BAB 3
ANALISIS DAN PERANCANGAN SISTEM

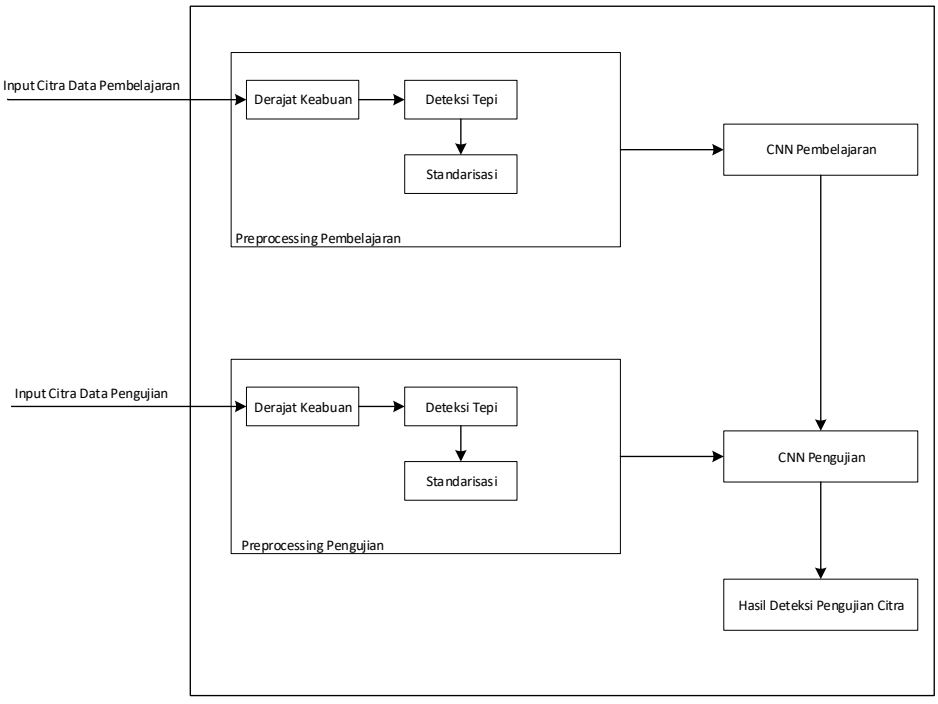
3.1 Analisis Masalah

Analisis masalah merupakan gambaran masalah berdasarkan latar belakang yang telah disampaikan yaitu mendeteksi hama pada daun teh menggunakan metode tertentu berdasarkan data citra daun, sehingga mendapatkan tingkat akurasi dalam mendeteksi.

Berdasarkan analisis masalah yang telah disampaikan, penulis akan mengimplementasikan suatu metode dalam menyelesaikan permasalahan yang timbul yaitu, metode *Convolutional Neural Network* (CNN) dalam mengklasifikasi data citra daun teh untuk mendapatkan hasil akurasi macam-macam jenis hama.

3.2 Analisis Sistem

Analisis proses didefinisikan sebagai dari proses utama ke dalam sub-sub proses dengan tujuan untuk mengidentifikasikan permasalahan-permasalahan yang ada dalam menghasilkan akurasi yang baik dari deteksi hama pada daun teh. Tahapan proses tersebut dapat dilihat pada gambar 3.1.



Gambar 3. 1 Deskripsi Umum Sistem

Pada deskripsi umum sistem diatas, proses awal yang dilakukan yaitu memasukkan citra daun hama ke dalam aplikasi. Langkah selanjutnya melalui tahap pengolahan citra meliputi Derajat Keabuan (*Grayscale*), Deteksi Tepi *(Edge Detection Sobel*) dan Normalisasi. Dari tahap pengolahan citra akan didapatkan nilai *pixel*, nilai *pixel* kemudian diolah kembali pada tahap klasifikasi dengan metode CNN baik pembelajaran maupun pengenalan. Hasil dari citra yang dimasukkan didapatkan akurasi dan pengenalan hama pada daun teh.

3.2.1 *Preprocessing*

Pada tahap ini citra penuh yang dijadikan masukan data diproses mulai dari dikonversi citra RGB ke citra keabuan dan deteksi tepi citra. Keluaran dari proses ini adalah citra daun nantinya digunakan untuk informasi kepada user tentang objek citra mana yang akan diolah oleh sistem. Tahapan proses ini dapat dilihat pada gambar 3.2.

Gambar 3. 2 *Flowchart Preprocessing*

1. RGB ke deajat keabuan

   Citra digital inputan memiliki tiga chanel warna RGB 3D dengan pixel 192x192 dikonversi ke satu chanel warna keabuan 1D bertujuan untuk mengurangi beban proses sistem.

2. Deteksi Tepi (*Sobel*)

   Operasi yang dijalankan untuk mendeteksi garis tepi (*edges*) yang membatasi dua wilayah citra homogen yang memiliki tingkat kecerahan yang berbeda. Tujuannnya adalah untuk mengubah citra 1D menjadi bentuk kurva dan dimana intensitas kecerahan berubah secara drastis.

3. Standarisasi

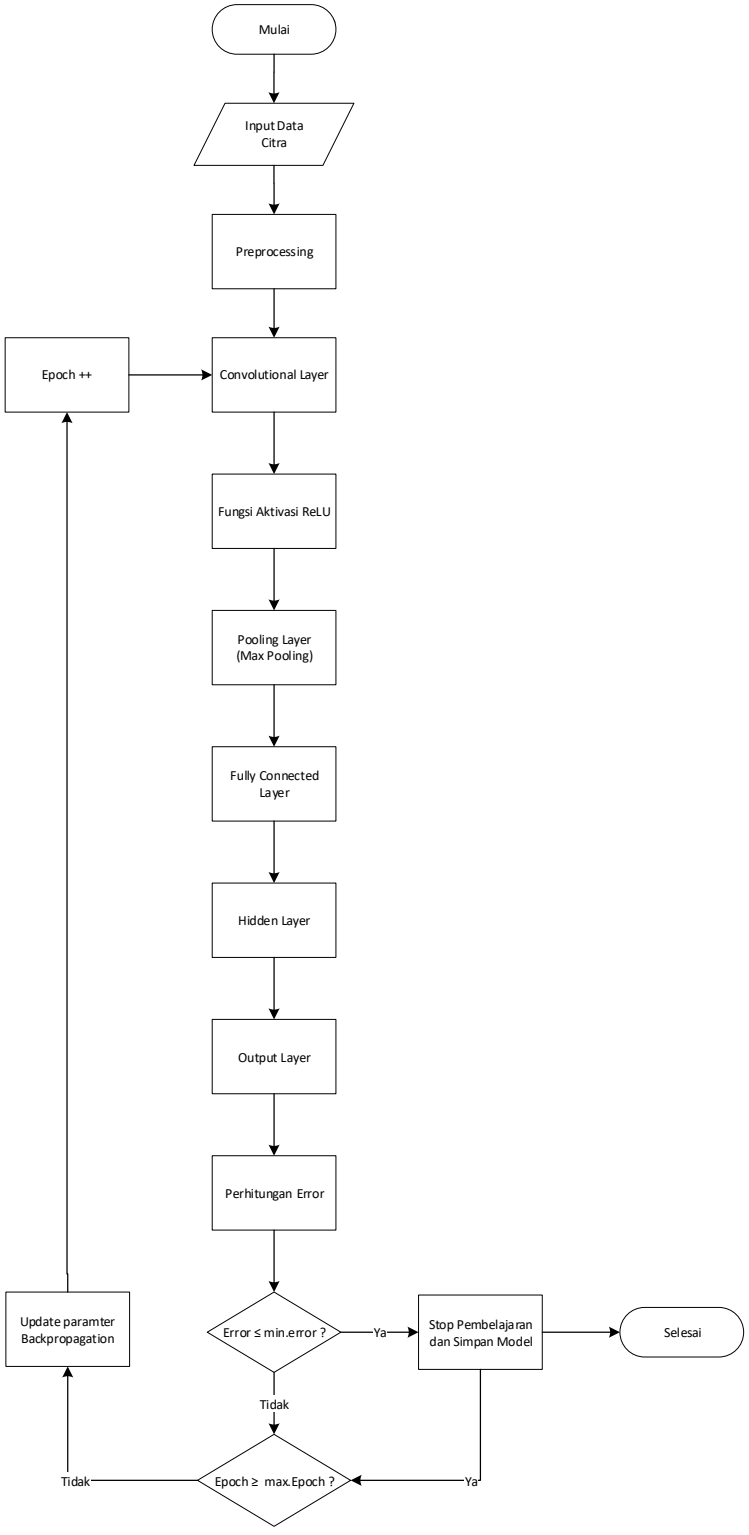
   Setelah dilakukasn deteksi tepi maka dilakukan standaisasi. Standarisasi mengubah nilai pixel tujuannya agar diperoleh interval data yang sesuai dengan fungsi aktivasi yang digunakan.

Contoh hasil dari tahap preprocessing dapat dilihat pada gambar 3.3.

Gambar 3. 3 Hasil Preprocessing

3.2.2 Pembelajaran

Proses Pembelajaran merupakan tahapan dimana CNN dilatih untuk memperoleh akurasi yang tinggi dari klasifikasi yang dilakukan. Dalam tahap ini user dapat memberikan label kesetiap objek citra selain itu dilakukan proses pembentukan *Convolutional Neural Network*. Tahapan pembelajaran dapat dilihat pada gambar 3.4.



Gambar 3. 4 *Flowchart* Pembelajaran

Keluaran dari proses ini adalah arsitektur beserta hasil pembelajaran convolutional neural network. Arsitektur berserta hasil pembelajaran neural network ini akan digunakan proses selanjutnya yakni pengenalan objek citra. Pada tahapan pembalajaran ini dapat dijelaskan sebagai berikut:

1. *Convolution*

   Convolution melakukan operasi konvolusi pada *output* dari *layer* sbelumnya. *Layer* tersebut adalah proses utama yang mendasari sebuah CNN. Konvolusi adalah suatu istilah matematis yang berarti mengaplikasi sebuah fungsi lain pada *output* fungsi lain secara berulang. Dalam pengolahan citra, konvolusi berarti mengaplikasikan sebuah *kernel*.

   Tujuan dilakukannya konvolusi pada data citra adalah untuk mengekstrasi fitur dari citra *input*. Konvolusi akan menghasilkan transformasi *linier* dari data *input* sesuai informasi spasial pada data. Bobot pada *layer* tersebut mengspesifikasikan *kernel* konvolusi yang digunakan, sehingga kernel konvolusi dapat dilatih berdasarkan *input* pada CNN.

2. Fungsi Aktivasi *ReLU*

   Fungsi ini melakukan *Thresholding* dengan nilai nol terhadap nilai piksel pada input citra. Aktivasi ini membuat seluruh nilai piksel yang bernilai kurang dari nol pada suatu citra akan dijadikan 0.

3. *Pooling Layer*

   Tujuan dari penggunaan pooling layer adalah mengurangi dimensi dari feature map (downsampling), sehingga mempercepat komputasi karena parameter yang harus diupdate semakin sedikit dan mengatasi overfitting. Pooling layer yang digunakan adalah *max pooling*. *Max pooling* membagi output dari *convolutional layer* menjadi beberapa *grid* kecil lalu mengambil nilai maksimal dari setiap *grid* untuk menyusun matriks citra yang telah direduksi.

4. *Fully Connected Layer*

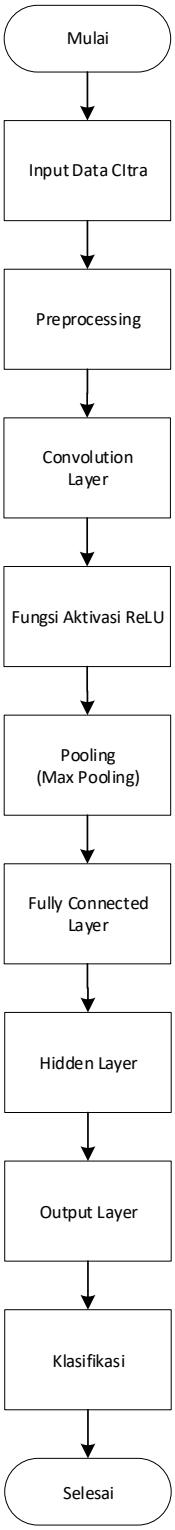
   Setiap *neuron* pada *convolutional layer* perlu ditransformasi menjadi data satu dimensi terlebih dahulu sebelum dapat dimasukkan ke dalam sebuah *fully connected layer*. Karena hal tersebut menyebabkan data kehilangan informasi spasialnya dan tidak reversible, *fully connected layer* hanya dapat diimplementasikan di akhir jaringan. Dari fully connected layer bekerja dimana citra vector akan melalui proses konvolusi dan *Max pooling* untuk mereduksi ukuran citranya dan memperbanyak neuronnya. Sehingga terbentuk banyak jaringan yang mana menambah variant data untuk dipelajari.

5. Perhitungan Error

   Dalam tahap ini, jika maksimal epoch dan nilai minimal error sudah selesai maka berhenti proses pembelajaran dan hasil model disimpan. Sebalikanya, jika maksimal epoch dan nilai minimal error belum selesai maka update parameter *backpropagation*.

3.2.3 Pengujian

Proses pengujian merupakan proses klasifikasi menggunakan bobot dan bias dari hasil proses pembelajaran yang dihasilkan dari perhitungan *Softmax* sehingga menghasilkan deteksi pengenalan dari citra yang dimasukkan. Tahapan pengenalan dapat dilihat pada gambar 3.5.



