

PENGEMBANGAN MODEL KLASIFIKASI  
*MORPHOLOGICAL NEURAL NETWORK* UNTUK  
SISTEM PENGENALAN EKSPRESI WAJAH

# 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 ekstraksi 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 ekstraksi fitur atau menggunakan model *Convolutional Neural Network* (CNN) dari model DL. Pada penelitian (Shan, Gong & McOwan, 2009), digunakan *Local Binary Pattern* (LBP) untuk ekstraksi fitur dan SVM sebagai model untuk mengenal ekspresi wajah berdasarkan dari hasil ekstraksi fitur LBP. Pada penelitian (Mohammad & Ali, 2011), digunakan *Local Monotonic Pattern* (LMP) untuk ekstraksi 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 ekstraksi fitur yaitu *Morphological Neural Network*

(MNN). Hasil penelitian (Shen, Zhong & Shih, 2022) 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.

## 1.4 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.

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 ekstraksi 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 ekstraksi fitur *Local Monotonic Pattern* (LMP) untuk pengenalan ekspresi wajah. Hasil dari LMP kemudian digunakan sebagai *input*

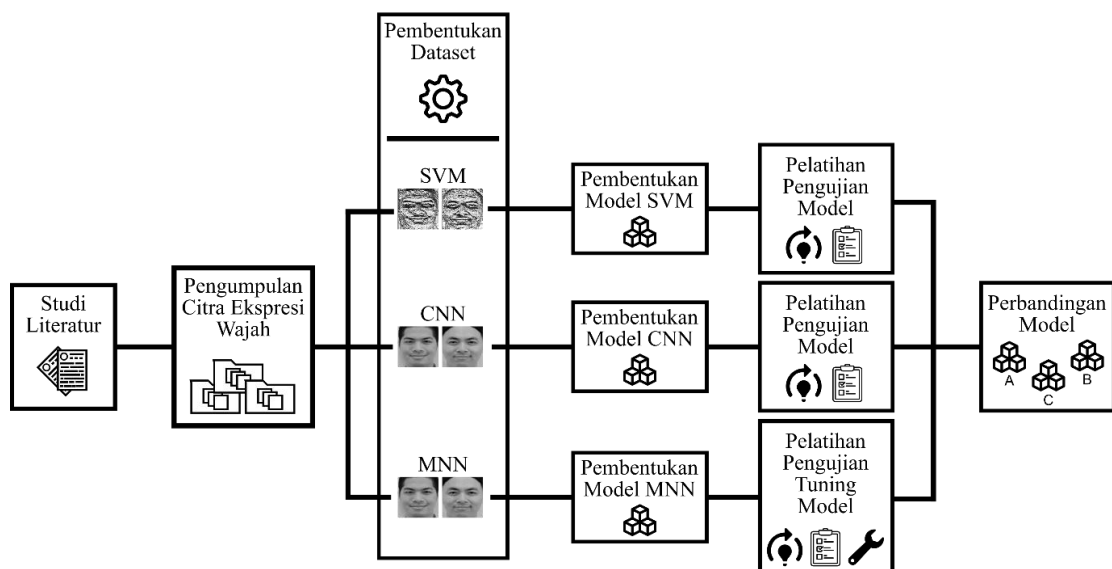
model SVM. Pada tahun 2019, dikembangkan sebuah model DL MNN yang memanfaatkan *opening* dari morfologi matematika untuk mengklasifikasi angka dan rambu lalu lintas (Shen et al., 2019). Pada tahun 2020, dilakukan penelitian pengenalan ekspresi wajah menggunakan *Local Binary Pattern* (LBP) sebagai ekstraksi 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.



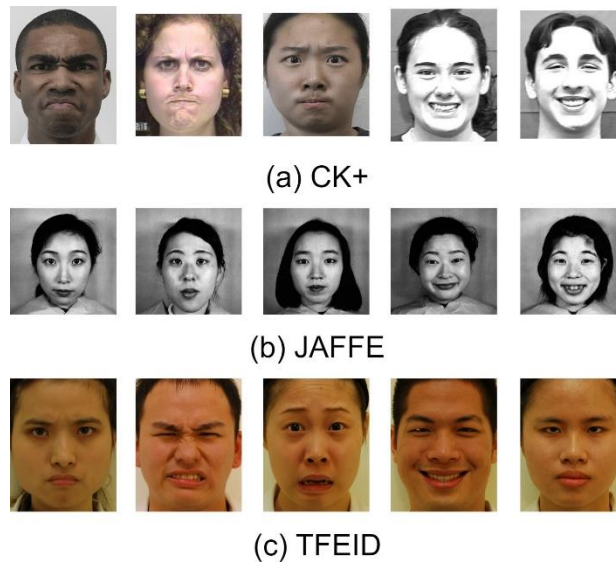
Gambar 3.2. Contoh 7 jenis ekspresi wajah

*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 ditiap kelas, ukuran citra dikisaran 481×600, ruang warna RGB.

