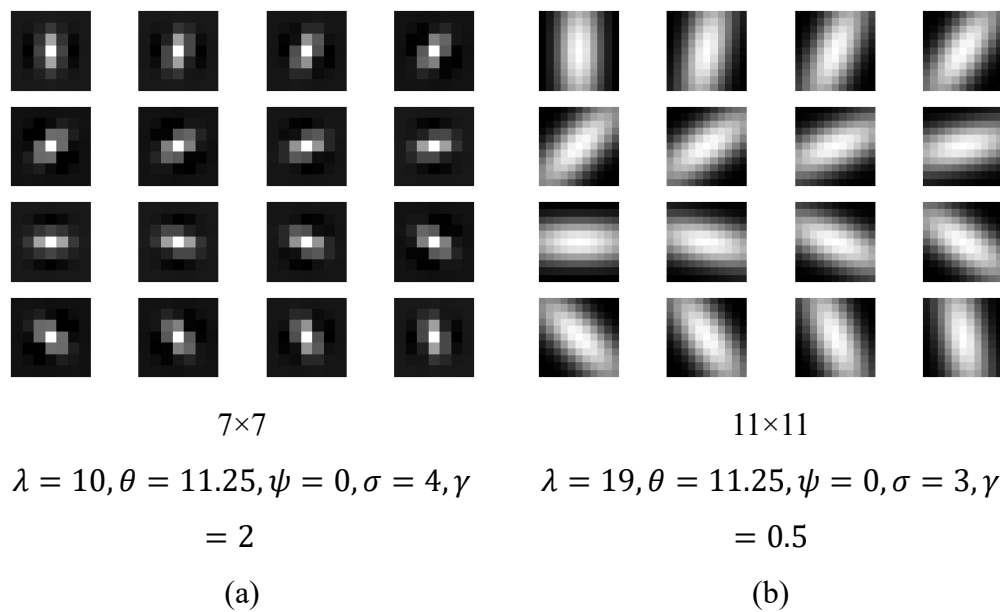
Input	: citra <i>grayscale</i> ukuran $[H, W] \rightarrow \text{img}$: jumlah filter dari 1 hingga $\infty \rightarrow$ num_filter : Kernel size $N \times N \rightarrow \text{ksize}$: Lambda $\lambda \rightarrow \text{lambd}$: Psi $\varphi \rightarrow \text{psi}$: Sigma $\sigma \rightarrow \text{sigma}$: Gamma $\gamma \rightarrow \text{gamma}$: threshold bawah $\rightarrow \text{min_int}$: threshold atas $\rightarrow \text{max_int}$
Output	: Citra ukuran $[H, W] \rightarrow \text{output}$

```

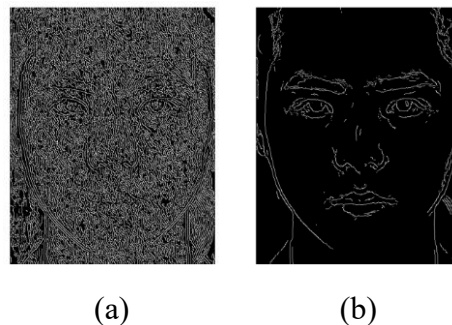
1 for i = 0 to 180 with increment 180/num_filter do
2     Create Gabor filter filter with kernel size =
      ksize, lambda = lambd, Theta = i, Psi = psi, Sigma =
      sigma, Gamma = gamma
3     # proses konvolusi
4     For i = 0 to num_filter do
5         Output = img  $\otimes$  filter[i]
6     # proses deteksi tepi
7     Apply canny edge detection on output with lower
      threshold = min_int, upper threshold = max_int

```

Perubahan nilai parameter dilakukan pada λ , γ , dan σ pada Gambar 3.14 menunjukkan hasil dari pengaplikasian Gabor filter, di mana (a) menunjukkan hasil awal (sebelum dilakukan *parameter tuning*) dan (b) hasil akhir (setelah *parameter tuning*). Berdasarkan Gambar 3.13 Sebelum dilakukan perubahan nilai parameter citra memiliki banyak *noise*, setelah dilakukan perubahan nilai parameter, fitur wajah seperti alis, mata, hidung, dan mulut terlihat.



Gambar 3.13. Gabor *Filter* (a) sebelum dan (b) sesudah *tuning* parameter



Gambar 3.14. Hasil algoritma Gabor *filter*

3.3.5 Ekstraksi fitur LMP

Proses ekstraksi fitur menggunakan metode LMP. Algoritma 3.6 menunjukkan proses dari ekstraksi fitur LMP. *Input* dari algoritma LMP adalah citra *grayscale* dengan ukuran $w \times h$. Parameter yang digunakan adalah *pattern* yang memiliki dimensi 8×2 yang digunakan untuk menentukan arah perhitungan tetangga. Hasil dari Algoritma 3.6 adalah citra LMP dengan ukuran $(w - 4) \times (h - 4)$, ukuran citra berkurang untuk menghindari perhitungan di luar dari ukuran citra. Baris 2 dan 3 dilakukan perulangan baris dan kolom citra yang digunakan untuk menentukan koordinat citra LMP yang akan dihitung. Baris 4 menyimpan nilai piksel berdasarkan koordinat baris 2 dan 3. Baris 5 dilakukan perulangan untuk mengambil arah dari *ptn* secara berurutan. Baris 6 dan 7 digunakan untuk megambil nilai piksel tetangga radius pertama (disimpan pada variable *cur_val1*) dan nilai piksel tetangga radius kedua (disimpan pada variable *cur_val2*). Baris 9 dilakukan perbandingan nilai tengah (*ctr_val*) dengan nilai tetangga radius pertama (*cur_val1*) berdasarkan (2.22). Baris 14 dilakukan perbandingan nilai tetangga radius pertama dengan nilai tetangga radius kedua (*cur_val2*) berdasarkan persamaan (2.21). Kemudian dilakukan perhitungan nilai LMP menggunakan persamaan (2.20). Gambar 3.15 menunjukkan contoh hasil implementasi algoritma LMP pada citra wajah.