Gambar 3. 5 *Flowchart* Pengenalan

3.3 Analisis data masukan

Data masukan yang digunakan adalah data citra daun yang diambil dari kebun teh yang berlokasi di Malabar Pangalengan Kabupaten Bandung milik PT. Perkebunan Nusantara VIII. Data masukan berupa citra daun yang terserang hama, jumlah citra daun yang terserang hama berjumlah 440 data dimana citra daun tersebut terserang hama *Blister blight* dan *Hellopeltis*. Dari 440 citra daun, 400 dijadikan data pembelajaran yang terbagi 200 citra daun yang terserang *Blister* blight dan 200 citra daun yang terserang *Hellopeltis*, sedangkan 40 dijadikan data pengujian yang terbagi 20 citra daun yang terserang *Blister* blight dan 20 citra daun yang terserang *Hellopeltis*. Ukuran *pixel* data pembelajaran dan pengujian menggunakan 192x192 *pixel*. Dari pixel 192x192 dilakukan *resize* menjadi 28x28, *resize* ini berpengaruh pada hasil pengujian dimana pixel 28x28 baik dalam mendeteksi dipenelitian ini. Contoh citra daun yang terserang oleh hama dapat dilihat pada Gambar 3.6 dan Gambar 3.7.

a. *Hellopeltis*

Gambar 3. 6 Citra Daun yang Terserang Hama *Hellopeltis*

b. *Blister blight*

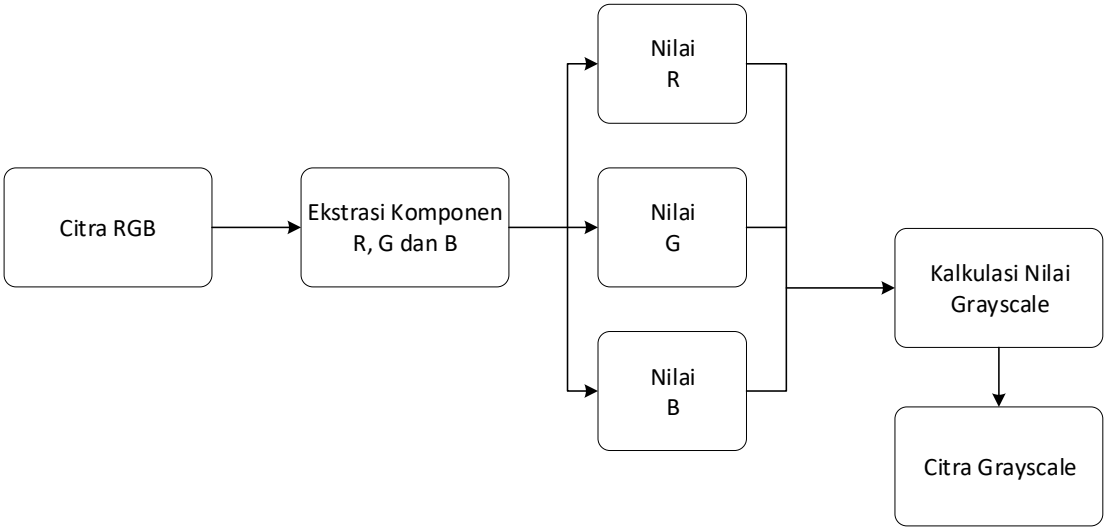
Gambar 3. 7 Citra Daun yang Terserang Hama *Blister blight*

3.4 Analisis *Preprocessing*

Analisis citra masukan akan diolah ke dalam pra proses yaitu proses derajat keabuan (*Grayscale*), deteksi tepi (*edge detection sobel*) dan standarisasi, citra masukan dilakukan pengecekan terhadap standarisasi dari deteksi tepi tersebut.

3.4.1 Citra keabuan

Citra penuh yang dijadikan masukan data diproses mulai dari dikonversi citra RGB ke citra keabuan (*Grayscale*). Mengubah format warna menjadi *derajat keabuan* berfungsi untuk mengecilkan range warna menjadi 0 sampai dengan 255. Proses ini akan memudahkan ketika ingin melakukan deteksi tepi (*edge detection*). Gambar 3.8 merupakan proses derajat keabuan.



Gambar 3. 8 Block Diagram Proses *Grayscale* Citra

Adapun langkah-langkah yang dilakukan sebagai berikut.

1. Warna citra dikelompokkan berdasarkan nilai *red, green* dan *blue*
2. Kemudian menggunakan persamaan (2.2), maka akan didapatkan nilai warna derajat keabuan.
3. Nilai derajat keabuan yang didapat menggunakan nilai RGB pada setiap *pixel.*

Adapun contoh gambar citra daun RGB pada gambar 3.6.

Gambar 3. 9 Citra Daun RGB

Dari citra daun RGB diatas memiliki nilai *pixel* 28x28 dapat dilihiat pada gambar 3.10.

xy	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	[234 234 232]	[235 235 233]	[236 236 234]	[237 237 235]	[238 238 236]	[238 238 236]	[240 240 238]	[189 202 170]	[102 141 49]	[107 145 57]	[113 149 65]	[118 152 67]	[119 153 68]	[123 157 77]
2	[235 235 233]	[236 236 234]	[237 237 235]	[238 238 236]	[238 238 236]	[238 238 236]	[230 232 224]	[144 170 106]	[104 143 53]	[108 146 60]	[117 152 70]	[119 153 69]	[118 153 67]	[122 156 74]
3	[236 236 234]	[237 237 235]	[237 237 235]	[238 238 236]	[238 238 236]	[239 239 236]	[195 207 177]	[98 138 43]	[101 140 51]	[108 145 58]	[116 151 67]	[118 153 68]	[117 154 67]	[119 155 70]
4	[237 237 235]	[238 238 236]	[238 238 236]	[239 239 237]	[239 239 237]	[230 233 224]	[139 166 98]	[92 131 40]	[98 136 48]	[108 145 58]	[116 151 67]	[120 155 73]	[115 152 66]	[119 155 69]
5	[238 238 236]	[238 238 236]	[239 239 237]	[239 239 237]	[239 239 237]	[210 218 197]	[96 134 45]	[87 127 37]	[92 131 42]	[106 143 57]	[118 153 69]	[120 155 69]	[117 153 68]	[118 155 69]
6	[238 238 236]	[238 238 236]	[239 239 237]	[240 240 238]	[238 239 236]	[176 192 149]	[90 130 40]	[81 121 32]	[91 130 39]	[100 139 49]	[111 148 60]	[116 152 64]	[119 154 68]	[121 156 72]
7	[239 239 237]	[238 238 236]	[240 240 238]	[240 240 238]	[237 238 234]	[143 165 111]	[89 126 37]	[84 123 34]	[89 129 37]	[99 138 46]	[100 139 47]	[111 148 60]	[118 154 68]	[122 156 71]
8	[239 239 237]	[239 239 237]	[240 240 238]	[241 241 239]	[229 231 225]	[137 146 104]	[100 111 38]	[89 127 36]	[90 129 38]	[98 138 46]	[93 134 35]	[104 142 50]	[117 154 69]	[122 156 73]
9	[239 239 237]	[240 240 238]	[240 240 238]	[237 238 235]	[153 170 126]	[83 107 26]	[107 115 40]	[94 131 42]	**[93 131 39]**	**[97 135 41]**	**[92 121 26]**	**[94 132 35]**	**[113 150 62]**	**[118 153 67]**
10	[240 240 238]	[240 240 238]	[241 241 239]	[187 197 170]	[89 126 29]	[73 114 15]	[95 129 38]	[99 138 48]	**[102 139 51]**	**[108 141 54]**	**[103 119 35]**	**[84 123 23]**	**[108 147 54]**	**[115 152 61]**
11	[239 239 237]	[241 241 239]	[231 233 228]	[121 147 77]	[82 124 17]	[76 120 17]	[96 135 42]	[105 142 54]	**[112 148 61]**	**[116 151 66]**	**[100 137 44]**	**[82 124 18]**	**[95 136 33]**	**[108 145 46]**
12	[240 240 238]	[240 240 238]	[184 193 168]	[91 129 32]	[87 129 25]	[81 123 20]	[97 132 40]	[107 143 55]	**[118 152 69]**	**[119 152 69]**	**[102 140 49]**	**[89 131 27]**	**[85 127 18]**	**[99 138 30]**
13	[240 240 238]	[220 223 214]	[105 131 57]	[87 126 19]	[91 132 30]	[86 121 22]	[108 116 37]	[110 142 56]	**[122 155 74]**	**[119 153 69]**	**[102 140 44]**	**[95 135 33]**	**[86 129 19]**	**[98 136 23]**
14	[236 237 232]	[152 168 123]	[84 121 15]	[91 130 24]	[94 135 36]	[97 127 33]	[109 123 39]	[109 144 55]	**[121 155 70]**	**[119 152 66]**	**[98 137 38]**	**[90 130 23]**	**[91 131 20]**	**[108 140 29]**
15	[167 181 143]	[86 121 24]	[88 128 20]	[93 133 30]	[99 139 42]	[105 141 47]	[103 138 46]	[104 141 52]	**[110 146 57]**	**[112 146 56]**	**[93 133 29]**	**[88 128 17]**	**[101 135 25]**	**[108 139 29]**
16	[84 123 19]	[83 125 18]	[93 133 32]	[101 139 42]	[105 143 45]	[112 147 54]	[111 138 50]	[101 135 42]	**[103 140 46]**	**[105 141 44]**	**[91 131 23]**	**[93 130 18]**	**[110 141 31]**	**[92 124 18]**
17	[90 131 27]	[96 135 38]	[102 140 46]	[107 143 49]	[104 136 38]	[105 133 39]	[120 119 41]	[102 127 37]	[110 145 54]	[104 142 41]	[94 132 23]	[104 137 24]	[101 134 23]	[92 130 19]
18	[100 138 42]	[106 143 50]	[110 146 55]	[109 143 53]	[115 127 43]	[130 132 50]	[117 119 35]	[103 131 37]	[109 144 52]	[106 142 41]	[100 135 23]	[107 140 28]	[89 127 13]	[87 129 14]
19	[106 142 53]	[107 143 53]	[108 144 55]	[107 137 51]	[136 135 66]	[158 151 75]	[142 140 58]	[105 120 30]	[101 136 36]	[100 137 32]	[110 142 30]	[95 132 20]	[89 129 14]	[87 128 14]
20	[113 148 63]	[111 146 59]	[107 143 53]	[103 131 45]	[141 138 73]	[158 150 76]	[152 147 67]	[126 113 42]	[94 121 22]	[102 136 25]	[105 139 28]	[94 134 20]	[94 135 20]	[94 134 20]
21	[116 150 65]	[109 145 55]	[103 140 47]	[96 129 36]	[124 130 63]	[158 152 84]	[142 143 61]	[113 113 31]	[94 120 16]	[107 127 27]	[104 134 30]	[99 140 28]	[97 139 24]	[103 142 33]
22	[112 148 58]	[96 135 37]	[96 135 36]	[92 131 30]	[90 118 27]	[113 120 43]	[112 125 35]	[96 127 18]	[105 136 23]	[100 130 25]	[95 132 25]	[95 135 25]	[95 135 21]	[101 141 30]
23	[95 135 37]	[84 125 20]	[87 127 22]	[89 128 24]	[95 133 29]	[111 122 36]	[103 109 25]	[101 131 19]	[101 134 22]	[93 133 22]	[91 132 23]	[93 134 25]	[93 133 22]	[91 132 19]
24	[84 125 22]	[82 124 16]	[82 123 16]	[84 125 20]	[99 137 37]	[110 130 40]	[96 117 17]	[101 134 19]	[95 134 22]	[95 136 29]	[98 137 33]	[94 135 29]	[90 130 21]	[91 132 22]
25	[78 120 15]	[80 122 14]	[82 123 16]	[81 123 16]	[91 131 26]	[92 124 23]	[97 129 15]	[97 134 21]	[100 139 31]	[104 143 41]	[111 148 50]	[107 144 45]	[100 140 37]	[104 143 42]
26	[74 117 10]	[78 120 11]	[82 123 17]	[82 124 16]	[86 125 17]	[91 122 15]	[95 129 17]	[100 138 30]	[109 146 43]	[117 153 55]	[122 157 63]	[117 152 57]	[110 147 49]	[112 148 52]
27	[72 114 7]	[76 117 10]	[82 123 16]	[81 122 12]	[82 117 9]	[93 122 14]	[84 123 11]	[88 129 17]	[104 143 39]	[114 150 51]	[118 154 58]	[118 154 59]	[116 152 57]	[109 146 49]
28	[77 113 12]	[94 110 24]	[85 124 17]	[79 118 10]	[86 117 9]	[84 119 9]	[82 123 8]	[80 122 9]	[82 124 13]	[90 131 20]	[101 140 34]	[118 153 58]	[114 151 56]	[108 146 48]