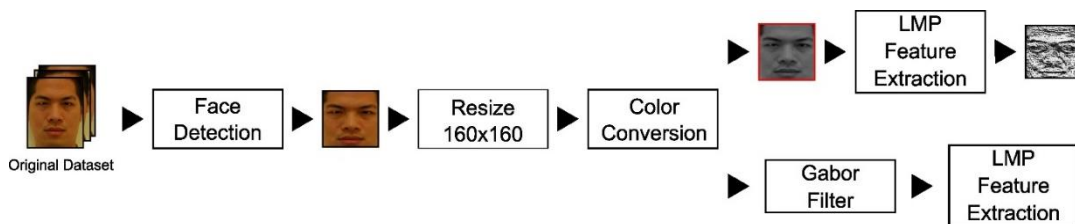
Tabel 3.1. Detail *dataset* sekunder

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

### 3.3 Pembentukan *Dataset*

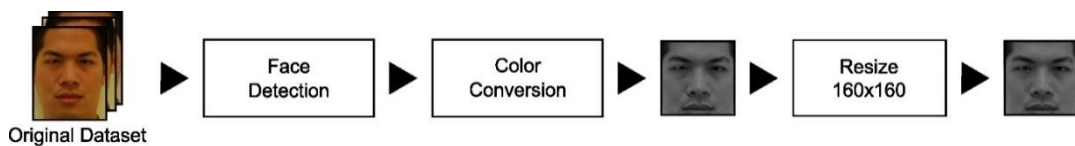
Secara garis besar, dalam pembuatan model AI (khususnya ML dan DL) terdapat proses yang berperan penting, yaitu *preprocessing dataset* seperti ekstrasi fitur, penyesuaian ukuran citra, dan augmentasi (Deshmukh et al., 2016; Franchi et al., 2020; Mohammad & Ali, 2011; Ravi et al., 2020; Sawardekar & 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 ekstrasi 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 diekstrasi fiturnya menggunakan LMP. Terakhir, terdapat dua proses ekstrasi fitur berbeda. Proses ekstrasi fitur pertama menggunakan LMP (Robert, 2023). Proses ekstrasi fitur kedua adalah usulan dari penelitian ini, di mana pertama diaplikasikan Gabor *Filter* terlebih dahulu kemudian diekstrasi menggunakan LMP.



Gambar 3.5. Pembentukan *dataset* untuk SVM

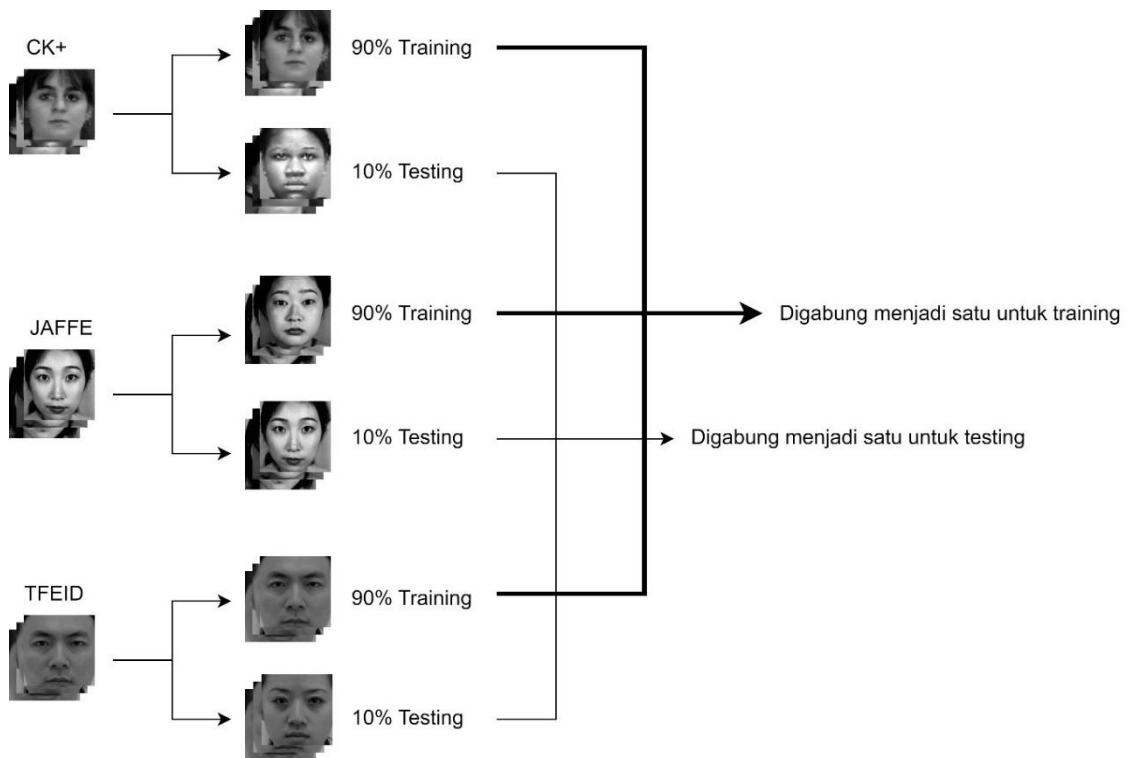
Gambar 3.6 menunjukkan alur pembentukan *dataset* untuk CNN dan MNN. Terdapat 3 proses yang akan dilakukan. Pertama, dilakukan deteksi wajah menggunakan VJA guna mengurangi *noise*. Kemudian dilakukan konversi warna dari RGB ke *Grayscale* karena fitur warna tidak dibutuhkan untuk mengenali ekspresi wajah. Terakhir, mengubah ukuran citra untuk menyamakan semua ukuran citra dan sesuai dengan dimensi *input* model. 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  $0^\circ$  hingga  $45^\circ$ . 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