Algoritma 3.6. *Local Monotonic Pattern*

```

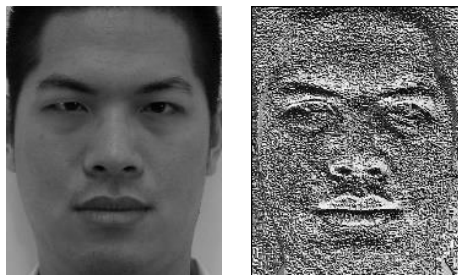
Input      : citra grayscale [w,h] → img_gry
Parameter  : pattern → ptn = [[0,1],[1,1],[1,0],[1,-1],
                                [0,-1],[-1,-1],[-1,0],[-1,1]]
Output     : citra LMP [w,h] → img_lmp
1  for i = 2 to w-2 do
2      for j = 2 to h-2 do
3          ctr_val = img_gry[i,j]
4          for p = 0 to 7 do
5              cur_val1 =
img_gry[x+ptn[p,0]], [y+ptn[p,1]]
6              cur_val2 = img_gry[x+(ptn[p,0]*2)],
[y+(ptn[p,1]*2)]
7                  # pers 2.20
8                  if (cur_val1 - ctr_val) > 0 do
9                      s1 = 1
10                 else
11                     s1 = 0

```

```

12         # pers 2.19
13         if (cur_val2 - cur_val1) > 0 do
14             s2 = 1
15         else
16             s2 = 0
17         # pers 2.18
18         img_lmp[i,j] = img_lmp + (s1 * s2 *
2p)
19     end for
20 end for
21 end for
22 return img_lmp

```



(a)

(b)

Gambar 3.15. Sebelum (a) dan sesudah (b) ekstrasi fitur LMP

3.4 Pembentukan Model

3.4.1 Pembentukan Model SVM

Dalam penelitian sebelumnya (Robert, 2023), telah dilakukan pengenalan ekspresi wajah menggunakan SVM dengan menggunakan 4 kernel yaitu: Linear, Polynomial, Sigmoid, dan RBF. Pada penelitian tersebut *dataset* TFEID yang digunakan untuk melatih dan menguji model. *Dataset* TFEID memiliki keterbatasan dalam jumlah dan variasi etnis, di mana hanya terdapat 1 etnis saja yaitu orang Taiwan. Berdasarkan dari penelitian (Robert, 2023), didapatkan model terbaik untuk mengenal ekspresi wajah adalah menggunakan kernel Sigmoid. Pada penelitian ini akan dilakukan pengujian ulang SVM dengan *dataset* gabungan berdasarkan usulan. Terdapat 4 kernel yang akan digunakan untuk model SVM dalam penelitian ini yaitu: *Linear*, *Polynomial*, *Sigmoid*, dan *RBF*. Persamaan (2.29), (2.30), (2.31), (2.32) menunjukkan persamaan dari keempat kernel yang digunakan secara berurutan. Selain kernel terdapat parameter yang juga akan dikonfigurasi yaitu *C*. Terdapat beberapa nilai *C* berada diantara 0 hingga 100.

3.4.2 Pembentukan Model CNN

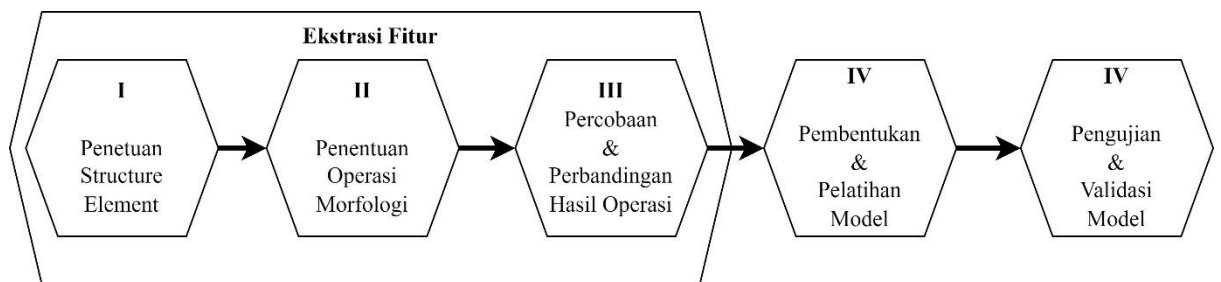
Dalam penelitian sebelumnya (Robert, 2023), model CNN yang digunakan adalah MobileNetV2, yang merupakan *pre-trained model*. Penelitian tersebut menggunakan *dataset* TFEID yang memiliki keterbatasan baik dalam jumlah maupun variasi dari etnis ekspresi wajah manusia. Oleh karena itu, dalam penelitian ini akan dilakukan pelatihan dan pengujian ulang menggunakan *dataset* gabungan sesuai dengan usulan yang telah diuraikan sebelumnya. Tabel 3.2 menunjukkan arsitektur MobileNetV2 dari konvolusi *layer* pertama hingga *output layer*. Kolom pertama menunjukkan tipe dari *layer* dan juga *stride* yang digunakan. Kolom kedua menunjukkan ukuran filter dan jumlah filter. Kolom terakhir menunjukkan ukuran masukan citra dari *layer* sebelumnya. Fungsi aktivasi (*activation function*) pada *output layer* adalah *Softmax*, selain itu model juga dilakukan *parameter tuning* pada *Learning rate*, *batch size*, dan fungsi aktivasi pada *FC-Layer* saat dilakukan proses pelatihan.

Tabel 3.2. Arsitektur MobileNetV2

Type/Stride	Filter Shape	<i>Input</i> Size
Conv / 2	3×3×3×32	160×160×3
Conv dw / 1	3×3×32 dw	112×112×32
Conv / 1	1×1×32×64	112×112×32
Conv dw / 2	3×3×64 dw	112×112×64
Conv / 1	1×1×64×128	56×56×64
Conv dw / 1	3×3×128 dw	56×56×128
Conv / 1	1×1×128×128	56×56×128
Conv dw / 2	3×3×128 dw	56×56×128
Conv / 1	1×1×128×256	28×28×128
Conv dw / 1	3×3×256 dw	28×28×256
Conv / 1	1×1×256×256	28×28×256
Conv dw / 2	3×3×256 dw	28×28×256
5×	Conv dw / 1	3×3×512 dw
	Conv / 1	1×1×512×512
	Conv dw / 2	3×3×512 dw
	Conv / 1	1×1×512×1024
	Conv dw / 2	3×3×1024 dw
Conv / 1	1×1×1024×1024	7×7×1024
Avg Pool / 1	Pool 7×7	7×7×1024
FC / 1	1024×7	1×1×1024
Softmax / 1	Classifier	1×1×7

3.4.3 Pembentukan Model MNN

Gambar 3.16 menunjukkan alur dari pembentukan MNN. Pertama dilakukan penentuan SE yang digunakan. Dalam penelitian ini akan digunakan bentuk SE berdasarkan penelitian lain (Shen et al., 2019), dan SE yang diusulkan pada penelitian ini. Selanjutnya dilakukan penentuan operasi morfologi, pertama akan digunakan operasi morfologi yang telah dilakukan peneliti lain dan juga operasi morfologi usulan pada penelitian ini. Kemudian dilakukan uji coba, analisis, dan dibandingkan pada semua operasi morfologi dan SE yang diusulkan pada penelitian ini. Terakhir dilakukan perancangan arsitektur MNN berdasarkan dari analisis dari tahap ketiga.