xy	15	16	17	18	19	20	21	22	23	24	25	26	27	28
1	[120 156 73]	[117 153 67]	[114 151 58]	[106 145 48]	[101 142 43]	[89 131 23]	[83 124 13]	[100 137 24]	[98 135 23]	[108 140 36]	[109 143 37]	[111 148 42]	[110 147 42]	[102 140 34]
2	[123 157 74]	[119 154 68]	[113 149 58]	[105 144 47]	[99 141 38]	[87 129 18]	[92 130 17]	[101 133 24]	[108 129 31]	[117 127 44]	[111 124 40]	[102 127 28]	[106 142 36]	[102 140 34]
3	[124 158 74]	[121 155 69]	[113 149 58]	[103 142 41]	[94 136 28]	[87 128 16]	[103 133 26]	[111 123 35]	[145 146 72]	[160 158 86]	[161 159 90]	[128 134 60]	[98 126 26]	[98 137 30]
4	[123 158 72]	[122 156 70]	[109 147 53]	[100 141 38]	[89 132 19]	[100 137 25]	[105 130 28]	[140 145 64]	[164 163 92]	[165 163 91]	[171 168 101]	[165 162 98]	[109 124 40]	[96 135 26]
5	[121 155 69]	[117 153 65]	[106 143 47]	[96 137 31]	[95 135 22]	[110 145 34]	[109 140 31]	[150 155 75]	[172 170 98]	[172 169 103]	[173 170 105]	[171 168 103]	[127 135 61]	[98 133 25]
6	[121 155 69]	[106 140 45]	[98 138 34]	[93 135 24]	[104 142 30]	[104 142 29]	[110 144 32]	[147 153 71]	[173 171 98]	[169 165 101]	[177 173 110]	[173 171 109]	[128 138 66]	[95 131 25]
7	[120 154 67]	[95 129 29]	[87 130 17]	[99 138 24]	[104 142 30]	[106 146 32]	[109 145 34]	[132 142 56]	[170 168 98]	[170 164 99]	[176 173 107]	[168 166 109]	[117 131 57]	[88 125 22]
8	[111 149 58]	[95 136 28]	[91 131 17]	[108 143 32]	[102 142 28]	[111 150 39]	[108 148 35]	[109 132 31]	[156 156 82]	[168 164 89]	[167 164 90]	[155 154 93]	[102 122 45]	[89 123 27]
9	**[102 142 44]**	**[94 136 25]**	[104 141 29]	[100 139 26]	[96 139 23]	[101 143 28]	[100 142 26]	[100 135 18]	[133 142 46]	[160 159 78]	[160 157 81]	[144 144 79]	[105 122 48]	[109 138 47]
10	**[99 139 36]**	**[101 139 27]**	[106 143 29]	[90 133 17]	[92 135 19]	[95 139 22]	[94 137 18]	[110 135 25]	[156 160 70]	[165 164 85]	[165 163 92]	[163 161 106]	[150 158 107]	[207 214 186]
11	**[98 137 27]**	**[108 143 29]**	[96 135 19]	[86 130 12]	[87 131 15]	[91 133 18]	[90 131 14]	[127 139 45]	[169 167 95]	[170 168 104]	[168 166 110]	[166 165 120]	[202 203 184]	[239 240 237]
12	**[110 145 33]**	**[104 139 26]**	[88 129 13]	[85 129 11]	[88 131 16]	[92 131 20]	[88 126 11]	[130 138 56]	[166 164 99]	[169 167 112]	[167 164 117]	[168 167 128]	[220 220 211]	[240 240 238]
13	**[113 146 34]**	**[89 128 13]**	[85 127 12]	[87 131 16]	[92 134 22]	[93 135 20]	[89 128 14]	[124 134 55]	[164 161 101]	[168 165 111]	[166 164 113]	[176 175 140]	[231 231 226]	[239 239 237]
14	**[100 135 23]**	**[87 129 14]**	[95 137 23]	[104 144 35]	[104 144 37]	[101 142 30]	[98 138 21]	[106 127 36]	[155 154 96]	[165 163 106]	[167 165 112]	[184 183 153]	[237 238 234]	[239 239 237]
15	**[94 133 19]**	**[98 139 26]**	[115 152 46]	[119 156 55]	[116 153 49]	[105 145 31]	[103 143 26]	[98 133 22]	[119 130 58]	[151 151 95]	[148 149 96]	[162 169 124]	[237 238 233]	[239 239 237]
16	**[94 120 20]**	**[105 143 32]**	[115 153 47]	[114 151 44]	[108 146 38]	[98 139 25]	[94 135 20]	[94 135 22]	[94 127 24]	[95 117 33]	[105 125 45]	[194 203 169]	[240 240 237]	[238 238 236]
17	[98 131 23]	[93 133 19]	[92 135 20]	[97 137 22]	[93 135 20]	[91 132 17]	[93 135 20]	[101 140 30]	[102 141 32]	[98 136 29]	[143 168 89]	[233 235 226]	[239 240 237]	[239 239 237]
18	[87 131 14]	[87 131 14]	[85 128 12]	[88 131 15]	[85 128 13]	[91 133 21]	[100 140 31]	[108 147 43]	[108 147 42]	[104 144 35]	[178 195 141]	[240 241 237]	[239 240 237]	[238 238 236]
19	[88 131 17]	[94 136 23]	[94 137 23]	[91 134 19]	[95 136 26]	[104 144 38]	[112 150 48]	[116 153 55]	[116 153 52]	[131 164 68]	[220 226 205]	[240 240 237]	[239 239 237]	[238 238 236]
20	[97 137 26]	[101 141 31]	[102 142 33]	[102 142 32]	[104 143 37]	[109 147 44]	[115 152 54]	[119 155 58]	[130 164 66]	[196 210 164]	[240 241 236]	[240 240 238]	[239 239 237]	[239 239 236]
21	[105 144 37]	[109 147 43]	[114 151 49]	[112 150 45]	[111 149 46]	[111 148 46]	[117 153 56]	[123 159 60]	[147 174 85]	[232 236 222]	[241 242 237]	[240 240 238]	[239 239 237]	[238 239 236]
22	[104 143 36]	[110 148 45]	[113 150 48]	[111 149 47]	[109 148 44]	[109 148 45]	[119 155 56]	[122 157 51]	[190 205 156]	[240 241 235]	[241 241 237]	[240 240 236]	[239 239 236]	[239 239 235]
23	[90 129 19]	[99 135 31]	[106 144 40]	[108 146 42]	[110 148 44]	[114 151 49]	[115 147 42]	[154 176 99]	[233 236 224]	[241 242 237]	[240 241 236]	[240 241 236]	[239 240 235]	[238 239 234]
24	[91 124 20]	[116 139 49]	[113 149 51]	[108 145 43]	[114 150 47]	[114 145 42]	[124 149 56]	[215 221 197]	[241 242 236]	[241 242 236]	[240 241 236]	[239 240 235]	[239 240 235]	[238 239 234]
25	[105 141 41]	[115 150 54]	[108 145 45]	[106 143 42]	[114 147 43]	[116 138 42]	[194 203 165]	[240 241 235]	[241 242 236]	[241 242 237]	[240 241 236]	[239 240 235]	[239 240 235]	[238 239 234]
26	[106 143 46]	[108 145 48]	[105 142 42]	[111 144 45]	[113 138 38]	[149 160 99]	[233 236 226]	[240 241 236]	[241 242 237]	[240 241 236]	[239 240 235]	[239 240 235]	[239 240 235]	[238 239 234]
27	[108 145 50]	[102 141 44]	[107 144 47]	[114 147 45]	[129 151 64]	[217 220 200]	[241 242 236]	[240 241 236]	[240 241 236]	[239 240 235]	[239 240 235]	[239 240 235]	[238 239 234]	[238 239 234]
28	[105 143 46]	[98 137 41]	[108 145 49]	[117 151 50]	[186 200 151]	[240 241 235]	[240 241 236]	[240 241 236]	[240 241 235]	[240 241 236]	[239 240 235]	[239 240 235]	[238 239 234]	[238 239 234]

Gambar 3. 10 Nilai *Pixel* Citra RGB

Dalam perhitungan RGB ke derajat keabuan, nilai pixel dari 28x28 diatas diambil sampel 8x8 dapat dilihat pada tabel 3.1.

Tabel 3. 1 Nilai *Pixel* Citra RGB

xy	1	2	3	…	8
1	R = 93 G = 131 B = 39	R = 97 G = 135 B = 41	R = 92 G = 121 B = 26	…	R = 94 G = 136 B = 25
2	R = 102 G = 139 B = 51	R = 108 G = 141 B = 54	R = 103 G = 119 B = 35	…	R = 101 G = 139 B = 27
3	R = 112 G = 148 B = 61	R = 116 G = 151 B = 66	R = 100 G = 137 B = 44	…	R = 108 G = 143 B = 29
…	…	…	…	…	…
8	R = 103 G = 140 B = 46	R = 105 G = 141 B = 44	R = 91 G = 131 B = 23	…	R = 105 G = 143 B = 32

Contoh perhitungan kita ambil pixel (1,1) berdasarkan persamaan (2.2).

$$(x, y) = 0.299.R + 0.587.G + 0.114.B$$

Maka dapat diilustrasikan sebagai berikut:

(1,1) = (0,2989 * R) + (0,5870 * G) + (0,1141 * B)

= (0,2989 * 93) + (0,5870 * 131) + (0,1141 * 39)

= 27,7977 + 76,897 + 4,4499

= 109,1446 ≈ 109,14

Maka derajat keabuan secara keseluruhan dapat dilihiat pada tabel 3.2.

Tabel 3. 2 Hasil Nilai *Pixel* Derajat Keabuan

xy	1	2	3	4	5	6	7	8
1	109,14	112,92	101,49	109,57	128,90	132,73	118,86	110,78
2	117,90	121,21	104,63	99,93	124,73	130,56	115,29	114,66
3	127,31	130,84	115,33	99,35	111,99	122,64	112,79	119,53
4	132,37	132,67	123,64	106,58	102,01	114,02	121,76	115,65
5	135,89	133,25	117,69	111,41	103,60	111,75	123,36	103,32
6	135,14	132,32	114,05	105,84	106,38	117,77	111,76	103,32
7	125,08	125,57	109,18	103,38	112,29	117,18	108,34	113,85
8	118,22	119,17	106,72	106,16	119,18	102,34	100,84	118,98

Sehingga dari hasil keseluruhan image derajat keabuan dapat dilihat pada gambar 3.11.

Gambar 3. 11 Citra Daun Derajat Keabuan

3.4.2 Deteksi Tepi (*Edge Detection Sobel*)

Dari hasil derajat keabuan maka dalam tahap ini derajat keabuan diubah ke dalam *edge detection* (*Sobel*).

Hasil derajat keabuan kemudian dihitung berdasarkan persamaan (2.3), (2.4) dan (2.5). dapat diilustrasikan sebagai berikut:

xy	1	2	3	4	5	6	7	8
1	109,14	112,92	101,49	109,57	128,90	132,73	118,86	110,78
2	117,90	121,21	104,63	99,93	124,73	130,56	115,29	114,66
3	127,31	130,84	115,33	99,35	111,99	122,64	112,79	119,53
4	132,37	132,67	123,64	106,58	102,01	114,02	121,76	115,65
5	135,89	133,25	117,69	111,41	103,60	111,75	123,36	103,32
6	135,14	132,32	114,05	105,84	106,38	117,77	111,76	103,32
7	125,08	125,57	109,18	103,38	112,29	117,18	108,34	113,85
8	118,22	119,17	106,72	106,16	119,18	102,34	100,84	118,98

Kernel Sobel:

$$x = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad y = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

- Rumus:

$$x = ((1,3) + 2(2,3) + (3,3)) - ((1,1) + 2(2,1) + (3,1))$$

$$y = \big((3,1) + 2(3,2) + (3,3)\big) - ((1,1) + 2(1,2) + (1,3))$$

- Gradient:

$$G = \sqrt{x^2 + y^2}$$

Keterangan:

G = Gradien Sobel

Contoh:

x = (127,3129+2*130,8400+115,3294) - (109,1446+2*112,9164+101,4924)

= 67,8525

y = (101,4924+2*104,6332+115,3294) - (109,1446+2*117,8999+127,3129)

= -46,1691

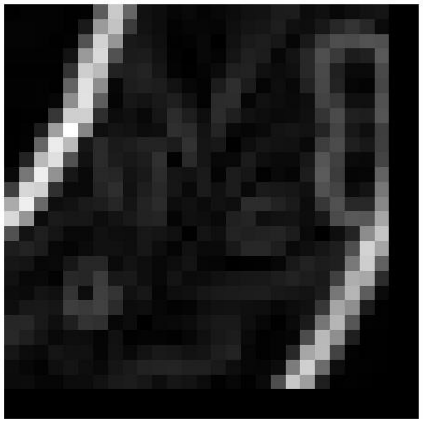
$G = \sqrt{67{,}8525^2 + -46{,}1691^2} = 82{,}0704 \approx 82{,}07$

Kernel sobel akan digeser 1 *stride* ke seluruh permukaan *pixel* derajat keabuan sehingga menghasilkan matriks deteksi tepi. Maka hasil keseluruhan deteksi tepi (*Sobel*) dapat dilihat pada tabel 3.3.

Tabel 3. 3 Hasil *Pixel* Deteksi Tepi

xy	1	2	3	4	5	6
1	82,07	85,09	68,43	107,70	67,54	58,03
2	72,53	123,79	12,61	101,13	50,75	20,49
3	50,19	107,24	63,32	41,57	63,18	10,36
4	66,66	98,38	57,87	23,25	64,68	41,34
5	83,81	102,39	30,74	40,79	26,95	43,08
6	81,87	88,26	12,60	37,23	35,73	22,12

Sehingga dari hasil keseluruhan image deteksi tepi dapat dilihat pada gambar 3.12.



Gambar 3. 12 Citra Daun Tepi Sobel

3.5 Analisis Metode Standarisasi

Proses ini untuk merubah atau menstandarkan tingkat intensitas cahaya pada citra, citra dipetakan pada pixel dengan ukuran tertentu sehingga memberikan representasi dimensi yang tetap. Sesuai dengan persamaan (2.7) dan (2.8) maka dapat diilustrasikan sebagai berikut:

$$s_j = \frac{x_j - \bar{x}}{\sigma}$$

Dimana:

x_j= data ke-j

$\bar{x}$ = rata – rata

σ = standar deviasi

Rata – rata:

$$\bar{x} = \frac{(1,1) + (2,1) + (3,1) + \cdots + (n,n)}{n}$$

Rumus untuk menghitung standar deviasi sebagai berikut:

$$\sigma = \sqrt{\frac{\sum_{j=1}^{n}(x_j - \bar{x})^2}{n-1}}$$

Keterangan:

σ : Standar Deviasi

x_j : nilai x ke-j

$\bar{x}$: Rata-rata

n : ukuran sampel

Maka dapat diilustrasikan sebagai berikut:

xy	1	2	3	4	5	6
1	82,07	85,09	68,43	107,70	67,54	58,03
2	72,53	123,79	12,61	101,13	50,75	20,49
3	50,19	107,24	63,32	41,57	63,18	10,36
4	66,66	98,38	57,87	23,25	64,68	41,34
5	83,81	102,39	30,74	40,79	26,95	43,08
6	81,87	88,26	12,60	37,23	35,73	22,12

1. Cari nilai rata -rata.

$$\bar{x} = \frac{82,07+85,09+68,43+107,70+67,54+58,03+72,53123,79+12,61+101,13+50,75+20,49+\cdots+22,12}{36}$$

$$= \frac{2143,7821}{36} = 59,5495 \approx 59,55$$

2. Cari nilai standar deviasi.

$$\sigma = \sqrt{\frac{\sum_{j=1}^{n}(x_j - \bar{x})^2}{n-1}}$$

Tabel 3. 4 Perhitungan Standar Deviasi

xy	x_j	$(x_j - \bar{x})$	$(x_j - \bar{x})^2$
(1,1)	82,0704	22,5208	507,1879
(2,1)	85,0864	25,5408	652,3341
(3,1)	68,4346	8,8808	78,8692
(4,1)	107,6990	48,1508	2318,50
⋮	⋮	⋮	⋮
(36,36)	22,1157	-37,4292	1400,9425
$\sum$			32755,6539
32755,6539 / 35			935,8758
$\sigma = \sqrt{935,8758}$			**30,5921**

3. Masukkan nilai yang sudah dihitung diatas ke rumus standarisasi.

$$s_j = \frac{x_j - \bar{x}}{\sigma}$$

$s_1 = \frac{82,0704-59,5495}{30,5921} = 0,7362$

$s_2 = \frac{85,0864-59,5495}{30,5921} = 0,8349$

$s_3 = \frac{68,4346-59,5495}{30,5921} = 0,2903$

$s_{36} = \frac{82,0704-59,5495}{30,5921} = -1,2235$

Maka hasil dari keseluruhan standarisasi dapat dilihat pada tabel 3.5.