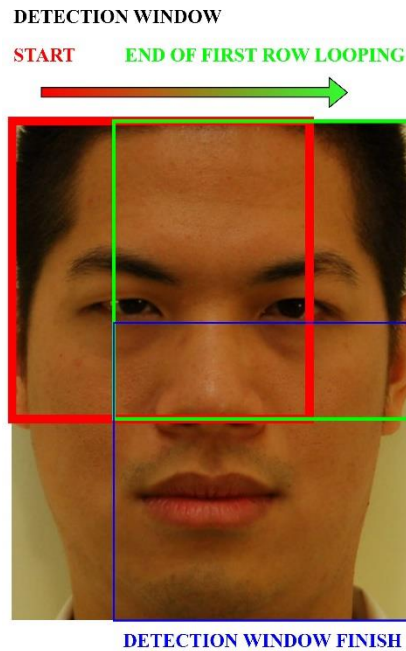
```

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)
            : threshold  $\rightarrow TH(a)$  hingga  $TH(e)$ 
Output     : koordinat wajah  $\rightarrow [x1, y1], [w2, h2]$ 

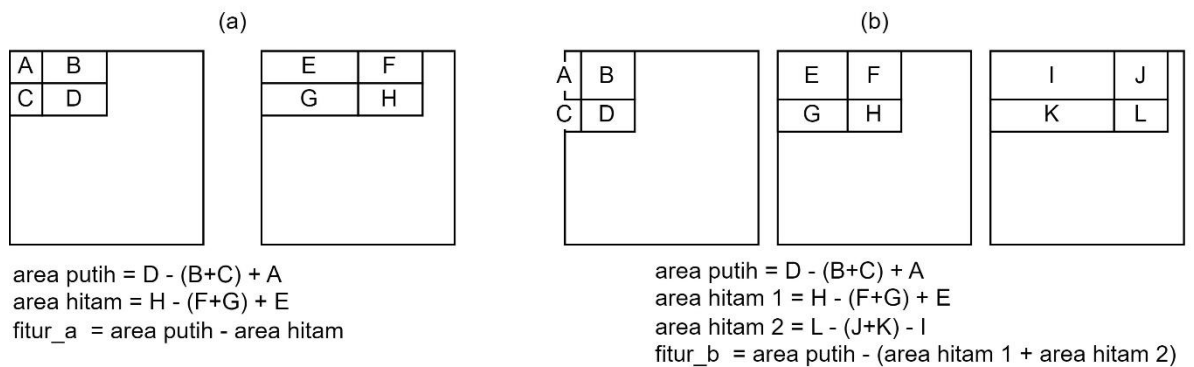
for i = 0 to (w-w2) do
    for j = 0 to (h-h2) do
        # hitung fitur Haar (a) pada Gambar 2.14
        hitung fitur_a = area putih Gambar 2.14 (a) -
1      area hitam Gambar 2.14 (a) untuk detection window
2      (i:i+w2, j:j+h2)
3      if fitur_a < TH_a do
4          continue
5
6      # hitung fitur 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)