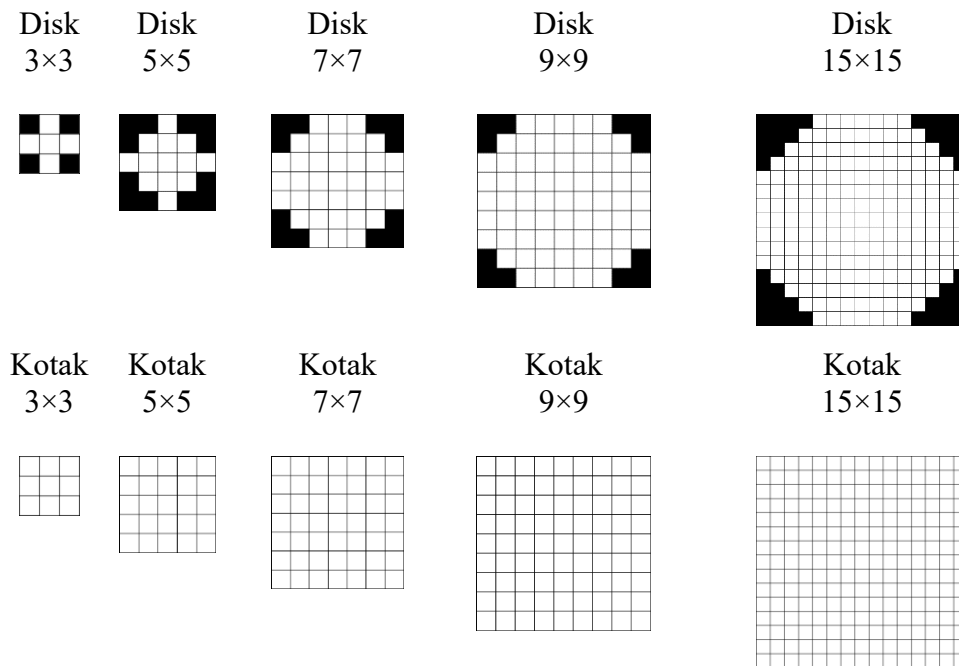


Gambar 3.16. Proses pembentukan model

3.4.3.1 Ekstraksi Fitur

3.4.3.1.1 Penentuan *Structure Element*

Dalam penelitian ini digunakan dua SE dan tiga operasi morfologi yang dibandingkan satu dengan yang lain. Gambar 3.17 menunjukkan SE bentuk beserta ukuran yang digunakan untuk operasi morfologi. Terdapat dua bentuk SE yang digunakan, yaitu *disk* dan kotak. Terdapat empat ukuran SE yang digunakan, yaitu 3×3 , 5×5 , 9×9 , dan 15×15 , ukuran digunakan untuk kedua bentuk. Di mana kotak berwarna putih bernilai 1 dan hitam adalah 0.



Gambar 3.17. *Struktur element* yang digunakan

3.4.3.1.2 Penentuan Operasi Morfologi

Tahap setelah penentuan bentuk dan juga ukuran SE yang akan digunakan adalah penentuan operasi morfologi yang digunakan. Terdapat tiga operasi yang digunakan dalam penelitian ini. Operasi pertama adalah operasi morfologi *opening* pada citra original, kemudian dilakukan pengurangan nilai piksel antara citra original dengan hasil *opening*, operasi *opening* itu sendiri adalah operasi erosi yang dilanjutkan operasi dilasi. Operasi kedua adalah erosi pada citra original, kemudian dilakukan pengurangan citra original terhadap hasil erosi. Ketiga adalah dilasi pada citra original, kemudian dilakukan pengurangan hasil dilasi terhadap citra original. Ketiga operasi tersebut menggunakan Algoritma 3.7, Algoritma 3.8, dan Algoritma 3.9 secara berurutan.

Algoritma 3.7. Gradien Morfologi Opening

```
input          : image grayscale -> image [W×H]
                  : Structure Element -> SE [N×N]
output         : feature image -> output [(W-2)×(H-2)]

1 # start
2 for i = 0 to W do
3   for j = 0 to H do
4     create empty list arr
5     for h = 0 to z do
6       for w = 0 to z do
7         if SE[h,w] == 1 do
8           arr append(image[i+(h-space),j+(h-space)])
9         end if
10      end for
11    end for
12    output[i-space,j-space] = min(arr)
13  end for
14 end for
15 for i = 0 to W do
16   for j = 0 to H do
17     create empty list arr
18     for h = 0 to z do
19       for w = 0 to z do
20         if SE[h,w] == 1 do
21           arr append(image[i+(h-space),j+(h-space)])
22         end if
23       end for
24     end for
25     output[i-space,j-space] = max(arr)
26   end for
27 end for
28 for i to w do
29   for j to h do
30     output[i,j] = image[i,j] - output[i,j]
31   end for
32 end for
```

Pada Algoritma 3.7. terdapat dua *input* yang digunakan, pertama adalah citra original dengan ukuran $W \times H$. Kedua adalah *structure element* dengan ukuran $N \times N$. Hasil dari algoritma ini adalah sebuah fitur dalam bentuk citra dengan ukuran $(W \times H)$. Pada baris pertama dilakukan perhitungan *space* yang digunakan untuk menentukan koordinat mulai operasi *opening*. Baris 2 dan 3 dilakukan perulangan baris dan kolom citra yang digunakan untuk menentukan koordinat operasi *opening*. Baris 4 digunakan untuk menyimpan kandidat nilai

piksel pada baris 5 hingga 11. Baris 12 dilakukan pengambilan nilai terkecil yang digunakan untuk sebagai hasil fitur citra berdasarkan persamaan. Kemudian baris 15 dan 16 dilakukan perulangan baris dan kolom citra untuk menentukan koordinat operasi *opening*. Baris 17 digunakan untuk menyimpan kandidat nilai piksel pada baris 18 hingga 24. Baris 25 dilakukan pengambilan nilai terbesar yang digunakan untuk sebagai hasil fitur citra berdasarkan Persamaan (2.14). Kemudian pada baris 28 dan 29 dilakukan perulangan kembali pada baris dan kolom citra untuk menentukan koordinat perhitungan. Baris 30 dilakukan pengurangan antara citra original dengan citra *opening*.

Algoritma 3.8. Gradien Morfologi Erosi

```

input          : image grayscale -> image [W × H]
                  : Structure Element -> SE [N × N]
output         : feature image -> output [(W-2) × (H-2)]
1  for i = 0 to W do
2    for j = 0 to H do
3      create empty list arr
4      for h = 0 to z do
5        for w = 0 to z do
6          if SE[h,w] == 1 do
7            arr append(image[i+(h-space),j+(h-space)])
8          end if
9        end for
10     end for
11     output[i-space,j-space] = min(arr)
12   end for
13 end for
14 for i to w do
15   for j to h do
16     output[i,j] =- image[i,j] - output[i,j]
17   end for
18 end for

```


Pada Algoritma 3.8. terdapat dua *input* yang digunakan, pertama adalah citra original dengan ukuran $W \times H$. Kedua adalah *structure element* dengan ukuran $N \times N$. Hasil dari algoritma ini adalah sebuah fitur dalam bentuk citra dengan ukuran $(W \times H)$. Pada baris pertama dilakukan perhitungan *space* yang digunakan untuk menentukan koordinat mulai operasi *opening*. Baris 1 dan 2 dilakukan perulangan baris dan kolom citra yang digunakan untuk menentukan koordinat operasi *opening*. Baris 3 digunakan untuk menyimpan kandidat nilai piksel pada baris 4 hingga 10. Baris 11 dilakukan pengambilan nilai terkecil yang digunakan untuk sebagai hasil fitur citra berdasarkan Persamaan (2.10). Kemudian pada baris 14 dan 15 dilakukan perulangan kembali pada baris dan kolom citra untuk menentukan koordinat perhitungan. Baris 16 dilakukan pengurangan antara citra original dengan citra *opening*.