Tabel 3. 5 Hasil *Pixel* Standarisasi

xy	1	2	3	4	5	6
1	0,74	0,83	0,29	1,57	0,26	-0,05
2	0,42	2,10	-1,53	1,36	-0,29	-1,28
3	-0,31	1,56	0,12	-0,59	0,12	-1,61
4	0,23	1,27	-0,05	-1,19	0,17	-0,60
5	0,79	1,40	-0,94	-0,61	-1,07	-0,54
6	0,73	0,94	-1,53	-0,73	-0,78	-1,22

3.6 CNN *Training*

CNN *Training* terdiri dari beberapa tahap, yaitu tahap inisialisasi, tahap *feedforward*, tahap *backpropagation*, dan tahap *update* bobot. Masukan data berupa citra hasil standarisasi dan keluaran adalah klasifikasi jenis hama.

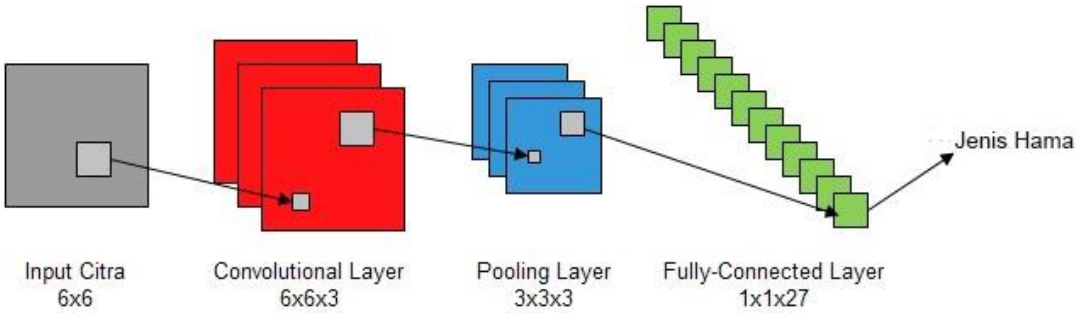
Data masukan yang digunakan dalam analisis ini adalah data citra hasil standarisasi jenis hama *Blister* pada tabel Tabel 3.6 yang telah di-*resize* sehingga menjadi berukuran 6*6 pixel. Berikut adalah matriks citranya.

Tabel 3. 6 Matriks Citra Masukan Hama *Blister*

xy	1	2	3	4	5	6
1	0,74	0,83	0,29	1,57	0,26	-0,05
2	0,42	2,10	-1,53	1,36	-0,29	-1,28
3	-0,31	1,56	0,12	-0,59	0,12	-1,61
4	0,23	1,27	-0,05	-1,19	0,17	-0,60
5	0,79	1,40	-0,94	-0,61	-1,07	-0,54
6	0,73	0,94	-1,53	-0,73	-0,78	-1,22

Dalam analisis ini akan diperiksa apakah citra masukan merupakan jenis hama *Blister* atau *Hellopeltis*. CNN yang digunakan memiliki 1 *convolutional layer* dengan 3 buah filter, 1 *pooling layer,* dan 1 *fully-connected layer*.

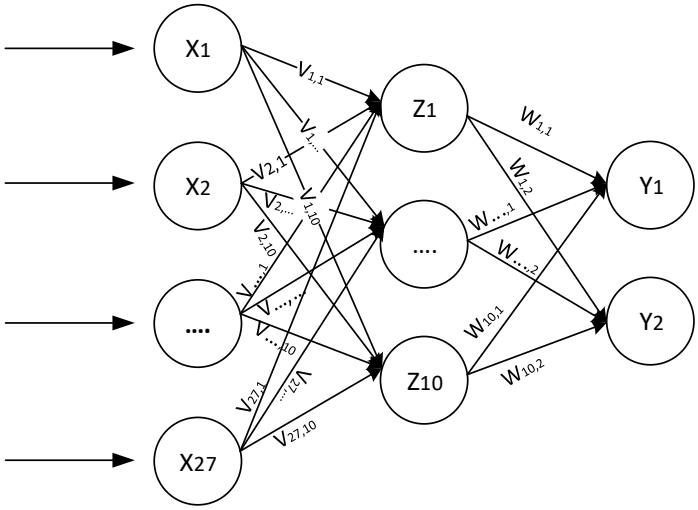
Dalam *convolutional layer* input citra A berukuran 6*6 pixel akan diberikan operasi konvolusi dengan 3 buah filter F berukuran 3*3 pixel yang akan menghasilkan *feature map* FM berukuran 6*6 pixel sebanyak 3 buah. Setelah itu dalam *pooling layer*, *feature map* FM akan diberikan operasi *max-pooling* dengan ukuran filter 2*2 pixel, yang akan menghasilkan *feature map* P dengan ukuran 3*3 pixel sebanyak 3 buah. Setelah itu ketiga *feature map* P akan dipisah menjadi matriks berukuran 1*1 sebanyak 27 buah, dengan kata lain direntangkan menjadi sebuah vektor X dengan jumlah baris 27 dan kolom 1. Vektor ini akan dimasukkan ke dalam *fully-connected layer* untuk memprediksi kelas huruf dari citra yang diinputkan. Berikut adalah arsitektur CNN yang digunakan dalam analisis ini.



Gambar 3. 13 Arsitektur CNN Analisis

Fully-connected layer adalah sebuah *neural network* yang terdiri dari 1 *input layer* dengan 27 *node*, 1 *hidden layer* dengan 10 *node*, dan 1 *output layer* dengan 3 *node*.

Vektor X yang dihasilkan dari *layer* sebelumnya akan menjadi *input layer*. Tiap *node input layer* X akan dikalikan dengan *weight* V dan hasilnya akan menjadi *node hidden layer* Z. Tiap *node hidden layer* Z akan dikalikan dengan *weight* W dan hasilnya akan menjadi hasil prediksi kelas huruf Y berbentuk matriks *one-hot*. Berikut adalah arsitektur *fully-connected layer* yang digunakan dalam analisis ini.



Gambar 3. 14 Arsitektur *Fully-Connected Layer CNN* Analisis

Keluaran yang dihasilkan adalah vektor *one-hot* jenis hama, yang berisi probabilitas citra masukan adalah sebuah jenis hama. [28] Karena jenis hama yang digunakan 2 buah (*Blister*, *Hellopeltis*), maka vektor *one-hot* yang digunakan memiliki 2 komponen. Jumlah *node output layer* Y pada *fully-connected layer* CNN juga berjumlah 2 buah, sehingga kita dapat memetakan nilai Y pada vektor *one-hot* kelas huruf, seperti pada tabel 3.7.

Tabel 3. 7 Contoh Vektor *One-Hot* Jenis Hama

Jenis Hama	Output Node
Blister	Y_1
Hellopeltis	Y_2

Maka nilai probabilitas citra masukan adalah jenis hama *blister* terdapat pada nilai Y_1, dan nilai probabilitas citra masukan adalah jenis hama *hellopeltis* terdapat pada nilai Y_2.

3.6.1 Inisialisasi CNN

CNN *Training* terdiri dari beberapa tahap, yaitu tahap inisialisasi bobot, tahap *feedforward*, tahap *backpropagation*, dan tahap *update* bobot. Masukan data hasil dari standarisasi dan keluaran adalah klasifikasi jenis hama. Inisialisasi CNN meliputi inisialisasi parameter pelatihan dan inisialisasi bobot.

1. Inisialisasi Parameter Pelatihan

   Parameter pelatihan adalah parameter-parameter yang menentukan kinerja *training* CNN. Parameter-parameter yang diinisialisasi adalah jumlah maksimum *epoch* (iterasi pelatihan), *learning rate* (laju pembelajaran), dan minimum *error* [25].

2. Inisialisasi Bobot

   Dalam analisis ini kelas bobot-bobot yang akan diinisialisasi adalah nilai-nilai awal bobot dan bias filter *convolutional layer* F, bobot dan bias *hidden layer* V, dan bobot dan bias *output layer fully-connected layer* W. Semua bobot diisi dengan nilai awal rentang nilai antara -0,5 dan 0,5 secara acak, sedangkan semua bias diisi dengan nilai awal 0.

   Filter *convolutional layer* F berjumlah 3 buah dengan masing-masing berukuran 3*3 pixel, maka jumlah bobot yang harus diisi adalah 27 buah. Berikut adalah matriks-matriks inisialisasi filternya.

Tabel 3. 8 Inisialisasi Matriks-Matriks Filter F

Filter F[1]			
x/y	**1**	**2**	**3**
1	0,42	0,49	0,01
2	0,48	-0,33	0,18
3	0,37	0,13	-0,34

Filter F[2]			
x/y	**1**	**2**	**3**
1	-0,36	-0,19	-0,29
2	0,29	0,44	-0,42
3	0,26	-0,23	-0,44

Filter F[3]			
x/y	**1**	**2**	**3**
1	-0,24	0,28	-0,46
2	0,15	0,09	-0,45
3	-0,5	0,22	0,5

Sedangkan bias-bias filter *convolutional layer* diisi dengan nilai awal 0.

- Bias F[1] (bF[1]) = 0
- Bias F[2] (bF[2]) = 0
- Bias F[3] (bF[3]) = 0

Matriks-matriks filter F akan digunakan pada proses konvolusi citra masukan pada *convolutional layer*.

Matriks inisialisasi nilai awal bobot-bobot dan bias *hidden layer* V dan *output layer* W dari *fully-connected layer* dapat dilihat pada tabel-tabel berikut. Terdapat 27 *node input layer* X dan 10 *node hidden layer* Z, sehingga jumlah bobot yang harus diisi untuk matriks bobot *hidden layer* V adalah 270 dan jumlah bias adalah 10. Berikut adalah matriks V. $V_{0,y}$ menandakan nilai bias (semua diberi nilai 0) dan $V_{x,y}$ menandakan nilai bobot.

Tabel 3. 9 Inisialisasi Matriks Bobot *Hidden Layer* V

x/y	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0
1	0,36	0,46	-0,43	-0,35	0,06	0,25	-0,16	-0,41	0,47	-0,19
2	-0,17	0,47	-0,48	0,31	0,37	-0,19	-0,33	0,24	-0,24	-0,36
3	-0,14	0,13	-0,49	-0,1	0,34	-0,46	0,26	-0,11	0,42	-0,46
4	0,25	-0,05	-0,22	-0,43	-0,12	0	-0,14	0,23	0,25	0,41
5	-0,21	0,04	0,01	-0,43	-0,17	-0,34	-0,17	-0,19	0,07	0,35
6	0,32	-0,19	-0,47	-0,36	0,42	-0,38	0,19	-0,24	-0,19	0,28
7	0,04	0,16	0,23	-0,21	-0,27	-0,32	-0,13	0,3	0,42	0,05
8	-0,44	-0,23	0,02	0,48	0,04	0,22	0,45	-0,38	0,2	-0,05
9	0,49	-0,08	-0,16	-0,14	0,47	0,19	0,5	0,29	-0,12	-0,34
10	0,22	-0,17	-0,14	0,46	0,4	-0,44	-0,07	-0,25	-0,14	0,05
11	-0,33	0,18	0,41	0,13	-0,23	-0,13	-0,18	0,16	0,46	0,15
12	-0,1	-0,04	-0,41	-0,06	0,2	-0,48	0,07	-0,13	0,4	-0,48
13	0,31	-0,37	-0,23	-0,48	-0,32	0,09	0,47	-0,02	-0,1	-0,05
14	-0,09	0,18	0,45	0,22	0,31	0,07	-0,37	-0,36	0,31	-0,13
15	0,09	0,37	0,16	-0,01	0,39	-0,28	0,21	-0,35	0,29	-0,31
16	-0,44	0,26	-0,19	-0,35	-0,43	0,22	-0,5	-0,22	-0,01	0,3
17	0,11	0,4	-0,04	-0,33	-0,11	0,19	0,45	-0,5	0,45	-0,33
18	-0,14	0,25	-0,42	-0,11	-0,26	-0,4	-0,09	0,44	0,34	-0,16
19	-0,13	-0,04	0	-0,13	0,19	0,39	-0,12	0,21	-0,01	-0,47
20	-0,15	-0,5	0,27	-0,43	-0,09	-0,09	0,31	-0,01	-0,27	-0,2
21	-0,24	0,05	0,01	0,14	-0,08	-0,45	0	-0,38	-0,41	0,41
22	0,06	-0,46	-0,19	0,42	-0,03	0,16	0,49	0,24	0,35	-0,06
23	-0,23	-0,26	-0,17	-0,12	0,23	0,33	0,11	-0,03	0,13	-0,07
24	-0,19	0,17	-0,46	0,43	-0,32	0,16	-0,26	0,5	0,26	-0,27
25	-0,3	0,38	-0,18	-0,29	0,49	0,23	-0,34	0,42	0,18	-0,19
26	0,08	0,26	0,1	-0,44	0,44	0,13	0,44	-0,47	0,03	-0,33
27	0,27	-0,09	-0,21	-0,23	0,1	-0,15	-0,44	0,36	-0,45	0,15

Matriks V akan digunakan sebagai bobot dari *input layer* ke *hidden layer* pada *fully-connected layer.*

Terdapat 10 *node hidden layer* Z dan 2 *node output layer* Y, sehingga jumlah bobot yang harus diisi untuk matriks W adalah 20 dan jumlah bias adalah 2.

Berikut adalah matriks W. $W_{0,y}$ menandakan nilai bias (semua diberi nilai 0) dan $V_{x,y}$ menandakan nilai bobot.

Tabel 3. 10 Matriks Inisialisasi *Hidden Layer* W

x/y	1	2
0	0	0
1	-0,13	0,35
2	0,42	0,28
3	0,31	0,24
4	0,39	-0,06
5	-0,09	-0,12
6	0,34	-0,35
7	-0,31	-0,21
8	-0,43	-0,26
9	0,13	-0,28
10	0,24	-0,09

Matriks V akan digunakan sebagai bobot dari *hidden layer* ke *output layer* pada *fully-connected layer.*

3.6.2 *Feedforward*

Pada tahap *feedforward* akan dilakukan proses CNN dari awal masukan melewati *convolutional, pooling,* dan *fully-connected layer* hingga dihasilkan sebuah vektor *one-hot* klasifikasi jenis hama.

1. *Convolutional Layer*

   Pada *convolutional layer* akan dilakukan konvolusi antara matriks citra masukan dengan matriks-matriks filter F. Filter-filter ini akan digeser ke seluruh permukaan gambar sehingga menghasilkan keluaran matriks *feature map* FM. Terdapat beberapa *hyperparameter* dalam *convolutional layer*, yaitu jumlah filter, *stride,* dan *zero-padding.* [29]

   1. Jumlah filter F berjumlah sebanyak 3 buah dengan ukuran 3*3 pixel.
   2. *Stride* adalah seberapa jauh jarak pergeseran filter setiap perkalian. Jika *stride* adalah 2, maka filter akan bergeser 2 pixel ke kanan lalu ke bawah. Dalam analisis ini *stride* yang digunakan adalah 1.