```





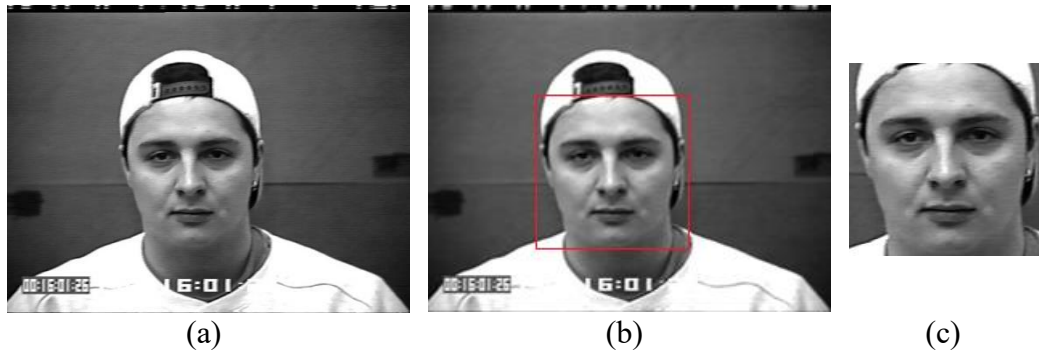
Gambar 3.8. Pergeseran Detection Window



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 Haar tersebut 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