Algoritma 3.9. Gradien Morfologi Dilasi

```

input          : image grayscale -> image [W × H]
                  : Structure Element -> SE [N × N]
output         : feature image -> output [(W-2) × (H-2)]
1  for i = 0 to W do
2    for j = 0 to H do
3      create empty list arr
4      for h = 0 to z do
5        for w = 0 to z do
6          if SE[h,w] == 1 do
7            arr append(image[i+(h-space),j+(h-space)])
8          end if
9        end for
10     end for
11     output[i-space,j-space] = max(arr)
12   end for
13 end for
14 for i to w do
15   for j to h do
16     output[i,j] =- image[i,j] - output[i,j]
17   end for
18 end for


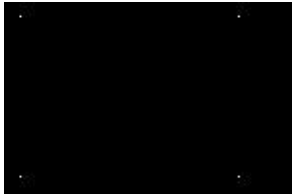




```











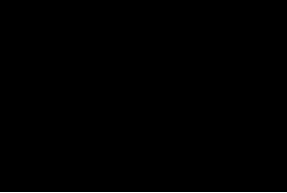
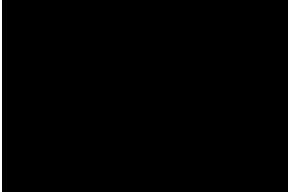
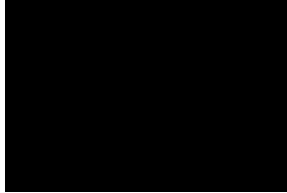
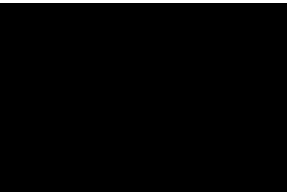
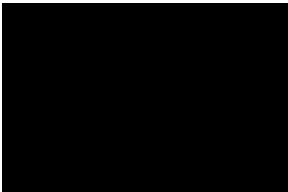
Pada Algoritma 3.9. terdapat dua *input* yang digunakan, pertama adalah citra original dengan ukuran $W \times H$. Kedua adalah *structure element* dengan ukuran $N \times N$. Hasil dari algoritma ini adalah sebuah fitur dalam bentuk citra dengan ukuran $(W \times H)$. Baris 1 dan 2 dilakukan perulangan baris dan kolom citra yang digunakan untuk menentukan koordinat operasi dilasi. Baris 3 digunakan untuk menyimpan kandidat nilai piksel pada baris 4 hingga 10. Baris 11 dilakukan pengambilan nilai terbesar yang digunakan untuk sebagai hasil fitur citra berdasarkan Persamaan (2.8). Kemudian pada baris 14 dan 15 dilakukan perulangan kembali pada baris dan kolom citra untuk menentukan koordinat perhitungan. Baris 16 dilakukan pengurangan antara citra dilasi dengan citra original.

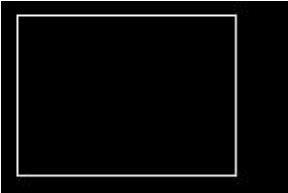




3.4.3.1.3 Percobaan dan Perbandingan Hasil Operasi Morfologi

Tabel 3.2 menunjukkan hasil operasi morfologi menggunakan SE berdasarkan 3.4.3.1 dan 3.4.3.2. Pada operasi morfologi Tabel 3.2. digunakan citra yang memiliki objek persegi panjang berwarna putih. Di mana putih pada citra bernilai 1 sedangkan hitam adalah 0.

Tabel 3.3. Operasi Morfologi dan *Structure Element* pada Persegi Panjang

<p style="text-align: center;"><i>Original Image</i></p> 	
SE & Operasi	Operasi dan Hasil
<i>Disk & Original - Opening</i>	  
	<p style="text-align: center;">3×3 5×5 7×7</p>
	 
	<p style="text-align: center;">9×9 15×15</p>

<p><i>Disk & Original - Erosi</i></p>	<div>    </div> <div> <p>3×3 5×5 7×7</p> </div> <div>   </div> <div> <p>9×9 15×15</p> </div>
<p><i>Disk & Dilasi - Original</i></p>	<div>    </div> <div> <p>3×3 5×5 7×7</p> </div> <div>   </div> <div> <p>9×9 15×15</p> </div>
<p><i>Kotak & Original - Opening</i></p>	<div>    </div> <div> <p>3×3 5×5 7×7</p> </div> <div>   </div> <div> <p>9×9 15×15</p> </div>







Kotak & <i>Original</i> - Erosi			
	3×3	5×5	7×7
Kotak & <i>Dilasi</i> - <i>Original</i>			
	9×9	15×15	







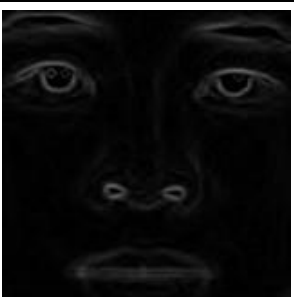
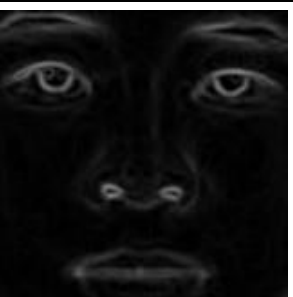


Berdasarkan dari Tabel 3.3 dapat dilihat operasi GMO dengan SE *disk* 3×3 menghasilkan *corner* pada persegi panjang. Operasi GMD dengan SE *disk* 3×3 memiliki hasil garis berbentuk persegi panjang dengan nilai tiap *corner* terdapat hilang. Operasi GME dengan disk 3×3 mirip dengan GMD dengan SE *disk* 3×3 namun *corner* dari GME utuh. Operasi GMO menggunakan SE kotak 3×3 tidak dapat mengekstrasi fitur dari persegi. Operasi GMD dengan SE kotak 3×3 menghasilkan garis yang berbentuk persegi panjang. GME dengan SE kotak 3×3 memiliki hasil yang mirip dengan GMD, namun memiliki luas yang berbeda. Pada GMD garis terlelak pada luar objek sedangkan GME berada pada dalam objek.



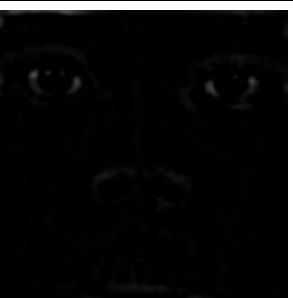







Kemudian dilakukan operasi menggunakan 3 ukuran SE lain yaitu 5×5 , 7×7 , 9×9 , dan 15×15 . Dari hasil yang didapatkan, tiap ukuran memiliki hasil yang mirip dengan operasi yang digunakan. GMO mengekstrasi tiap sudut pada persegi dengan perbedaan pada ukuran sudut, di mana semakin besar SE semakin besar sudut yang didapatkan. GMD dapat mengekstrasi tepi dari persegi, di mana semakin besar ukuran SE semakin tebal garis yang didapatkan. GME dapat mengekstrasi tepi dari persegi, di mana semakin besar ukuran SE semakin tebal garis yang didapatkan. Operasi GMD dan GME memiliki hasil yang mirip dengan perbedaan tepi pada GMD menebal ke arah luar persegi, sedangkan tepi pada GME menebal ke arah dalam.