3. *Zero-padding* adalah jumlah lapisan pixel berisi nilai 0 yang ditambahkan ke setiap sisi matriks masukan. Dalam analisis ini *zero-padding* yang digunakan adalah 1.

Dengan *hyperparameter* di atas maka akan dihasilkan *feature map* FM dengan ukuran yang didapat dari rumus berikut [25]:

featurezise = ukuran *feature map*

inputsize (ukuran matriks masukan) = 6

filtersize (ukuran filter) = 3

pad (*zero padding*) = 1

str (*stride*) = 1

$$featuresize = \frac{inputsize - filtersize + 2pad}{str} + 1$$
$$= \frac{6-3+2*1}{1} + 1$$
$$= 6$$

Maka *feature map* FM yang dihasilkan berukuran 6*6 pixel.

Langkah-langkah yang dilakukan pada *convolutional layer* adalah sebagai berikut. Pertama-tama di setiap sisi matriks citra masukan akan ditambah pixel berisi nilai 0 sesuai dengan nilai *zero-padding*, sebagai berikut.

Tabel 3. 11 Matriks A (Matriks Citra Masukan Dengan *Zero-Padding*)

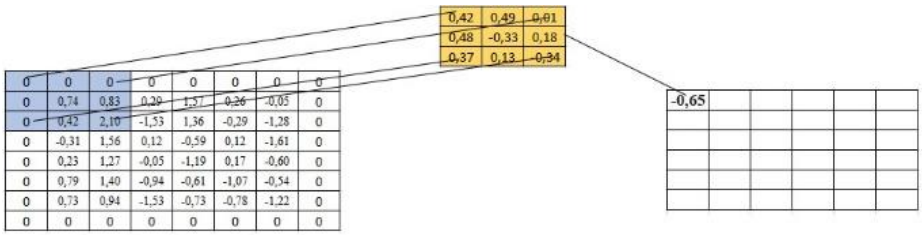
x/y	1	2	3	4	5	6	7	8
1	0	0	0	0	0	0	0	0
2	0	0,74	0,83	0,29	1,57	0,26	-0,05	0
3	0	0,42	2,10	-1,53	1,36	-0,29	-1,28	0
4	0	-0,31	1,56	0,12	-0,59	0,12	-1,61	0
5	0	0,23	1,27	-0,05	-1,19	0,17	-0,60	0
6	0	0,79	1,40	-0,94	-0,61	-1,07	-0,54	0
7	0	0,73	0,94	-1,53	-0,73	-0,78	-1,22	0
8	0	0	0	0	0	0	0	0

Setelah itu lakukan operasi konvolusi antara citra masukan dengan filter F, dimulai dari pojok kiri atas, lalu bergeser ke kanan lalu ke bawah. Tambah setiap hasil dengan bias filter bF. Lakukan untuk setiap filter F. Filter-filter F yang digunakan dapat dilihat pada Tabel 3.12.

Sebagai contoh, akan dilakukan operasi konvolusi [26] antara citra masukan A dengan *zero-padding* dengan filter F[1]. Posisikan filter F[1] di atas citra masukan A. Lalu kalikan tiap pixel A dengan pixel filter F[1] yang sesuai. Setelah itu jumlahkan semua hasilnya, dan tambah dengan bias bF[1] untuk menjadi pixel pertama pada *feature map* FM[1]. Berikut adalah perhitungannya berdasarkan persamaan (2.9).

$$\begin{aligned}\mathrm{FM}[1]_{1,1} &= (\mathrm{A}_{1,1} * \mathrm{F}[1]_{1,1} + \mathrm{A}_{1,2} * \mathrm{F}[1]_{1,2} + \mathrm{A}_{1,3} * \mathrm{F}[1]_{1,3} + \mathrm{A}_{2,1} * \mathrm{F}[1]_{2,1} \\ &\quad + \mathrm{A}_{2,2} * \mathrm{F}[1]_{2,2} + \mathrm{A}_{2,3} * \mathrm{F}[1]_{2,3} + \mathrm{A}_{3,1} * \mathrm{F}[1]_{3,1} + \mathrm{A}_{3,2} \\ &\quad * \mathrm{F}[1]_{3,2} + \mathrm{A}_{3,3} * \mathrm{F}[1]_{3,3}) + \mathrm{bF}[1] \\ &= (0 * 0.42 + 0 * 0.49 + 0 * 0.01 + 0 * 0.48 + 0.74 * (-0.3) + \\ &\quad 0.83 * 0.18 + 0 * 0.37 + 0.42 * 0.13 + 2.10 * (-0.3)) + 0 \\ &= \mathbf{-0.65}\end{aligned}$$

Berikut adalah visualiasi operasi konvolusi filter F[1] dengan matriks masukan A untuk pixel $\mathrm{FM}[1]_{1,1}$.

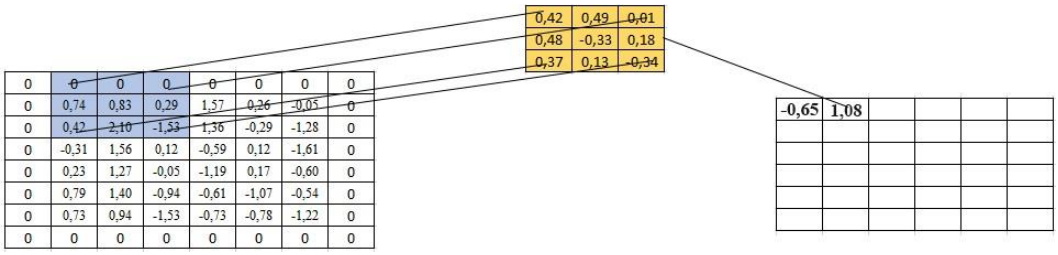


Gambar 3. 15 Contoh 1 Operasi Dot Citra Input dan Filter F[1]

Setelah itu geser filter ke kanan sebanyak 1 pixel sesuai *stride*, dan lakukan kembali operasi konvolusi.

$$\begin{aligned}\mathrm{FM}[1]_{1,2} &= (\mathrm{A}_{1,2} * \mathrm{F}[1] + \mathrm{A}_{1,3} * \mathrm{F}[1]_{1,2} + \mathrm{A}_{1,4} * \mathrm{F}[1]_{1,3} + \mathrm{A}_{2,2} * \mathrm{F}[1]_{2,1} \\ &\quad + \mathrm{A}_{2,3} * \mathrm{F}[1]_{2,2} + \mathrm{A}_{2,4} * \mathrm{F}[1]_{2,3} + \mathrm{A}_{3,2} * \mathrm{F}[1]_{3,1} + \mathrm{A}_{3,3} \\ &\quad * \mathrm{F}[1]_{3,2} + \mathrm{A}_{3,4} * \mathrm{F}[1]_{3,3}) + \mathrm{bF}[1] \\ &= (0 * 0.42 + 0 * 0.49 + 0 * 0.01 + 0.74 * 0.48 + 0.83 * -0.3 + \\ &\quad 0.29 * 0.18 + 0.42 * 0.37 + 2.10 * 0.13 + (-1.53) * (-0.3)) + \\ &\quad 0 \\ &= \mathbf{1.08}\end{aligned}$$

Berikut adalah visualisasi operasi konvolusi filter F[1] dengan matriks masukan A untuk pixel $FM[1]_{1,2}$.



Gambar 3. 16 Contoh 2 Operaso Dot Citra Input dan Filter F[1]

Lakukan pergeseran hingga ke seluruh bagian citra, hingga didapat hasil *feature map* sebagai berikut.

Tabel 3. 12 *Feature Map* FM[1]

x/y	1	2	3	4	5	6
1	-0,65	1,08	0,70	-0,62	1,56	-0,13
2	0,04	0,00	3,06	-0,41	1,65	0,20
3	0,21	0,82	1,62	0,06	-0,41	-0,18
4	-0,36	1,11	1,72	0,10	-1,12	-0,93
5	-0,11	1,38	1,77	-1,44	-0,42	-1,01
6	0,33	0,77	0,95	-1,34	-1,10	-0,69

Ulangi untuk setiap filter, sehingga menghasilkan 3 *feature map* FM. Hasil *feature map* FM[2] untuk filter F[2] dan *feature map* FM[3] untuk F[3] dapat dilihat pada tabel 3.13 dan 3.14.

Tabel 3. 13 *Feature Map* FM[2]

x/y	1	2	3	4	5	6
1	-1,04	0,76	0,01	0,08	1,57	0,27
2	-1,69	0,69	-0,81	-0,09	0,73	-0,33
3	-2,09	0,23	0,76	0,29	0,41	-0,14
4	-1,62	0,72	1,28	-0,21	0,96	-0,11
5	-1,23	1,58	1,06	0,22	0,68	-0,42
6	-0,63	0,99	-0,24	0,33	0,54	-0,28

Tabel 3. 14 *Feature Map* FM[3]

x/y	1	2	3	4	5	6
1	0,84	-0,46	-1,26	0,99	-1,10	-0,10
2	-0,37	1,42	-2,32	0,14	-0,01	-0,65
3	-0,89	1,37	-2,29	0,64	1,16	-0,63
4	-0,48	0,07	-0,56	-0,64	0,82	-0,09
5	-0,45	0,07	-0,54	0,10	0,25	-0,30
6	-0,78	1,52	0,01	0,60	0,46	-0,12

Lalu jalankan fungsi aktivasi pada tiap *feature map* FM. Setiap pixel *feature map* FM akan dimasukkan ke dalam fungsi aktivasi ReLU, dimana setiap pixel yang memiliki nilai < 0 akan dijadikan 0, dengan rumus $ReLU(x) = \max(0, x)$. *Feature map* FM yang telah dimasukkan ke dalam fungsi aktivasi ReLU akan menghasilkan *feature map* R. Ulangi untuk tiap *feature map* FM, sehingga akan menghasilkan tiga *feature map* R. Berikut adalah *feature map* R[1], R[2], dan R[3], dengan *font* tebal menandakan bahwa pixel tersebut awalnya adalah < 0 pada *feature map* FM.

Tabel 3. 15 *Feature Map* ReLU R[1]

x/y	1	2	3	4	5	6
1	**0**	1,08	0,70	**0**	1,56	**0**
2	0,04	0,00	3,06	**0**	1,65	0,20
3	0,21	0,82	1,62	0,06	**0**	**0**
4	**0**	1,11	1,72	0,10	**0**	**0**
5	**0**	1,38	1,77	**0**	**0**	**0**
6	0,33	0,77	0,95	**0**	**0**	**0**

Tabel 3. 16 *Feature Map* ReLU[2]

x/y	1	2	3	4	5	6
1	**0**	0,76	0,01	0,08	1,57	0,27
2	**0**	0,69	**0**	**0**	0,73	**0**
3	**0**	0,23	0,76	0,29	0,41	**0**
4	**0**	0,72	1,28	**0**	0,96	**0**
5	**0**	1,58	1,06	0,22	0,68	**0**
6	**0**	0,99	**0**	0,33	0,54	**0**

Tabel 3. 17 *Feature Map* ReLU[3]

x/y	1	2	3	4	5	6
1	0,84	**0**	**0**	0,99	**0**	**0**
2	**0**	1,42	**0**	0,14	**0**	**0**
3	**0**	1,37	**0**	0,64	1,16	**0**
4	**0**	0,07	**0**	**0**	0,82	**0**
5	**0**	0,07	**0**	0,10	0,25	**0**
6	**0**	1,52	0,01	0,60	0,46	**0**

Feature map R[1], R[2] dan R[3] akan digunakan pada tahap selanjutnya yaitu *pooling layer*.

2. *Pooling Layer*

   Pooling layer bertujuan untuk mengecilkan ukuran *feature map* R. Dalam penelitian ini jenis *pooling* yang digunakan adalah *max pooling*, yaitu memilih nilai maksimum dalam suatu jendela. Pemilihan ini akan diulangi dengan menggeser jendela ke seluruh permukaan citra sehingga menghasilkan keluaran matriks *feature map* P yang berisi nilai-nilai maksimum yang terpilih. Lakukan untuk ketiga *feature map* R, hingga menghasilkan *feature map* P[1], P[2], dan P[3].

   Terdapat beberapa *hyperparameter* dalam *pooling layer*, yaitu ukuran jendela dan *stride.* Dalam *pooling layer* ini ukuran jendela yang digunakan adalah 2*2 pixel dengan *stride* adalah 2. Dengan *hyperparameter* di atas maka akan dihasilkan *feature map* P dengan ukuran yang didapat dari rumus berikut: [25]

   featurezise = ukuran *feature map*

   inputsize = ukuran matriks masukan = 6

   windowsize = ukuran jendela = 2

   str = *stride* = 2

$$featuresize = \frac{inputsize - windowsize}{str} + 1$$

$$= \frac{6-2}{2} + 1$$

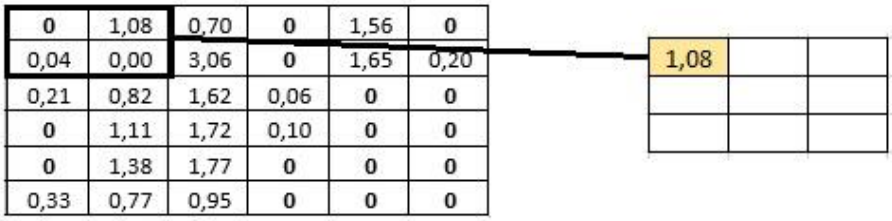
$$= 3$$

   Maka *feature map* P yang dihasilkan berukuran 3*3 pixel.

Sebagai contoh, akan dilakukan operasi *max pooling* pada *feature map* R[1] yang akan menghasilkan feature map P[1]. Letakkan jendela pada pojok kiri atas *feature map* R[1]. Berarti *pixel* yang diperiksa adalah $R[1]_{1,1}, R[1]_{1,2}, R[1]_{2,1}, R[1]_{2,2}$. Berikut adalah perhitungan operasi *max pooling* untuk $P[1]_{1,1}$ (2.10).

$$P[1]_{1,1} = \max(R[1]_{1,1}, R[1]_{1,2}, R[1]_{2,1}, R[1]_{2,2})$$
$$= \max(0,1.08,0.04,0) = 1.08$$

Dari keempat pixel tersebut, yang memiliki nilai tertinggi adalah $R[1]_{2,2} = 1.08$. Sehingga nilai pixel $P[1]_{1,1}$ adalah 1,08. Berikut adalah visualisasi operasi *max pooling* pada *feature map* R[1].