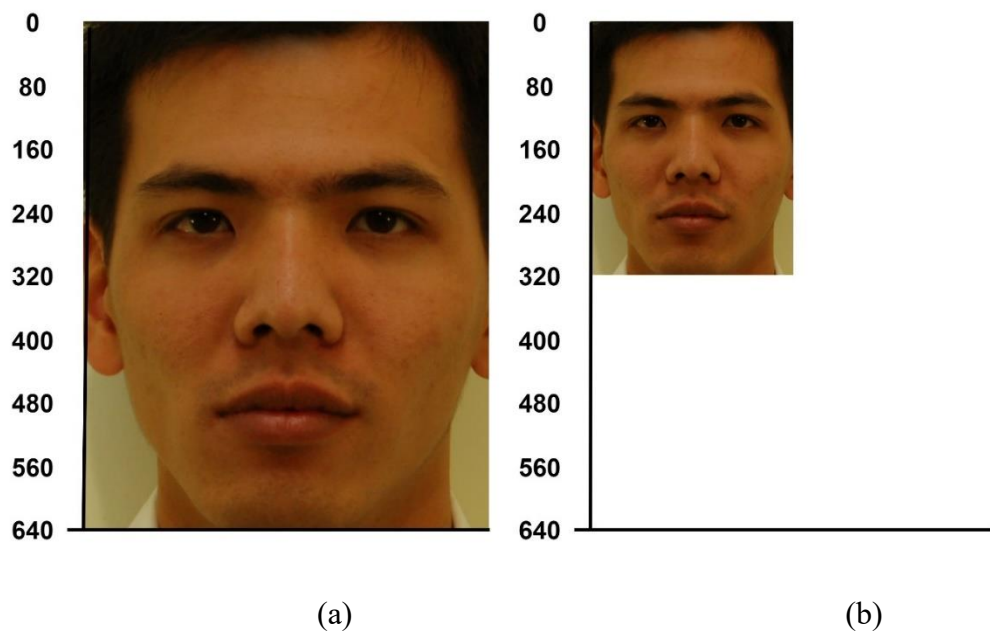
Proses perubahan ukuran citra menggunakan metode *bicubic interpolation*. Algoritma 3.3 menunjukkan cara kerja dari *bicubic interpolation*. *Input* berupa citra (yang ingin diubah ukurannya) dan rasio (skala ukuran yang diinginkan). Parameter berupa koefisien  $a$  yang dapat mempengaruhi koordinat 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  $4 \times 4$  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  $4 \times 4$  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  $dH = H*r, dW = W*r, h = 1/\text{ratio}$  #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  $x1 = 1 + x - \text{floor}(x)$   
9         Calculate  $x2 = x - \text{flood}(x)$   
10        Calculate  $x3 = \text{floor}(x) + 1 - x$   
11        Calculate  $x4 = \text{flood}(x) + 2 - x$   
12        Calculate  $y1 = 1 + y - \text{floor}(y)$   
13        Calculate  $y2 = y - \text{flood}(y)$   
14        Calculate  $y3 = \text{floor}(y) + 1 - y$   
15        Calculate  $y4 = \text{flood}(y) + 2 - y$   
16        Create matrix  $\text{mat\_l}$  dimension  $[1,4],$   
         $\text{mat\_r}$  dimension  $[4,1]$   
        #  $h()$  menggunakan pers (2.19)  
17        Calculate  $\text{mat\_l}[0,0] = h(x1),$   
         $\text{mat\_l}[0,1] = h(x2), \text{mat\_l}[0,2] = h(x3), \text{mat\_l}[0,3] =$   
         $h(x4)$  # bobot horizontal  
18        Calculate  $\text{mat\_r}[0,0] = h(y1),$   
         $\text{mat\_l}[0,1] = h(y2), \text{mat\_l}[0,2] = h(y3), \text{mat\_l}[0,3] =$   
         $h(y4)$  # bobot vertical  
19        Create matrix  $\text{mat\_m}$  dimension  $[4,4]$   
20         $\text{mat\_l}[0,0] = \text{img}[\bar{y}-y1, x-x1]$   
21         $\text{mat\_l}[1,0] = \text{img}[\bar{y}-y2, x-x1]$   
22         $\text{mat\_l}[2,0] = \text{img}[\bar{y}+y3, x-x1]$   
23         $\text{mat\_l}[3,0] = \text{img}[\bar{y}+y4, x-x1]$   
24         $\text{mat\_l}[0,1] = \text{img}[\bar{y}-y1, x-x2]$   
25         $\text{mat\_l}[1,1] = \text{img}[\bar{y}-y2, x-x2]$   
26         $\text{mat\_l}[2,1] = \text{img}[\bar{y}+y3, x-x2]$   
27         $\text{mat\_l}[3,1] = \text{img}[\bar{y}+y4, x-x2]$   
28         $\text{mat\_l}[0,2] = \text{img}[\bar{y}-y1, x+x3]$   
29         $\text{mat\_l}[1,2] = \text{img}[\bar{y}-y2, x+x3]$   
30         $\text{mat\_l}[2,2] = \text{img}[\bar{y}+y3, x+x3]$   
31         $\text{mat\_l}[3,2] = \text{img}[\bar{y}+y4, x+x3]$   
32         $\text{mat\_l}[0,3] = \text{img}[\bar{y}-y1, x+x4]$   
33         $\text{mat\_l}[1,3] = \text{img}[\bar{y}-y2, x+x4]$   
34         $\text{mat\_l}[2,3] = \text{img}[\bar{y}+y3, x+x4]$   
35         $\text{mat\_l}[3,3] = \text{img}[\bar{y}+y4, x+x4]$ 
```

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



(a) (b)  
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  $YCbCr$ . *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  $YCbCr$ ).

#### 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] +
4           0.587*img_rgb[i,j,1] + 0.114*img_rgb[i,j,2]
5     end for
6 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 ( $\lambda$ ), psi ( $\varphi$ ), sigma  $\sigma$ , dan gamma  $\gamma$  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*.

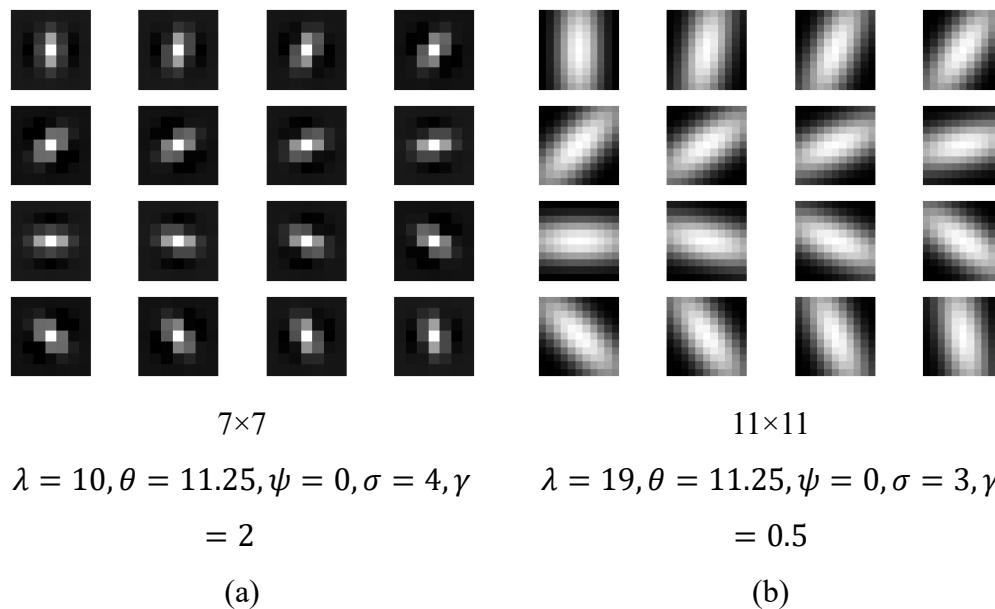
Algoritma 3.5. Gabor Filter

<b>Input</b>	: 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 $\rightarrow \text{max\_int}$
<b>Output</b>	: Citra ukuran $[H, W] \rightarrow \text{output}$

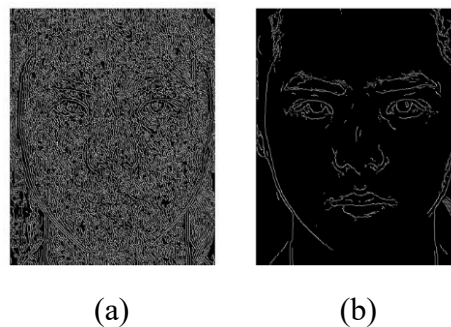
  