Tabel 3.4 merupakan hasil morfologi menggunakan citra wajah dengan warna *grayscale*. Nilai intensitas piksel pada citra berkisaran antara 0 hingga 255. Operasi morfologi dan SE yang digunakan berdasarkan usulan pada subbab 3.4.3.1 dan 3.4.3.2.

Tabel 3.4. *Structure Element* dan Operasi Morfologi pada Wajah

Citra Original	
	
SE & Operasi	Operasi dan Hasil
Disk & GMO	
	3×3
	
	5×5
	
	7×7
	
	9×9
	
	15×15

<p><i>Disk & GME</i></p>	<div data-bbox="435 304 729 600"></div> <div data-bbox="549 600 611 633">3×3</div> <div data-bbox="735 304 1029 600"></div> <div data-bbox="852 600 914 633">5×5</div> <div data-bbox="1035 304 1329 600"></div> <div data-bbox="1142 600 1204 633">7×7</div> <div data-bbox="435 633 729 929"></div> <div data-bbox="549 929 611 963">9×9</div> <div data-bbox="735 633 1029 929"></div> <div data-bbox="839 929 927 963">15×15</div>
<p><i>Disk & GMD</i></p>	<div data-bbox="435 972 729 1267"></div> <div data-bbox="549 1267 611 1301">3×3</div> <div data-bbox="735 972 1029 1267"></div> <div data-bbox="852 1267 914 1301">5×5</div> <div data-bbox="1035 972 1329 1267"></div> <div data-bbox="1142 1267 1204 1301">7×7</div> <div data-bbox="435 1301 729 1597"></div> <div data-bbox="549 1597 611 1630">9×9</div> <div data-bbox="735 1301 1029 1597"></div> <div data-bbox="839 1597 927 1630">15×15</div>

Kotak & GMO	<div data-bbox="435 304 729 600"></div> <div data-bbox="555 600 611 633">3×3</div> <div data-bbox="735 304 1029 600"></div> <div data-bbox="855 600 911 633">5×5</div> <div data-bbox="1035 304 1329 600"></div> <div data-bbox="1142 600 1198 633">7×7</div> <div data-bbox="435 633 729 929"></div> <div data-bbox="555 929 611 963">9×9</div> <div data-bbox="735 633 1029 929"></div> <div data-bbox="834 929 911 963">15×15</div>
Kotak & GME	<div data-bbox="435 969 729 1265"></div> <div data-bbox="536 1265 592 1299">3×3</div> <div data-bbox="735 969 1029 1265"></div> <div data-bbox="807 1265 863 1299">5×5</div> <div data-bbox="1035 969 1329 1265"></div> <div data-bbox="1078 1265 1134 1299">7×7</div> <div data-bbox="435 1299 729 1594"></div> <div data-bbox="544 1594 600 1628">9×9</div> <div data-bbox="735 1299 1029 1594"></div> <div data-bbox="794 1594 871 1628">15×15</div>

Berdasarkan Tabel 3.4, dapat dilihat hasil ekstraksi fitur menggunakan berbagai kombinasi dua bentuk dan lima ukuran. Pertama dilakukan ekstraksi fitur menggunakan SE ukuran 3×3 terhadap semua bentuk SE dan operasi. SE *disk* 3×3 dengan operasi GMO, fitur tidak terlihat. SE *disk* 3×3 dengan operasi GME atau GMD, fitur terlihat namun tidak jelas. SE kotak 3×3 dengan operasi GMO, fitur juga tidak terlihat. SE kotak 3×3 dengan operasi GME atau GMD fitur sedikit terlihat dan tidak memiliki perbedaan secara kasat mata.

Kemudian dilakukan menggunakan SE ukuran 5×5 . SE *disk* 5×5 dengan operasi GMO terdapat sangat sedikit fitur yang didapatkan yaitu pada bagian mata. SE *disk* 5×5 dengan operasi GME atau GMD fitur lebih jelas terlihat dan terdapat perbedaan, di mana operasi GME tepi pada bagian mata dengan bola mata bergabung sedangkan GMD tidak. Selain itu luas lubang hidung dan mulut juga lebih besar pada operasi GME dibandingkan operasi GMD. SE kotak 5×5 juga memiliki hasil yang sama dengan *disk* 5×5 dengan perbedaan yang tidak dapat dilihat kasat mata.

Kemudian dilakukan operasi menggunakan SE ukuran 7×7 . SE *disk* 7×7 dengan operasi GMO terdapat sangat sedikit fitur yang didapatkan yaitu pada bagian mata. SE *disk* 7×7 dengan operasi GME atau GMD fitur lebih jelas terlihat dan terdapat perbedaan, di mana operasi GME tepi pada bagian mata dengan bola mata bergabung sedangkan GMD tidak. Selain itu luas lubang hidung dan mulut juga lebih besar pada operasi GME dibandingkan operasi GMD. SE kotak 7×7 juga memiliki hasil yang sama dengan *disk* 7×7 dengan perbedaan yang tidak dapat dilihat kasat mata.

Pada operasi menggunakan SE ukuran 9×9 . SE *disk* 9×9 dengan operasi GMO fitur yang didapatkan tidak akurat karena tidak mengekstraksi tepi (bentuk) dari ekspresi wajah. SE *disk* 9×9 dengan operasi GME dan GMD memiliki fitur yang mirip dengan SE 5×5 , di mana operasi GME tepi pada bagian mata bergabung sedangkan GMD tidak. Selain itu luas pada lubang hidung dan mulut juga lebih luas dibandingkan dengan SE 5×5 , dan memiliki garis tepi yang lebih tebal. SE kotak 9×9 dengan operasi GMO, fitur yang didapatkan mirip dengan *disk* 9×9 dengan operasi GMO namun memiliki fitur yang lebih jelas terlihat. SE kotak 9×9

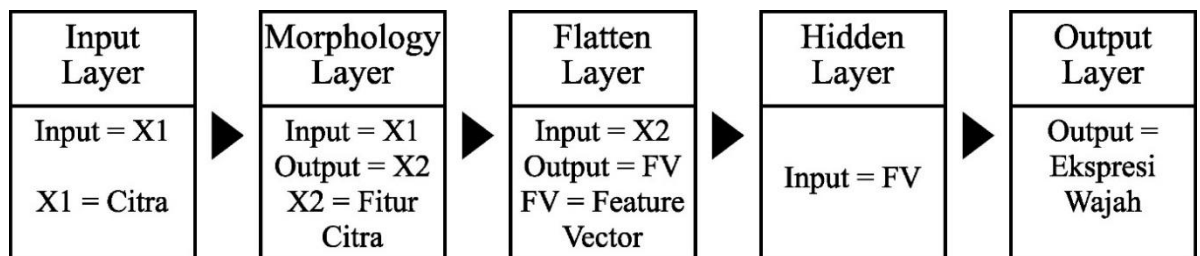
dengan operasi 9×9 GME fitur yang didapatkan terlihat dengan jelas dan mirip dengan hasil yang menggunakan SE *disk* 9×9 dengan operasi GME. Namun terdapat perbedaan fitur yang signifikan pada bagian hidung, di mana bentuk dari hidung sedikit berubah menjadi sedikit kotak. SE kotak 9×9 dengan operasi GMD, hasil yang didapatkan hampir sama dengan hasil yang menggunakan SE *disk* 9×9 dengan operasi GMD, hanya terdapat perbedaan pada tebal garis pada hidung.