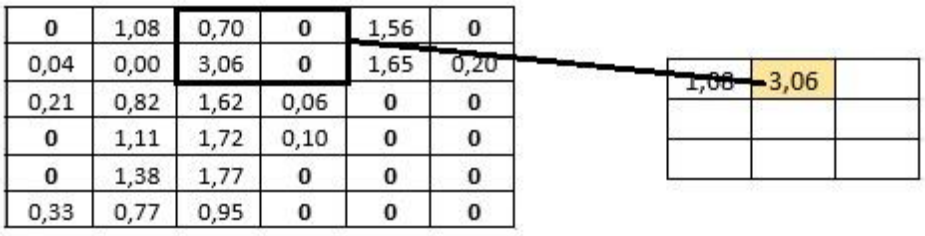


Gambar 3. 17 Contoh 1 Operasi *Max Pooling* P[1]

Setelah itu geser filter ke kanan sebanyak 2 pixel sesuai *stride*, dan lakukan kembali operasi *max pooling*.

$$P[1]_{1,2} = \max(R[1]_{1,3}, R[1]_{1,4}, R[1]_{2,3}, R[1]_{2,4})$$
$$= \max(0.70,0,3.06,0)$$
$$= 3.06$$

Dari $R[1]_{1,3}, R[1]_{1,4}, R[1]_{2,3}, R[1]_{2,4}$, yang memiliki nilai tertinggi adalah $R[1]_{2,4} = 3.06$. Sehingga nilai pixel $P[1]_{1,2}$ adalah 3.06.



Gambar 3. 18 Contoh 2 Operasi *Max Pooling* P[1]

Lakukan pergeseran hingga ke seluruh bagian citra, hingga didapat hasil *feature map* P[1] sebagai berikut.

Tabel 3. 18 *Feature Map* P[1]

x/y	1	2	3
1	1.08	3.06	1.65
2	1.11	1.72	0
3	1.38	1.77	0

Ulangi untuk setiap *feature map*. Hasil *feature map* P[2] untuk *feature map* R[2] dan hasil *feature map* P[3] untuk *feature map* R[3] dapat dilihat pada tabel 3.19 dan 3.20.

Tabel 3. 19 *Feature Map* P[2]

x/y	1	2	3
1	0.76	0.08	1.57
2	0.72	1.28	0.96
3	1.58	1.06	0.68

Tabel 3. 20 *Feature Map* P[3]

x/y	1	2	3
1	0.84	0.99	0
2	1.37	0.64	1.16
3	1.52	0.60	0.46

Feature map P[1], P[2] dan P[3] akan digunakan pada tahap selanjutnya yaitu *fully-connected layer*.

3. *Fully-Connected Layer*

Dalam CNN *fully-connected layer* adalah sebuah *neural network* yang memiliki *input layer, hidden layer* dan *output layer.*

3.1 Input Layer

Input layer X adalah penggabungan matriks *feature map* P[1], P[2] dan P[3] yang didapat dari *pooling layer* dan direntangkan menjadi sebuah vektor sepanjang jumlah pixel keseluruhan ketiga *feature map.* Total pixel *feature map* P[1], P[2], P[3] adalah 27, sehingga *input layer* X memiliki 27 *node*. Berikut adalah *input layer fully-connected layer.*

Tabel 3. 21 *Input Layer* X

x_i	Nilai	x_i	Nilai	x_i	Nilai
x_1	1.08	x_{10}	0.76	x_{19}	0.84
x_2	3.06	x_{11}	0.08	x_{20}	0.99
x_3	1.65	x_{12}	1.57	x_{21}	0
x_4	1.11	x_{13}	0.72	x_{22}	1.37
x_5	1.72	x_{14}	1.28	x_{23}	0.64
x_6	0	x_{15}	0.96	x_{24}	1.16
x_7	1.38	x_{16}	1.58	x_{25}	1.52
x_8	1.77	x_{17}	1.06	x_{26}	0.60
x_9	0	x_{18}	0.68	x_{27}	0.46

Nilai-nilai *input layer* X akan digunakan pada perhitungan pada *hidden layer* Z.

3.2 Hidden Layer

Setiap *node* di *input layer* X akan mengirimkan sinyal input kepada setiap *hidden layer* Z. Setiap *node* X_i akan dikalikan dengan bobot $V_{j,i}$ lalu ditambahkan dengan bias $V_{0,i}$ untuk menghasilkan nilai masukan z_in_i. Nilai-nilai bobot V dapat dilihat pada Tabel 3.21. Contoh penghitungan masukan *hidden layer* z_in adalah sebagai berikut berdasarkan persamaan (2.17).

$$z\_in_i = \sum_{i=1}^{n} X_j * V_{j,i} + V_{0,i}$$

z_in_i = masukan untuk *node hidden layer Z* ke-*i* dengan jumlah node *n*

X_j = *node X* ke-*j*

$V_{j,i}$ = *weight V* untuk *node Xj* dan *node Zi*

$V_{o,i}$ = bias *V* untuk *node* z_in_i

$$z\_in_1 = \sum_{i=1}^{10} X_j * V_{j,1} + V_{0,1}$$

$$= (X_{1,1} * V_{1,1} + X_{1,2} * V_{2,1} + X_{1,3} * V_{3,1} + \cdots + X_{1,27} * V_{27,1}) + V_{0,1}$$

$$= (1.08 * 0.36 + 3.06 * -0.2 + 1.65 * -0.1 + \cdots + 0.46 * 0.27) + 0$$

$$= \mathbf{-2.4922}$$

Hidden layer Z memiliki 10 *node*, sehingga akan dihasilkan 10 nilai z_in. Hasil penghitungan z_in yang lainnya dapat dilihat pada Tabel 3.22.

Tabel 3. 22 Matriks z_in

z_in_i	**Nilai**
z_in_1	-2.4922
z_in_2	2.6871
z_in_3	-4.3430
z_in_4	-1.3850
z_in_5	2.0584
z_in_6	-1.4301
z_in_7	-0.6588
z_in_8	-0.4128
z_in_9	4.3249
z_in_{10}	-3.6672

Setelah itu hitung nilai keluaran Z dengan mengaktifkan nilai masukan z_in menggunakan fungsi aktivasi ReLU, dengan rumus $ReLU(x) = \max(0, x)$ (2.19).

$$Z_i = \max(0, z\_in_i)$$

Z_i = keluaran untuk *node hidden layer* ke-i

z_in_i = masukan untuk *node hidden layer* ke-i

$$Z_1 = \max\left(0, z_{in_1}\right)$$

$= \max(0, -2.4922)$

$= \mathbf{0}$

Hasil penghitungan Z yang lainnya dapat dilihat pada Tabel 3.23.

Tabel 3. 23 Matriks Z

Z_i	Nilai
Z_1	0
Z_2	2.6871
Z_3	0
Z_4	0
Z_5	2.0584
Z_6	0
Z_7	0
Z_8	0
Z_9	4.3249
Z_{10}	0

Nilai-nilai matriks *hidden layer* Z akan digunakan pada perhitungan *output layer* Y.

3.3 Output Layer

Setiap *node* di *output layer* Z_j akan mengirimkan sinyal input kepada setiap *output layer* Y_i. Setiap *node* Z akan dikalikan dengan bobot $W_{j,i}$ lalu ditambahkan dengan bias $W_{0,i}$. Nilai-nilai bobot W dapat dilihat pada Tabel 3.11. Penghitungan masukan *output layer* y_in adalah sebagai berikut berdasarkan persamaan (2.18).

$$y\_in_i = \sum_{i=1}^{m} Z_j * W_{j,i} + W_{0,i}$$

y_in_i = masukan untuk *node hidden layer Z* ke-*i* dengan jumlah node *m*

Z_j = *node Z* ke-*j*

$W_{j,i}$ = *weight W* untuk *node Zj* dan *node Yi*

$W_{o,i}$ = bias W untuk *node* y_in_i

$$y\_in_i = \sum_{j=1}^{m} Z_j * W_{j,i} + W_{0,i}$$

$y\_in_1 = \sum_{j=1}^{3} Z_{1,j} * W_{j,1} + W_{0,1}$

$= (Z_{1,1} * W_{1,1} + Z_{1,2} * W_{2,1} + Z_{1,3} * W_{3,1} + \cdots + Z_{1,10} * W_{10,1}) + W_{0,1}$

$= (0 * -0.1 + 2.6871 * 0.42 + 0 * 0.31 + 0 * 0.39 + \cdots + 0 * 0.24) + 0$

$= 1.5056$

Output layer Y memiliki 10 *node*, sehingga akan dihasilkan 3 nilai y_in. Hasil penghitungan y_in yang lainnya dapat dilihat pada Tabel 3.24.

Tabel 3. 24 Matriks y_in

y_in_i	**Nilai**
y_in_1	1.5056
y_in_2	-0.7056

Setelah itu aktifkan nilai keluaran Y dengan menggunakan fungsi aktivasi *softmax*, dengan rumus $softmax(x)_i = \frac{e^{x_i}}{\sum_{i=1}^{m} e^{M}}$. berdasarkan persamaan (2.20).

$$Y_i = \frac{e^{y\_in_i}}{\sum_{i=1}^{m} e^{M}}$$

Y_i = keluaran untuk *node output layer* ke-*i*

y_in_i = masukan untuk *node output layer* ke-*i*

M = semua masukan untuk *node output layer*, berjumlah *m* buah

$$Y_1 = \frac{e^{y\_in_i}}{\sum_{i=1}^{m} e^{M}} = \frac{e^{1.5056}}{e^{1.5056} + e^{-0.7056}} = \mathbf{0,901251}$$

Hasil penghitungan Y yang lainnya dapat dilihat pada Tabel 3.25.

Tabel 3. 25 Matriks Y

Y_i	Nilai
Y_1	0.901251
Y_2	0.098749

Dari hasil keluaran Y dapat dipetakan klasifikasi kelas huruf sebagai berikut.

Tabel 3. 26 Matriks Y

Jenis Hama	Tingkat Keyakinan
Y_1 = Blister	90%
Y_2 = Hellopeltis	10%

3.6.3 *Backpropagation*

Pada tahap *backpropagation* [14] akan dilakukan pelatihan pada jenis hama *Blister*. Pada tahap ini akan dilakukan proses penyesuaian tiap *weight* dan *bias* berdasarkan *error* yang didapat pada tahap *feedforward.* Pertama akan dihitung gradien dari *loss function* terhadap semua parameter (*weight* dan *bias*) yang ada dengan mencari turunan parsial (*partial derivative*) dari fungsi tersebut. Setelah itu *update* semua parameter dengan menggunakan *Stochastic Gradient Descent (SGD).* [27]

1. Penghitungan *loss function*

   Loss function adalah sebuah fungsi yang mengukur seberapa bagus performa dari *neural network* dalam melakukan prediksi terhadap target. *Loss function* akan menghitung *loss*, yaitu selisih dari nilai *output* dengan nilai target yang diharapkan. Untuk masalah klasifikasi, *loss function* yang biasa digunakan adalah *cross-entropy loss function* berdasarkan persamaan (2.11).

$$\mathrm{L} = -\sum_{i}^{m} t_i \log(Y_i)$$

   dimana L adalah *loss*, t adalah vektor *one-hot* target yang diharapkan, dan Y adalah nilai matriks output pada tahap *feedforward* dengan jumlah kelas *m*

buah. Nilai matriks Y diambil dari Tabel 3.26, Sedangkan karena jenis hama citra yang diharapkan adalah *Blister*, nilai vektor t adalah sebagai berikut.

Tabel 3. 27 Vektor t

t_i	Nilai
t_1 (Blister)	1
t_2 (Hellopeltis)	0

Vektor t di atas memaksimalkan probabilitas jenis hama *Blister* (1 atau 100%) dan meminimalkan probabilitas jenis hama *Hellopeltis* (0 atau 0%).

Berikut perhitungan *cross-entropy loss function*.

$$L = -\sum_i^m t_i \log(Y_i)$$

$$L = -(t_1 \log(Y_1) + t_2 \log(Y_2) + t_3 \log(Y_3))$$

$$L = -(1 * \log(0.901251) + 1 * \log(0.098749))$$

$$L = 1.0506$$

2. Penghitungan gradien kesalahan terhadap parameter bobot W_{ji}

Setelah itu hitung gradien kesalahan terhadap parameter bobot W_{ji}, dengan menggunakan rumus *chain rule* berdasarkan persamaan (2.12). [28]

$$\frac{\partial L}{\partial W_{ji}} = (Y_i - t_i)Z_j$$

Perhitungan gradien kesalahan terhadap parameter bobot W_{11} adalah sebagai berikut.

$$\frac{\partial L}{\partial W_{1,1}} = \frac{\partial L}{\partial Y_1}\frac{\partial Y_1}{\partial y\_in_1}\frac{\partial y\_in_1}{\partial W_{1,1}}$$

$$\frac{\partial L}{\partial W_{1,1}} = (Y_i - t_i)Z_j$$

$$\frac{\partial L}{\partial W_{1,1}} = (0.901251 - 1) * 0$$

$$\frac{\partial L}{\partial W_{1,1}} = 0$$

Hasil penghitungan Y yang lainnya dapat dilihat pada Tabel 3.28.

Tabel 3. 28 Matriks Gradien $\frac{\partial L}{\partial W_{ji}}$

j/i	1	2
1	0	0
2	0.2391	0.2391
3	0	0
4	0	0
5	0.1832	0.1832
6	0	0
7	0	0
8	0	0
9	0.3849	0.3849
10	0	0

3. Penghitungan gradien kesalahan terhadap parameter bobot V_{kj}

   Setelah itu hitung gradien kesalahan terhadap parameter bobot V_{kj}, dengan menggunakan rumus *chain rule* berdasarkan persamaan (2.13).

$$\frac{\partial L}{\partial V_{kj}} = \sum_{i}^{m} (Y_i - t_i)(W_{ji})(Z_j(1 - Z_j))(X_k)$$

   Dengan m adalah jumlah *node output layer* Y.

   Perhitungan gradien kesalahan terhadap parameter bobot V_{11} adalah sebagai berikut.

$$\frac{\partial L}{\partial V_{1,1}} = \sum_{i=1}^{2} (Y_i - t_i)(W_{1,i})(0(1 - 0))(1{,}08)$$

$$\frac{\partial L}{\partial V_{1,1}} = \sum_{i=1}^{2} (Y_i - t_i)(W_{1,i})(0)$$

$$\frac{\partial L}{\partial V_{1,1}} = (Y_1 - t_1)(W_{1,1})(0) + (Y_2 - t_2)(W_{1,2})(0)$$

$$\frac{\partial L}{\partial V_{1,1}} = (0.901251 - 1)(-0{,}13)(0) + (0.098749 - 0)(0{,}35)(0)$$

$$\frac{\partial L}{\partial V_{1,1}} = 0$$

   Hasil penghitungan Y yang lainnya dapat dilihat pada Tabel 3.29.