```
1 for i = 0 to 180 with increment 180/num_filter do
2   Create Gabor filter filter with kernel size =
   ksize, lambda = lambd, Theta = i, Psi = psi, Sigma =
   sigma, Gamma = gamma
3 # proses konvolusi
4 For i = 0 to num_filter do
5   Output = img  $\otimes$  filter[i]
6 # proses deteksi tepi
7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

Perubahan nilai parameter dilakukan pada  $\lambda$ ,  $\gamma$ , dan  $\sigma$  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  $8 \times 2$  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.

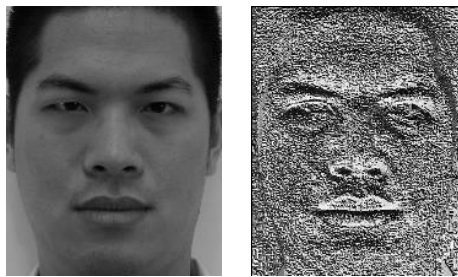
Algoritma 3.6. *Local Monotonic Pattern*

```
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

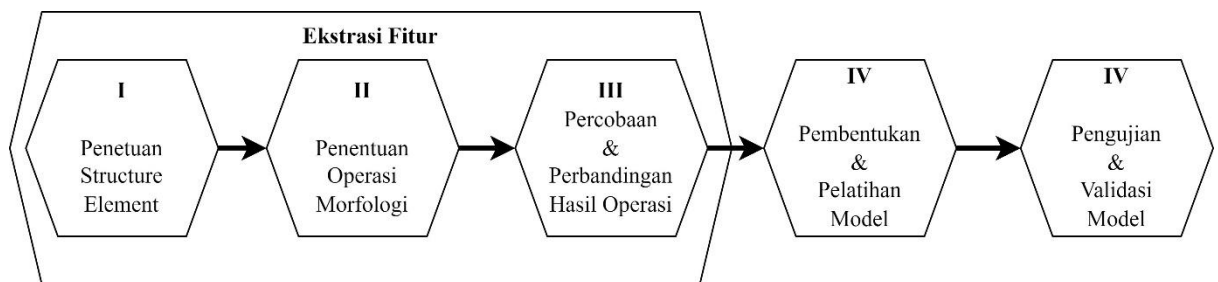
Dalam penelitian sebelumnya (Robert, 2023), model CNN yang digunakan adalah MobileNetV2, yang merupakan *pre-trained model*. Penelitian tersebut menggunakan *dataset* TFEID yang memiliki keterbatasan baik dalam jumlah maupun variasi dari etnis ekspresi wajah manusia. Oleh karena itu, dalam penelitian 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	56x56x128
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
	Avg Pool / 1	Pool 7×7
	FC / 1	1024×7
	Softmax / 1	Classifier

### 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.

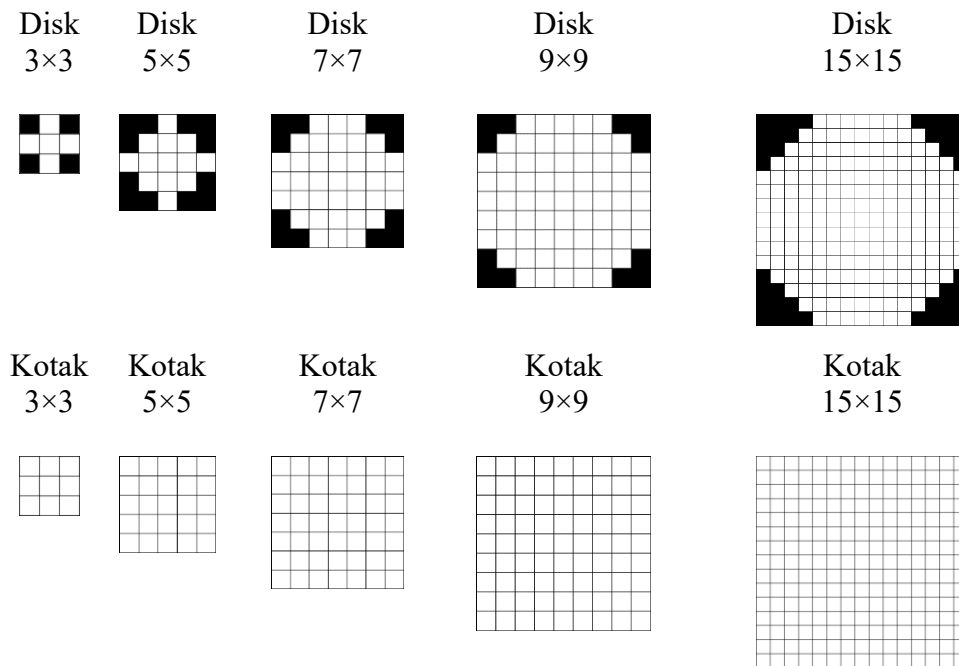


Gambar 3.16. Proses pembentukan model

#### 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  $3 \times 3$ ,  $5 \times 5$ ,  $9 \times 9$ , dan  $15 \times 15$ , 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