Operasi menggunakan SE 15×15 , hasil yang didapatkan memiliki pola yang sama dengan operasi yang menggunakan SE 9×9 . SE *disk* atau kotak 15×15 dengan operasi GMO fitur lebih terlihat jelas dibandingkan dengan ukuran 9×9 . SE *disk* atau kotak 15×15 dengan operasi GME fitur lebih tebal, namun untuk SE kotak bagian hidung menjadi lebih kotak dari ukuran sebelumnya.

Berdasarkan dari hasil yang didapatkan dan analisis. SE *disk* 5×5 atau kotak 5×5 dengan operasi GMD memiliki hasil terbaik. Tepi pada bagian mata tergabung, bagian mulut tidak menjadi lebih luas. Namun pada bagian hidung menjadi lebih kecil dibandingkan citra original. Selain itu operasi menggunakan GME dengan SE *disk* 5×5 atau kotak 5×5 juga memberikan hasil yang baik. Tepi dari tiap komponen wajah terlihat namun tepi pada mata dan bola mata tergabung, namun memberikan hasil ekstrasi fitur bagian hidung lebih baik dibandingkan GMD. Karena luas hidung lebih sesuai pada GME dibandingkan GMD.

3.4.3.2 Pembentukan dan Pelatihan Model

Gambar 3.18 menunjukkan arsitektur dari MNN secara garis besar. Di mana terdapat 5 *layer* utama yang memiliki tugas masing-masing. Di mana terdapat beberapa variasi arsitektur Variasi pertama terdapat pada *morphology layer*, di mana akan digunakan dua jenis ekstraksi fitur berdasarkan pada hasil subbab 3.4.3.3. Variasi kedua terdapat pada *hidden layer*, di mana akan digunakan beberapa kombinasi *fully-connected layer*.



Gambar 3.18. Model MNN yang diusulkan

Input Layer adalah *layer* pertama dari model MNN yang bertugas untuk menerima *input* berupa citra. Di mana dimensi dari *input layer* itu sendiri sesuai dengan ukuran citra pada *dataset* yaitu 160×160 .

Kedua adalah *morphology layer*. Menerima masukan dari *input layer* kemudian dilakukan proses morfologi. Terdapat dua jenis morfologi *layer* yang akan digunakan. Morfologi *layer* pertama adalah *dilation layer* kemudian *subtraction layer*. Di mana lapisan morfologi jenis pertama menggunakan hasil terbaik dari operasi morfologi dilasi pada subbab 3.4.3.3. Morfologi *layer* jenis kedua adalah *erosion layer* dilanjutkan *subtraction layer*. Di mana lapisan morfologi jenis kedua ini menggunakan hasil terbaik dari operasi morfologi erosi pada subbab 3.4.3.3.

Lapisan ketiga adalah *flatten layer*, lapisan yang bertugas mengubah citra menjadi feature vector. Masukan dari lapisan ini adalah hasil dari *subtraction layer* pada lapisan morfologi. Di mana hasil dari *subtraction layer* adalah citra dengan ukuran 160×160 yang kemudian diubah menjadi satu dimensi yaitu 25600.

Lapisan keempat adalah *Fully Connected Layer* (FC Layer) yang bertugas untuk mempelajari dan menganalisa nilai feature vector dari *flatten layer*. Pada lapisan ini dilakukan beberapa konfigurasi FC Layer mulai dari jumlah FC Layer, dan jumlah Neuron pada FC Layer. Konfigurasi pertama akan diuji coba 2 FC Layer dengan masing-masing *neuron* adalah 512, dan 256. Konfigurasi kedua akan dicoba 1024, dan 512. Kemudian dari situ akan dicoba analisis mana yang lebih baik sehingga dapat dikonfigurasi lebih lanjut. Jika konfigurasi pertama memiliki hasil lebih baik artinya memungkinkan FC Layer untuk dibuat lebih sederhana dengan mengurangi jumlah *neuron*. Jika konfigurasi kedua lebih baik artinya terdapat kemungkinan untuk meningkatkan performa dari model karena *dataset* memiliki kompleksitas tinggi.

Beberapa lapisan akan dilakukan *tuning* parameter. *Tuning* pertama terdapat pada bentuk SE, ukuran SE, jenis operasi pada *morphological layer*. *Tuning* kedua terdapat pada FC-Layer yaitu fungsi aktivasi (*ReLU/Sigmoid/Tanh*), dan jumlah neuron pada tiap hidden layer. Selain arsitektur, pada proses pelatihan juga dilakukan *tuning* pada learning rate, jumlah epoch, dan batch size.

Terakhir adalah *Output Layer*, merupakan penentuan dari ekspresi berdasarkan dari bobot *hidden layer*. Di mana fungsi aktivasi yang digunakan untuk *output layer* adalah *softmax* yang artinya *output* berupa probabilitas dari tiap ekspresi. Kemudian untuk *loss function* yang akan digunakan adalah *categorical crossentropy*, di mana fungsi *loss* ini digunakan jika model memprediksi multi-kelas (*multi-class prediction*).

3.5 Time Table

SEMESTER 1						
AKTIVITAS	1	2	3	4	5	6
Studi Literatur						
Pembuatan Proposal						
Pengumpulan <i>Dataset</i> (Sekunder saja)						
SEMESTER 2						
Pengumpulan <i>Dataset</i> (Primer & Sekunder)						
Pembentukan <i>dataset</i>						
Pembentukan model						
Pelatihan, Pengujian Model (beserta <i>tuning</i> model)						
Menuliskan hasil penelitian (BAB 4)						
SEMESTER 3						
Pelatihan, Pengujian Model (beserta <i>tuning</i> model)						
Menuliskan hasil penelitian (BAB 4 & BAB 5)						
Pembuatan Jurnal Pertama						
SEMESTER 4						
Pembuatan Jurnal Pertama & Submit Jurnal						
Pembuatan Jurnal Kedua						
SEMESTER 5						
Pembuatan Jurnal Kedua & Submit Jurnal						