Tabel 3. 29 Matriks Gradien $\frac{\partial L}{\partial V_{kj}}$

k/j	1	2	3	4	5	6	7	8	9	10
1	0	0,534050073	0	0	0,437965956	0	0	0	-5,060743626	0
2	0	1,513141873	0	0	1,240903543	0	0	0	-14,33877361	0
3	0	0,815909833	0	0	0,669114655	0	0	0	-7,731691651	0
4	0	0,548884797	0	0	0,450131677	0	0	0	-5,201319838	0
5	0	0,85052419	0	0	0,697501338	0	0	0	-8,059702812	0
6	0	0	0	0	0	0	0	0	0	0
7	0	0,682397315	0	0	0,559623166	0	0	0	-6,466505744	0
8	0	0,87524873	0	0	0,717777539	0	0	0	-8,293996498	0
9	0	0	0	0	0	0	0	0	0	0
10	0	0,375813014	0	0	0,308198266	0	0	0	-3,561264033	0
11	0	0,039559265	0	0	0,032441923	0	0	0	-0,374869898	0
12	0	0,776350569	0	0	0,636672733	0	0	0	-7,356821753	0
13	0	0,356033382	0	0	0,291977304	0	0	0	-3,373829084	0
14	0	0,632948234	0	0	0,519070763	0	0	0	-5,997918372	0
15	0	0,474711176	0	0	0,389303072	0	0	0	-4,498438779	0
16	0	0,781295477	0	0	0,640727973	0	0	0	-7,40368049	0
17	0	0,524160257	0	0	0,429855476	0	0	0	-4,967026151	0
18	0	0,336253749	0	0	0,275756343	0	0	0	-3,186394135	0
19	0	0,415372279	0	0	0,340640188	0	0	0	-3,936133931	0
20	0	0,4895459	0	0	0,401468793	0	0	0	-4,63901499	0
21	0	0	0	0	0	0	0	0	0	0
22	0	0,677452407	0	0	0,555567926	0	0	0	-6,419647007	0
23	0	0,316474117	0	0	0,259535381	0	0	0	-2,998959186	0
24	0	0,573609337	0	0	0,470407879	0	0	0	-5,435613524	0
25	0	0,751626028	0	0	0,616396531	0	0	0	-7,122528066	0
26	0	0,296694485	0	0	0,24331442	0	0	0	-2,811524237	0
27	0	0,227465772	0	0	0,186541055	0	0	0	-2,155501915	0

4. *Update* nilai parameter

   Setelah semua gradien parameter dihitung, *update* parameter-parameter dengan menggunakan *Stochastic Gradient Descent*. Pada tahap ini akan digunakan

learning rate α=0,01. Berikut adalah perhitungan untuk *update* nilai parameter bobot W_{11} berdasarkan persamaan (2.14).

$$W_{1,1}' = W_{1,1} - \alpha(\frac{\partial L}{\partial W_{1,1}})$$

$$W_{1,1}' = -0{,}13 - 0.01(0)$$

$$W_{1,1}' = -0{,}13$$

Hasil penghitungan W_{ji}' yang lainnya dapat dilihat pada Tabel 3.43.

Tabel 3. 30 Matriks W_{ji}'

j/i	1	2
1	-0,13	0,35
2	0,48502335	0,257730643
3	0,31	0,24
4	0,39	-0,06
5	0,002542248	-0,151694097
6	0,34	-0,35
7	-0,30399558	-0,212056409
8	-0,43	-0,26
9	0,220818908	-0,311103883
10	0,24	-0,09

Selanjutnya berikut adalah perhitungan untuk *update* nilai parameter bobot V_{11} berdasarkan persamaan (2.16).

$$V_{1,1}' = V_{1,1} - \alpha(\frac{\partial L}{\partial V_{1,1}})$$

$$V_{1,1}' = 0{,}36 - 0.03(0)$$

$$V_{1,1}' = 0.36$$

Hasil penghitungan $V_{kj}{}'$ yang lainnya dapat dilihat pada Tabel 3.31

Tabel 3. 31 Matriks $V_{kj}{}'$

k/j	1	2	3	4	5	6	7	8	9	10
1	0,36	0,45323095	-0,43	-0,35	0,059302931	0,25	-0,16	-0,41	0,407119849	-0,19
2	-0,17	0,450821024	-0,48	0,31	0,36802497	-0,19	-0,33	0,24	-0,418160428	-0,36
3	-0,14	0,119658395	-0,49	-0,1	0,338935033	-0,46	0,26	-0,11	0,323933102	-0,46
4	0,25	-0,05695708	-0,22	-0,43	-0,12071643	0	-0,14	0,23	0,185373178	0,41
5	-0,21	0,02921966	0,01	-0,43	-0,17111015	-0,34	-0,17	-0,19	-0,030142463	0,35
6	0,32	-0,19	-0,47	-0,36	0,42	-0,38	0,19	-0,24	-0,19	0,28
7	0,04	0,151350658	0,23	-0,21	-0,2708907	-0,32	-0,13	0,3	0,33965314	0,05
8	-0,44	-0,24109372	0,02	0,48	0,038857581	0,22	0,45	-0,38	0,096946419	-0,05
9	0,49	-0,08	-0,16	-0,14	0,47	0,19	0,5	0,29	-0,12	-0,34
10	0,22	-0,17476341	-0,14	0,46	0,39950947	-0,44	-0,07	-0,25	-0,184248995	0,05
11	-0,33	0,179498589	0,41	0,13	-0,23005163	-0,13	-0,18	0,16	0,455342211	0,15
12	-0,1	-0,04984019	-0,41	-0,06	0,198986668	-0,48	0,07	-0,13	0,308590891	-0,48
13	0,31	-0,3745127	-0,23	-0,48	-0,32046471	0,09	0,47	-0,02	-0,141920101	-0,05
14	-0,09	0,171977422	0,45	0,22	0,309173844	0,07	-0,37	-0,36	0,235475376	-0,13
15	0,09	0,363983066	0,16	-0,01	0,389380383	-0,28	0,21	-0,35	0,234106532	-0,31
16	-0,44	0,25009713	-0,19	-0,35	-0,43101979	0,22	-0,5	-0,22	-0,101991332	0,3
17	0,11	0,393356302	-0,04	-0,33	-0,11068416	0,19	0,45	-0,5	0,388284296	-0,33
18	-0,14	0,245738005	-0,42	-0,11	-0,2604389	-0,4	-0,09	0,44	0,300408794	-0,16
19	-0,13	-0,04526482	0	-0,13	0,189457835	0,39	-0,12	0,21	-0,058906784	-0,47
20	-0,15	-0,50620496	0,27	-0,43	-0,09063898	-0,09	0,31	-0,01	-0,327640139	-0,2
21	-0,24	0,05	0,01	0,14	-0,08	-0,45	0	-0,38	-0,41	0,41
22	0,06	-0,46858667	-0,19	0,42	-0,03088425	0,16	0,49	0,24	0,270235364	-0,06
23	-0,23	-0,26401129	-0,17	-0,12	0,229586922	0,33	0,11	-0,03	0,092737688	-0,07
24	-0,19	0,162729538	-0,46	0,43	-0,3207487	0,16	-0,26	0,5	0,19246206	-0,27
25	-0,3	0,370473188	-0,18	-0,29	0,48901894	0,23	-0,34	0,42	0,09150201	-0,19
26	0,08	0,256239416	0,1	-0,44	0,439612739	0,13	0,44	-0,47	-0,004933417	-0,33
27	0,27	-0,09288311	-0,21	-0,23	0,0997031	-0,15	-0,44	0,36	-0,476782287	0,15

Selanjutnya hitung *update* nilai parameter filter F[1], F[2], dan F[3]. Berikut adalah perhitungan untuk *update* nilai parameter bobot $F[1]_{1,1}$ berdasarkan persamaan (2.15).

$$F[1]_{1,1}' = F[1]_{1,1} - \alpha(\frac{\partial P[1]}{\partial F[1]_{1,1}})$$

$$F[1]_{1,1}' = 0{,}42 - 0.01(0{,}065398437)$$

$$F[1]_{1,1}' = 0{,}413460156$$

Hasil penghitungan F′ yang lainnya dapat dilihat pada tabel-tabel berikut.

Tabel 3. 32 Matriks Filter F[1]'

x/y	1	2	3
1	0,413460156	0,486682966	0,004707284
2	0,481975681	-0,325868427	0,181373232
3	0,365876572	0,12311823	-0,339018551

Tabel 3. 33 Matriks Filter F[2]'

x/y	1	2	3
1	-0,36	-0,190452622	-0,290365179
2	0,295300423	0,448465013	-0,415191152
3	0,260658688	-0,23	-0,44

Tabel 3. 34 Matriks Filter F[3]'

x/y	1	2	3
1	-0,24220851	0,272574835	-0,46
2	0,141376843	0,085187922	-0,452847668
3	-0,50553824	0,214461764	0,495228183

Hasil *update* parameter-parameter bobot V', bobot W' dan filter F' akan digunakan pada iterasi *feedforward* selanjutnya.

3.7 Analisis Kebutuhan Non Fungsional

Analisi Kebutuhan non fungsional adalah sebuah langkah untuk menganalisis sumber daya yang akan digunakan perangkat lunak yang dibangun. Analisis non fungsional akan dilakukan dibagi 3 tahap, yaitu :

1. Analisis Kebutuhan Perangkat Keras
2. Analisis Kebutuhan Perangkat Lunak
3. Analisis Pengguna

3.7.1 Analisis Kebutuhan Perangkat Keras

Analisis kebutuhan perangkat keras pada penelitian ini merupakan kebutuhan perangkat keras yang digunakan dalam membangun system untuk implementasi algoritma yang digunakan dalam penelitian. Adapun perangkat keras yang digunakan sebagai berikut :

1. Laptop intel(R) Core(TM) i5-3210M CPU @2,50GHz (4 CPUs), ~2.5GHz
2. Ram 4Gb
3. Harddisk 500Gb
4. VGA AMD Radeon R5 M330 Graphics 2Gb

3.7.2 Analisis Kebutuhan Perangkat Lunak

Analisis kebutuhan perangkat lunak pada penelitian ini merupakan *tools* yang digunakan dalam membangun sistem untuk implementasi algoritma yang digunakan dalam penelitian. Adapun perangkat lunak yang digunakan sebagai berikut:

1. OS Linux Ubuntu 16.04 LTS 64bit
2. Python 3.0
3. PyCharm 2018

3.7.3 Analisis Pengguna

Karakteristik pengguna yang dapat menjalankan sistem yang akan dibangun hanya terdapat satu jenis pengguna yaitu seorang penguji. Penguji bekerja untuk menjalankan serta mengetahui hasil dari aplikasi yang dijalankan. Adapun spesifikasi pengguna yang dibutuhkan adalah sebagai berikut:

1. Menguasai penggunaan komputer.
2. Mengerti secara teknis *tools* dan *software* pendukung dalam menjalankan aplikasi.
3. Mengerti tahap-tahapan dalam menjalankan aplikasi.
4. Memahami proses dan kebutuhan dalam menggunakan aplikasi.

3.8 Analisis Kebutuhan Fungsional

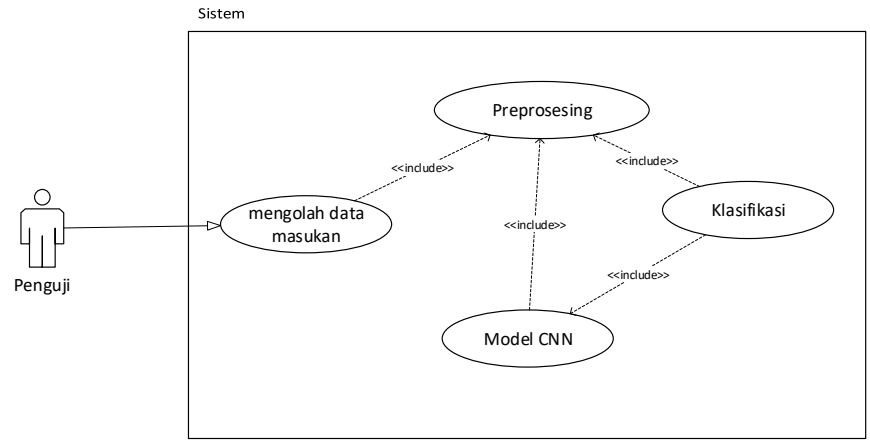
Analisis kebutuhan fungsional bertujuan untuk menganalisis proses yang diterapkan dalam sistem dan menjelaskan kebutuhan yang diperlukan. Menganalisis operasional sistem dengan mendefinisikan skenario penggunaan aplikasi. Analisis yang dilakukan dimodelkan dengan menggunakan UML (*Unified Modeling Language*).

UML merupakan bahasa standar untuk merancang dan mendokumentasikan perangkat lunak dengan cara berorientasi objek. Bagian-bagian yang dilakukan dalam analisis tersebut antara lain *use case diagram, use case scenario, activity diagram, class diagram* dan *sequence diagram.*

3.8.1 *Use Case* Diagram

Use case diagram adalah gambaran umum sistem dari sudut pandang pengguna sistem. Tujuan dari *use case* adalah untuk menggambarkan apa yang sistem dapat lakukan. *Use case diagram* dibentuk dari skenario tentang kegunaan sistem yang dinotasikan dengan sebuah *use case*. Setiap skenario menjelaskan suatu alur kegiatan, dapat diinisialisai oleh pengguna sistem yang disebut aktor.

Use case diagram dapat memperlihatkan hubungan-hubungan yang terjadi antara aktor-aktor dengan *use case* dalam sistem. Pengguna dapat mengamati *use case diagram* untuk mendapatkan pemahaman yang utuh tentang pembangunan aplikasi mengenai, implementasi metode *convolutional neural network* dalam memprediksi kemenangan atlet dalam suatu pertandingan berdasarkan faktor kondisi fisik dan kondisi kesehatan atlet saat itu. Adapun *use case diagram* dapat dilihat pada gambar 3.19.



Gambar 3. 19 *Use Case* Diagram

Berikut ini merupakan deskripsi actor yang dapat dilihat pada tabel 3.26 Berikut:

Tabel 3. 35 Deskripsi Aktor

Aktor	Deskrispi
Penguji	Penguji dengan aturan ini memiliki kewenangan untuk melakukan persiapan mengolah data masukan seperti data pembelajaran, data pengenalan, *preprocesing* pembuatan model, dan klasifikasi.