```

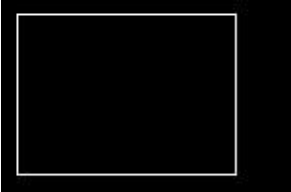









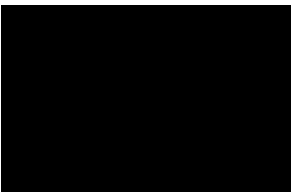
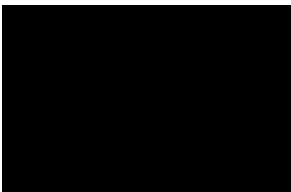
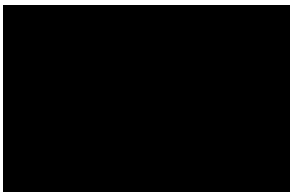
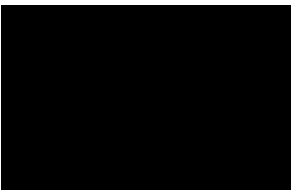
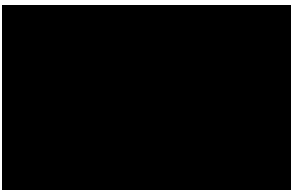
Pada Algoritma 3.9. terdapat dua *input* yang digunakan, pertama adalah citra original dengan ukuran  $W \times H$ . Kedua adalah *structure element* dengan ukuran  $N \times N$ . Hasil dari algoritma ini adalah sebuah fitur dalam bentuk citra dengan ukuran  $(W \times H)$ . Baris 1 dan 2 dilakukan perulangan 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.

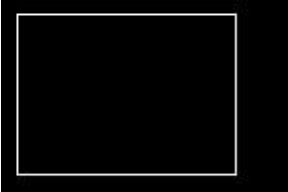




3.4.3.1.3 Percobaan dan Perbandingan Hasil Operasi Morfologi

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

<div>Original Image</div> <div></div>	
SE & Operasi	Operasi dan Hasil
Disk & Original - Opening	<div><div></div><div></div><div></div></div> <div><div>3×3</div><div>5×5</div><div>7×7</div></div>
	<div><div></div><div></div></div> <div><div>9×9</div><div>15×15</div></div>

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

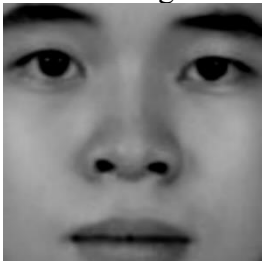

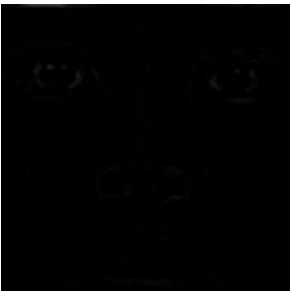
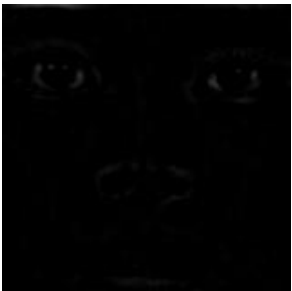


Kotak & <i>Original</i> - Erosi			
	3×3	5×5	7×7
Kotak & <i>Dilasi</i> - <i>Original</i>			
	9×9	15×15	











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  $5 \times 5$ ,  $7 \times 7$ ,  $9 \times 9$ , dan  $15 \times 15$ . 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 kearah luar persegi, sedangkan tepi pada GME menebal kearah 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

<div>Citra Original</div> 	
SE & Operasi	Operasi dan Hasil
Disk & GMO	<div>    </div> <div> <div>3×3</div> <div>5×5</div> <div>7×7</div> </div>
	<div>   </div> <div> <div>9×9</div> <div>15×15</div> </div>