Bibliografi

- Abiodun, O. I., Jantan, A., Omolara, A. E., Dada, K. V., Mohamed, N. A., & Arshad, H. (2018). State-of-the-art in artificial neural network applications: A survey. *Heliyon*, 4(11), e00938. <https://doi.org/10.1016/j.heliyon.2018.e00938>
- Adak, C. (2013). Gabor filter and rough clustering based edge detection. *2013 International Conference on Human Computer Interactions (ICHCI)*, 1–5. Chennai, India: IEEE. <https://doi.org/10.1109/ICHCI-IEEE.2013.6887768>
- Ahmad, I., Moon, I., & Shin, S. J. (2018). Color-to-grayscale algorithms effect on edge detection—A comparative study. *2018 International Conference on Electronics, Information, and Communication (ICEIC)*, 1–4. Honolulu, HI: IEEE. <https://doi.org/10.23919/ELINFOCOM.2018.8330719>
- Alom, M. Z., Taha, T. M., Yakopcic, C., Westberg, S., Sidike, P., Nasrin, M. S., ... Asari, V. K. (2019). A State-of-the-Art Survey on Deep Learning Theory and Architectures. *Electronics*, 8(3), 292. <https://doi.org/10.3390/electronics8030292>
- Alzubaidi, L., Zhang, J., Humaidi, A. J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., ... Farhan, L. (2021). Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions. *Journal of Big Data*, 8(1), 53. <https://doi.org/10.1186/s40537-021-00444-8>
- Awad, M., & Khanna, R. (2015). *Efficient Machine Learning*. Apress Media, LLC. Retrieved from <https://library.oapen.org/>

- Babu, S. B. G. T., & Rao, C. S. (2023). Copy-Move Forgery Verification in Images Using Local Feature Extractors and Optimized Classifiers. *Big Data Mining and Analytics*, 6(3), 347–360.
<https://doi.org/10.26599/BDMA.2022.9020029>
- Ben-Hur, A., Ong, C. S., Sonnenburg, S., Schölkopf, B., & Rätsch, G. (2008). Support Vector Machines and Kernels for Computational Biology. *PLoS Computational Biology*, 4(10), e1000173.
<https://doi.org/10.1371/journal.pcbi.1000173>
- Bhavsar, H., & Panchal, M. H. (2012). *A Review on Support Vector Machine for Data Classification*. 1(10).
- Bovik, A. C. (2009). *Chapter 4—Basic Binary Image Processing*.
<https://doi.org/10.1016/b978-0-12-374457-9.00004-4>
- Bre, F., Gimenez, J. M., & Fachinotti, V. D. (2018). Prediction of wind pressure coefficients on building surfaces using artificial neural networks. *Energy and Buildings*, 158, 1429–1441.
<https://doi.org/10.1016/j.enbuild.2017.11.045>
- Chen, L.-F., & Yen, Y.-S. (2007). Taiwanese facial expression image database. *Brain Mapping Laboratory, Institute of Brain Science, National Yang-Ming University, Taipei, Taiwan*.
- Coman, M. N., Rontescu, C., Bogatu, A. M., & Cicic, D. T. (2021). Analysis of robotic welding possibilities of a car chassis assembly. *IOP Conference Series: Materials Science and Engineering*, 1182(1), 012015.
<https://doi.org/10.1088/1757-899X/1182/1/012015>

- Davidson, J. L., & Ritter, G. X. (1990, July 1). *Theory of morphological neural networks* (R. Arrathoon, Ed.). Los Angeles, CA. <https://doi.org/10.1117/12.18085>
- Deepthi. B, Gupta, P., Rai, P., & Arora, H. (2022). Assessing the Dynamics of AI Driven Technologies in Indian Banking and Financial Sector. *Vision: The Journal of Business Perspective*, 097226292210873. <https://doi.org/10.1177/09722629221087371>
- Deshmukh, S., Patwardhan, M., & Mahajan, A. (2016). Survey on real-time facial expression recognition techniques. *IET Biometrics*, 5(3), 155–163. <https://doi.org/10.1049/iet-bmt.2014.0104>
- El Naqa, I., & Murphy, M. J. (2015). What Is Machine Learning? In I. El Naqa, R. Li, & M. J. Murphy (Eds.), *Machine Learning in Radiation Oncology* (pp. 3–11). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-18305-3_1
- Eleyan, A. (2023). Statistical local descriptors for face recognition: A comprehensive study. *Multimedia Tools and Applications*. <https://doi.org/10.1007/s11042-023-14482-2>
- Frith, C. (2009). Role of facial expressions in social interactions. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 364(1535), 3453–3458. <https://doi.org/10.1098/rstb.2009.0142>
- Garcia, E., Jimenez, M. A., De Santos, P. G., & Armada, M. (2007). The evolution of robotics research. *IEEE Robotics & Automation Magazine*, 14(1), 90–103. <https://doi.org/10.1109/MRA.2007.339608>

- George, J. M., & Dane, E. (2016). Affect, emotion, and decision making. *Organizational Behavior and Human Decision Processes*, 136, 47–55.
<https://doi.org/10.1016/j.obhdp.2016.06.004>
- Gonzalez, R. C., Woods, R. E., & Masters, B. R. (2009). Digital Image Processing, Third Edition. *Journal of Biomedical Optics*, 14(2), 029901.
<https://doi.org/10.1117/1.3115362>
- Goyal, M. (2011). *Morphological Image Processing*. 2(4).
- Grandini, M., Bagli, E., & Visani, G. (2020, August 13). *Metrics for Multi-Class Classification: An Overview*. arXiv. Retrieved from <http://arxiv.org/abs/2008.05756>
- Hegel, F., Muhl, C., Wrede, B., Hielscher-Fastabend, M., & Sagerer, G. (2009). Understanding Social Robots. *2009 Second International Conferences on Advances in Computer-Human Interactions*, 169–174. Cancun, Mexico: IEEE. <https://doi.org/10.1109/ACHI.2009.51>
- Janiesch, C., Zschech, P., & Heinrich, K. (2021). Machine learning and deep learning. *Electronic Markets*, 31(3), 685–695.
<https://doi.org/10.1007/s12525-021-00475-2>
- Johal, W. (2020). Research Trends in Social Robots for Learning. *Current Robotics Reports*, 1(3), 75–83. <https://doi.org/10.1007/s43154-020-00008-3>
- Kanade, T., Cohn, J. F., & Tian, Y. (2000). Comprehensive database for facial expression analysis. *Proceedings Fourth IEEE International Conference on Automatic Face and Gesture Recognition (Cat. No. PR00580)*, 46–53. IEEE.

- Khanzode, K. C. A. (2020). *ADVANTAGES AND DISADVANTAGES OF ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING: A LITERATURE REVIEW*.
- Kim, S. (2023). *Applications of Convolution in Image Processing with MATLAB*.
- Li, Z., Liu, F., Yang, W., Peng, S., & Zhou, J. (2022). A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects. *IEEE Transactions on Neural Networks and Learning Systems*, 33(12), 6999–7019. <https://doi.org/10.1109/TNNLS.2021.3084827>
- Lmp, P. (2011). *Robust Facial Expression Recognition Based on Local Mono - { I. (Iccit)*.
- Lucey, P., Cohn, J. F., Kanade, T., Saragih, J., Ambadar, Z., & Matthews, I. (2010). The extended cohn-kanade dataset (ck+): A complete dataset for action unit and emotion-specified expression. *2010 Ieee Computer Society Conference on Computer Vision and Pattern Recognition-Workshops*, 94–101. IEEE.
- Lyons, M. J. (2021, July 27). “Excavating AI” Re-excavated: Debunking a Fallacious Account of the JAFFE Dataset. arXiv. Retrieved from <http://arxiv.org/abs/2107.13998>
- Lyons, M. J., Kamachi, M., & Gyoba, J. (2020). *Coding Facial Expressions with Gabor Wavelets (IVC Special Issue)*. <https://doi.org/10.5281/zenodo.4029679>
- Madenda, S. (2015). *PENGOLAHAN CITRA & VIDEO DIGITAL*. Jakarta: Erlangga.

- Madsen, C. A. B. C. W., Lucca, G., Daniel, G. B., & Adamatti, D. F. (2012). FURG Smart Games: A Proposal for an Environment to Game Development with Software Reuse and Artificial Intelligence. In R. Benlamri (Ed.), *Networked Digital Technologies* (pp. 369–381). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-30507-8_32
- Maind, S. B., & Wankar, P. (n.d.). Research Paper on Basic of Artificial Neural Network. *International Journal on Recent and Innovation Trends in Computing and Communication*, 2(1).
- Mohammad, T., & Ali, Md. L. (2011). Robust facial expression recognition based on Local Monotonic Pattern (LMP). *14th International Conference on Computer and Information Technology (ICCIT 2011)*, 572–576. Dhaka, Bangladesh: IEEE. <https://doi.org/10.1109/ICCITechn.2011.6164854>
- Montero, C. D. L., Gundul, B. J. A., Frayco, J. G., & Jr, J. Q. C. (2023). *CONVOLUTIONAL NEURAL NETWORK-BASED IMAGE CLASSIFICATION FOR IMPROVED COCONUT DISEASE IDENTIFICATION. II*.
- Mukherjee, D., Gupta, K., Chang, L. H., & Najjaran, H. (2022). A Survey of Robot Learning Strategies for Human-Robot Collaboration in Industrial Settings. *Robotics and Computer-Integrated Manufacturing*, 73, 102231. <https://doi.org/10.1016/j.rcim.2021.102231>

- Munawar, H. S., Aggarwal, R., Qadir, Z., Khan, S. I., Kouzani, A. Z., & Mahmud, M. A. P. (2021). A Gabor Filter-Based Protocol for Automated Image-Based Building Detection. *Buildings*, 11(7), 302. <https://doi.org/10.3390/buildings11070302>
- O'Shea, K., & Nash, R. (2015, December 2). *An Introduction to Convolutional Neural Networks*. arXiv. Retrieved from <http://arxiv.org/abs/1511.08458>
- Ou, J., Bai, X.-B., Pei, Y., Ma, L., & Liu, W. (2010). Automatic Facial Expression Recognition Using Gabor Filter and Expression Analysis. *2010 Second International Conference on Computer Modeling and Simulation*, 215–218. Sanya, China: IEEE. <https://doi.org/10.1109/ICCMS.2010.45>
- Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, É. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12(85), 2825–2830.
- Pisner, D. A., & Schnyer, D. M. (2019). Support vector machine. In *Machine Learning: Methods and Applications to Brain Disorders* (pp. 101–121). IEEE. <https://doi.org/10.1016/B978-0-12-815739-8.00006-7>
- Poynton, C. A. (2003). *Digital video and HDTV: Algorithms and interfaces*. Amsterdam ; Boston: Morgan Kaufmann Publishers.
- Ravi, R., Yadhukrishna, S. V., & Prithviraj, R. (2020). A Face Expression Recognition Using CNN & LBP. *2020 Fourth International Conference on Computing Methodologies and Communication (ICCMC)*, 684–689. Erode, India: IEEE. <https://doi.org/10.1109/ICCMC48092.2020.ICCMC-000127>

- Rawal, N., & Stock-Homburg, R. M. (2022). Facial Emotion Expressions in Human–Robot Interaction: A Survey. *International Journal of Social Robotics*, 14(7), 1583–1604. <https://doi.org/10.1007/s12369-022-00867-0>
- Retto, J. (2017). *SOPHIA, FIRST CITIZEN ROBOT OF THE WORLD*.
- Robert, R. (2023). *IMPLEMENTASI FEATURE DESCRIPTOR DAN MACHINE LEARNING UNTUK KLASIFIKASI EKSPRESI WAJAH*. Universitas Gunadarma, Indonesia.
- Robertson, G., & Watson, I. (2014). A Review of Real-Time Strategy Game AI. *AI Magazine*, 35(4), 75–104. <https://doi.org/10.1609/aimag.v35i4.2478>
- Sawardekar, S., & Naik, S. R. (2018). *Facial Expression Recognition using Efficient LBP and CNN*. 05(06).
- Shan, C., Gong, S., & McOwan, P. W. (2009). Facial expression recognition based on Local Binary Patterns: A comprehensive study. *Image and Vision Computing*, 27(6), 803–816. <https://doi.org/10.1016/j.imavis.2008.08.005>
- Shen, Y., Shih, F. Y., Zhong, X., & Chang, I.-C. (2019). Deep Morphological Neural Networks. *International Journal of Pattern Recognition and Artificial Intelligence*, 36(12), 2252023. <https://doi.org/10.1142/S0218001422520231>
- Shinde, P. P., & Shah, S. (2018). A Review of Machine Learning and Deep Learning Applications. *2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA)*, 1–6. Pune, India: IEEE. <https://doi.org/10.1109/ICCUBEA.2018.8697857>

- Soille, P. (2004). *Morphological Image Analysis*. Berlin, Heidelberg: Springer Berlin Heidelberg. <https://doi.org/10.1007/978-3-662-05088-0>
- Turabzadeh, S., Meng, H., Swash, R., Pleva, M., & Juhar, J. (2018). Facial Expression Emotion Detection for Real-Time Embedded Systems. *Technologies*, 6(1), 17. <https://doi.org/10.3390/technologies6010017>
- Viola, P., & Jones, M. J. (2001). Rapid object detection using a boosted cascade of simple features. *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, 1*, I-511-I-518. Kauai, HI, USA: IEEE Comput. Soc. <https://doi.org/10.1109/CVPR.2001.990517>
- Viola, P., & Jones, M. J. (2004). *Robust Real-Time Face Detection*.
- Wang, H., Xiong, J., Yao, Z., Lin, M., & Ren, J. (2017). Research Survey on Support Vector Machine. *Proceedings of the 10th EAI International Conference on Mobile Multimedia Communications*. Presented at the 10th EAI International Conference on Mobile Multimedia Communications, Chongqing, People's Republic of China. Chongqing, People's Republic of China: EAI. <https://doi.org/10.4108/eai.13-7-2017.2270596>
- Wang, Y.-Q. (2014). An Analysis of the Viola-Jones Face Detection Algorithm. *Image Processing On Line*, 4, 128–148. <https://doi.org/10.5201/ipol.2014.104>

Yuqian Zhao, Weihua Gui, & Zhencheng Chen. (2006). Edge Detection Based on Multi-Structure Elements Morphology. *2006 6th World Congress on Intelligent Control and Automation*, 9795–9798. Dalian, China: IEEE.
<https://doi.org/10.1109/WCICA.2006.1713908>