<p><i>Disk &amp; GME</i></p>	<div data-bbox="435 138 729 430">  <p>3×3</p> </div> <div data-bbox="735 138 1029 430">  <p>5×5</p> </div> <div data-bbox="1035 138 1329 430">  <p>7×7</p> </div> <div data-bbox="435 464 729 755">  <p>9×9</p> </div> <div data-bbox="735 464 1029 755">  <p>15×15</p> </div>
<p><i>Disk &amp; GMD</i></p>	<div data-bbox="435 809 729 1100">  <p>3×3</p> </div> <div data-bbox="735 809 1029 1100">  <p>5×5</p> </div> <div data-bbox="1035 809 1329 1100">  <p>7×7</p> </div> <div data-bbox="435 1134 729 1426">  <p>9×9</p> </div> <div data-bbox="735 1134 1029 1426">  <p>15×15</p> </div>



Kotak & GMO			
	3×3	5×5	7×7
Kotak & GME			
	9×9	15×15	
Kotak & GME			
	3×3	5×5	7×7
Kotak & GME			
	9×9	15×15	

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  $5 \times 5$ . SE *disk*  $5 \times 5$  dengan operasi GMO terdapat sangat sedikit fitur yang didapatkan yaitu pada bagian mata. SE *disk*  $5 \times 5$  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  $5 \times 5$  juga memiliki hasil yang sama dengan *disk*  $5 \times 5$  dengan perbedaan yang tidak dapat dilihat kasat mata.

Kemudian dilakukan operasi menggunakan SE ukuran  $7 \times 7$ . SE *disk*  $7 \times 7$  dengan operasi GMO terdapat sangat sedikit fitur yang didapatkan yaitu pada bagian mata. SE *disk*  $7 \times 7$  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  $7 \times 7$  juga memiliki hasil yang sama dengan *disk*  $7 \times 7$  dengan perbedaan yang tidak dapat dilihat kasat mata.

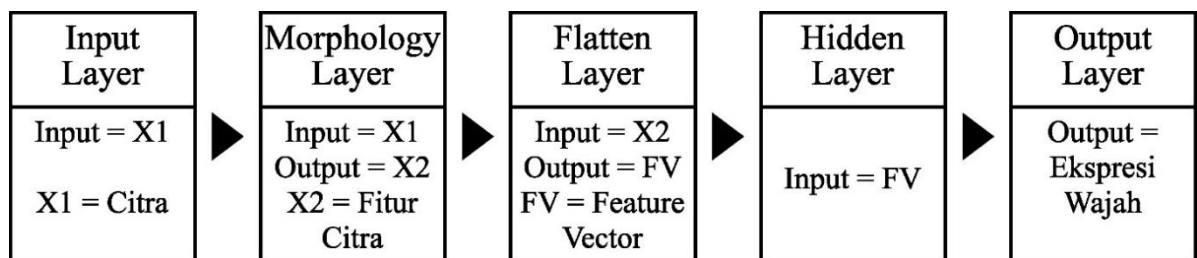
Pada operasi menggunakan SE ukuran  $9 \times 9$ . SE *disk*  $9 \times 9$  dengan operasi GMO fitur yang didapatkan tidak akurat karena tidak mengekstraksi tepi (bentuk) dari ekspresi wajah. SE *disk*  $9 \times 9$  dengan operasi GME dan GMD memiliki fitur yang mirip dengan SE  $5 \times 5$ , di mana operasi GME tepi pada bagian mata bergabung sedangkan GMD tidak. Selain itu luas pada lubang hidung dan mulut juga lebih luas dibandingkan dengan SE  $5 \times 5$ , dan memiliki garis tepi yang lebih tebal. SE kotak  $9 \times 9$  dengan operasi GMO, fitur yang didapatkan mirip dengan *disk*  $9 \times 9$  dengan operasi GMO namun memiliki fitur yang lebih jelas terlihat. SE kotak  $9 \times 9$  dengan operasi  $9 \times 9$  GME fitur yang didapatkan terlihat dengan jelas dan mirip dengan hasil yang menggunakan SE *disk*  $9 \times 9$  dengan operasi GME. Namun terdapat perbedaan fitur yang signifikan pada bagian hidung, di mana bentuk dari hidung sedikit berubah menjadi sedikit kotak. SE kotak  $9 \times 9$  dengan operasi GMD, hasil yang didapatkan hampir sama dengan hasil yang menggunakan SE *disk*  $9 \times 9$  dengan operasi GMD, hanya terdapat perbedaan pada tebal garis pada hidung.

Operasi menggunakan SE  $15 \times 15$ , hasil yang didapatkan memiliki pola yang sama dengan operasi yang menggunakan SE  $9 \times 9$ . SE *disk* atau kotak  $15 \times 15$  dengan operasi GMO fitur lebih terlihat jelas dibandingkan dengan ukuran  $9 \times 9$ . SE *disk* atau kotak  $15 \times 15$  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*  $5 \times 5$  atau kotak  $5 \times 5$  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  $5 \times 5$  atau kotak  $5 \times 5$  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  $160 \times 160$ .

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  $160 \times 160$  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*).