Berikut merupakan deskripsi *use case* yang dapat dilihat pada tabel 3.27 Berikut:

Tabel 3. 36 Deskripsi *Use Case* Diagram

No	Use Case	Deskrispi
1	Mengolah data masukan	Fungsionalitas yang digunakan oleh penguji untuk memilih data, data pembelajaran dan data pengenalan ke dalam sistem.
2	Preprocesing	Fungsionalitas yang digunakan oleh penguji untuk memproses data pembelajaran dan data pengenalan ke dalam sistem. Data masukan diproses menjadi citra keabuan dengan merubah dimensi image 3 dimensi pada data masukan menjadi 1 dimensi, kemudian data citra keabuan diproses dengan menggukan metode *deteksi tepi sobel* dan selanjutnya dinormalisasi.
3	Model CNN	Fungsionalitas yang digunakan oleh penguji untuk membuat model pada data pembelajaran menggunakan metode CNN *(convolution neural network)*
4	Klasifikasi CNN	Fungsionalitas yang digunakan oleh penguji pada data pengenalan untuk mendapatkan prediksi hasil klasifikasi dari model CNN *(convolution neural network)*

3.8.2 *Use Case Scenario*

Bagian ini menjelaskan skenario untuk tiap *use case* yang menggambarkan urutan interaksi aktor dengan sistem.

Tabel 3. 37 *Use Case Scenario* Mengolah Data Masukan

Use Case Name	Mengolah data masukan	
Related Requirements	Data masukan berupa citra	
Goal Context	Penguji dapat mengolah data masukan untuk digunakan sebagai pembentukan model.	
Preconditions	Penguji mempunyai data masukan	
Successful End Condition	Data masukan berhasil diproses ke tahap selanjutnya	
Failed End Condition	Data masukan tidak berhasil diproses ke tahap selanjutnya	
Primary Actor	Penguji	
Trigger	Sistem menampilkan data masukan yang sudah terpilih	
Main Flow	*Step*	*Action*
	1	Pengui memilih tombol *upload*
	2	Sistem mengambil data masukan
	3	Data masukan ditampilkan
Extension	*Step*	*Branching Action*
	1	Data masukan berhasi mask sistem
	2	Data masukan gagal mask sistem

Tabel 3. 38 *Use Case Scenario Preprocesing*

Use Case Name	Preprocesing
Related Requirements	-
Goal Context	Melakukan proses deraat keabuan pada data masukan kemudian melakukan deteksi tepi menggunakan metode *sobel edge detection*
Preconditions	Data masukan berhasil masuk sistem
Successful End Condition	Sistem berhasil memproses *preprocessing*
Failed End Condition	Sistem gagal memproses derajat *preprocessing*
Primary Actor	penguji

Trigger			
Main Flow	*Step*	*Action*	
		User	*System*
	1	Data masukan citra berhasil masuk sistem	
	2	Sistem melakukan proses citra keabuan	
	3	Sistem melakukan proses deteksi tepi	
Extension	*Step*	*Branching Action*	
	1	Sistem berhasil melakukan proses preprocessing	
	2	Sistem gagal melakukan proses preprocessing	

Tabel 3. 39 *Use Case Scenario* Model CNN

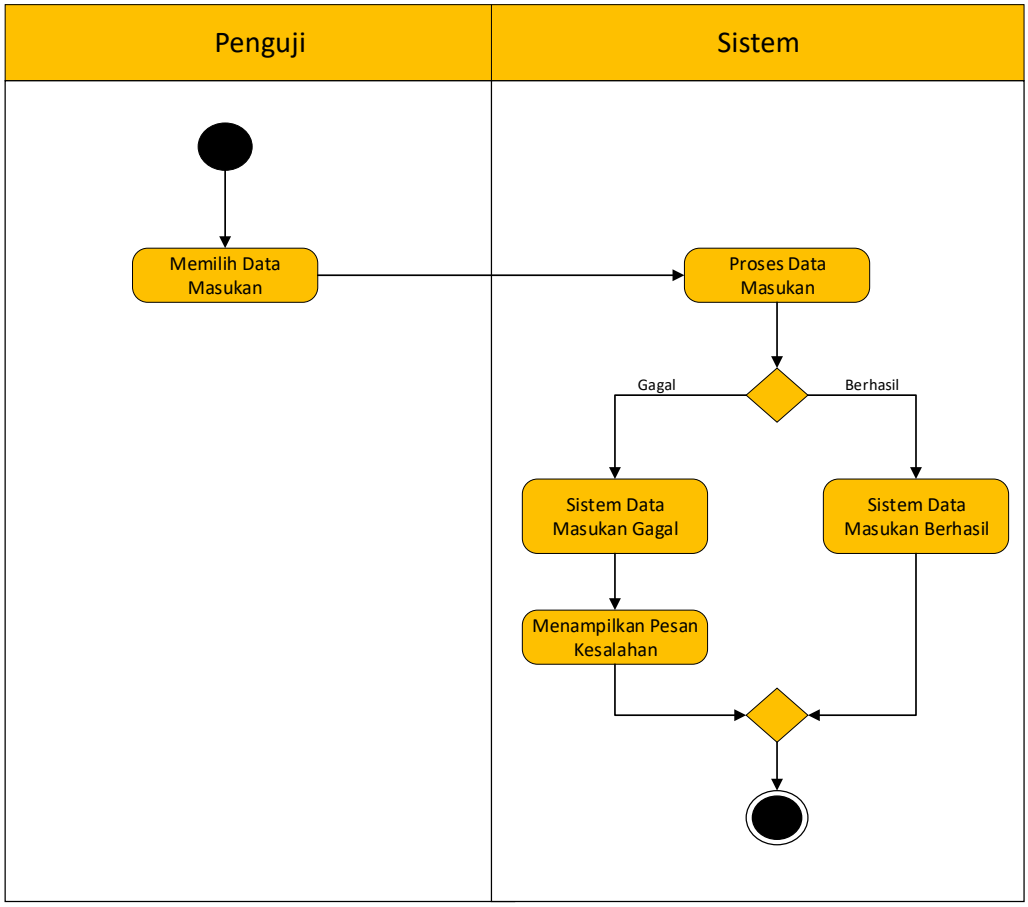
Use Case Name	Model CNN		
Related Requirements	Data hasil preprcessing berhasil masuk sistem		
Goal Context	Penguji dapat memasukan data pembelaaran untuk pebentukan model CNN		
Preconditions	Sistem berhasil melakukan preprocessing		
Successful End Condition	Model CNN berhasil disimpan		
Failed End Condition	Model CNN gagal disimpan		
Primary Actor	Penguji		
Trigger			
Main Flow	*Step*	*Action*	
		User	*System*
	1	Sistem memiliki data hasil *preprocessing*	
	2	Sistem melakukan proses *convolution*	
	3	Sistem melakukan proses *non linearity*	
	4	Sistem melakukan proses *maxpooling*	
	5	Sistem melakukan proses *fully conected*	
	6	Sistem melakukan proses *pembentukan model*	
Extension	*Step*	*Branching Action*	
	1	Sistem berhasil melakukan pembentukan model	
	2	Sistem gagal melakukan proses pembentukan model	

Tabel 3. 40 *Use Case Scenario* Klasifikasi CNN

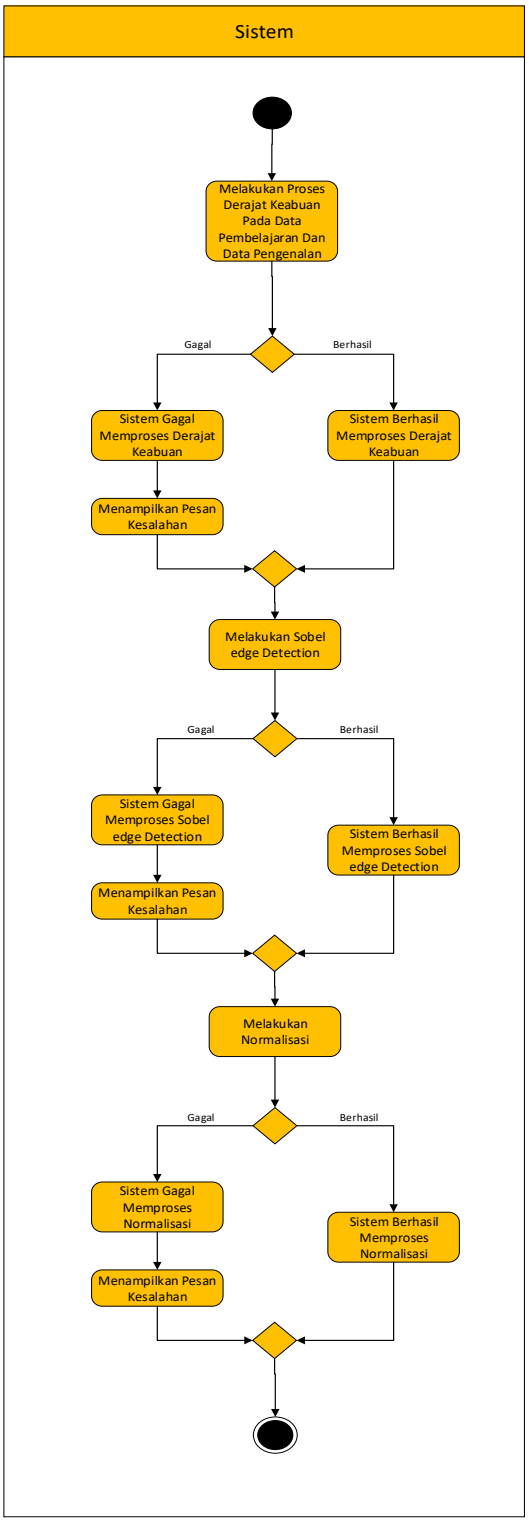
Use Case Name	Klasifikasi CNN		
Related Requirements	-		
Goal Context	Penguji dapat mengkalsifikasi data pengenalan		
Preconditions	Sistem telah memiliki model CNN		
Successful End Condition	Berhasil mengklasifikasi data pengenalan		
Failed End Condition	Gagal mengklasifikasi data pengenalan		
Primary Actor	Penguji		
Trigger			
Main Flow	*Step*	*Action*	
		User	*System*
	1	Sistem memiliki data hasil *preprocessing*	
	2	Sistem melakukan proses *convolution*	
	3	Sistem melakukan proses *non linearity*	
	4	Sistem melakukan proses *maxpooling*	
	5	Sistem melakukan proses *fully conected*	
	6	Sistem melakukan klasifikasi menggunakan model CNN	
Extension	*Step*	*Branching Action*	
	1	Sistem berhasil melakukan klasifikasi	
	2	Sistem gagal melakukan klasifikasi	

3.8.3 *Activity* Diagram

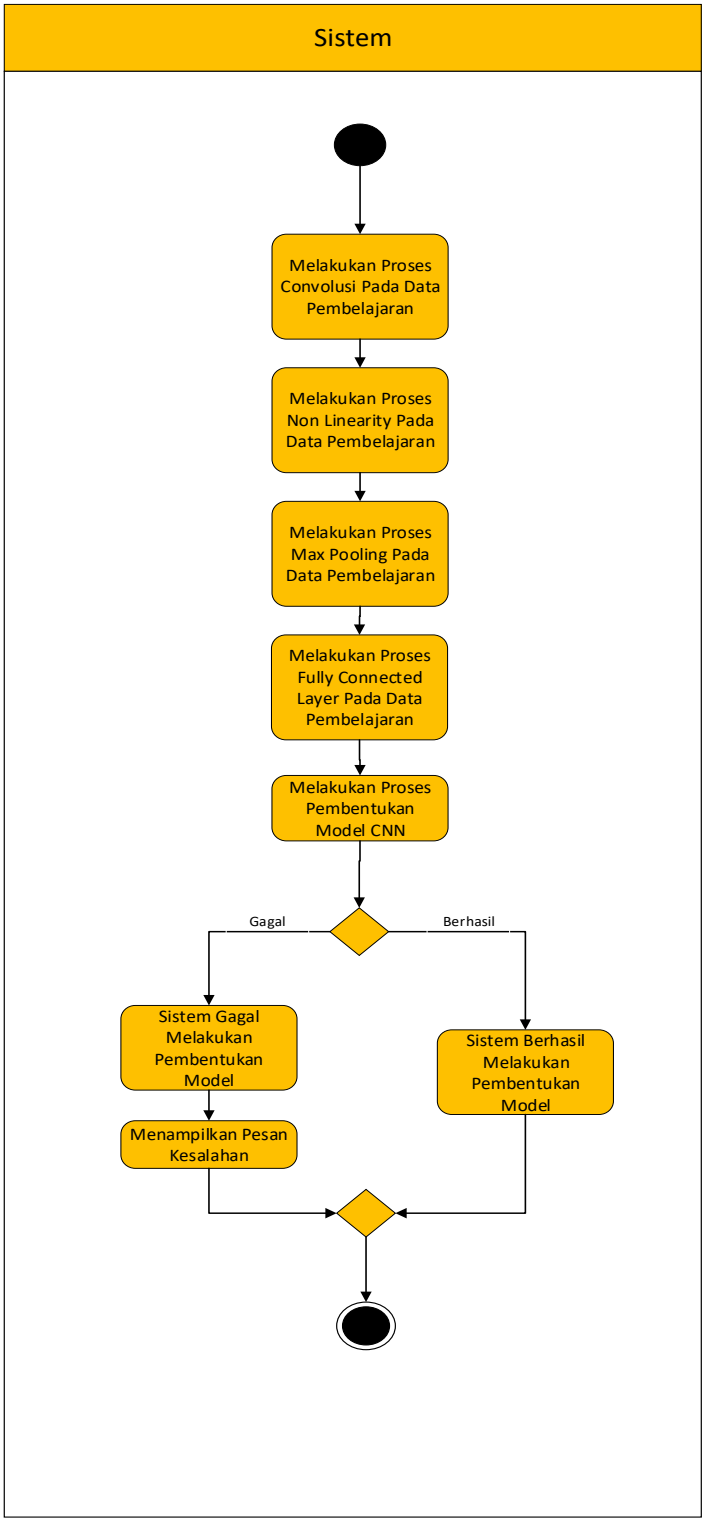
Activity diagram menggambarkan berbagai alur aktivitas dalam sistem yang dirancang, menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum, menggambarkan proses bisnis dan urutan aktivitas dalam sebuah proses, bagaimana masing-masing alur berawal, *decision* yang mungkin terjadi, dan bagaimana alur berakhir. Adapun *activity diagram* dari masing-masing *use case scenario*.



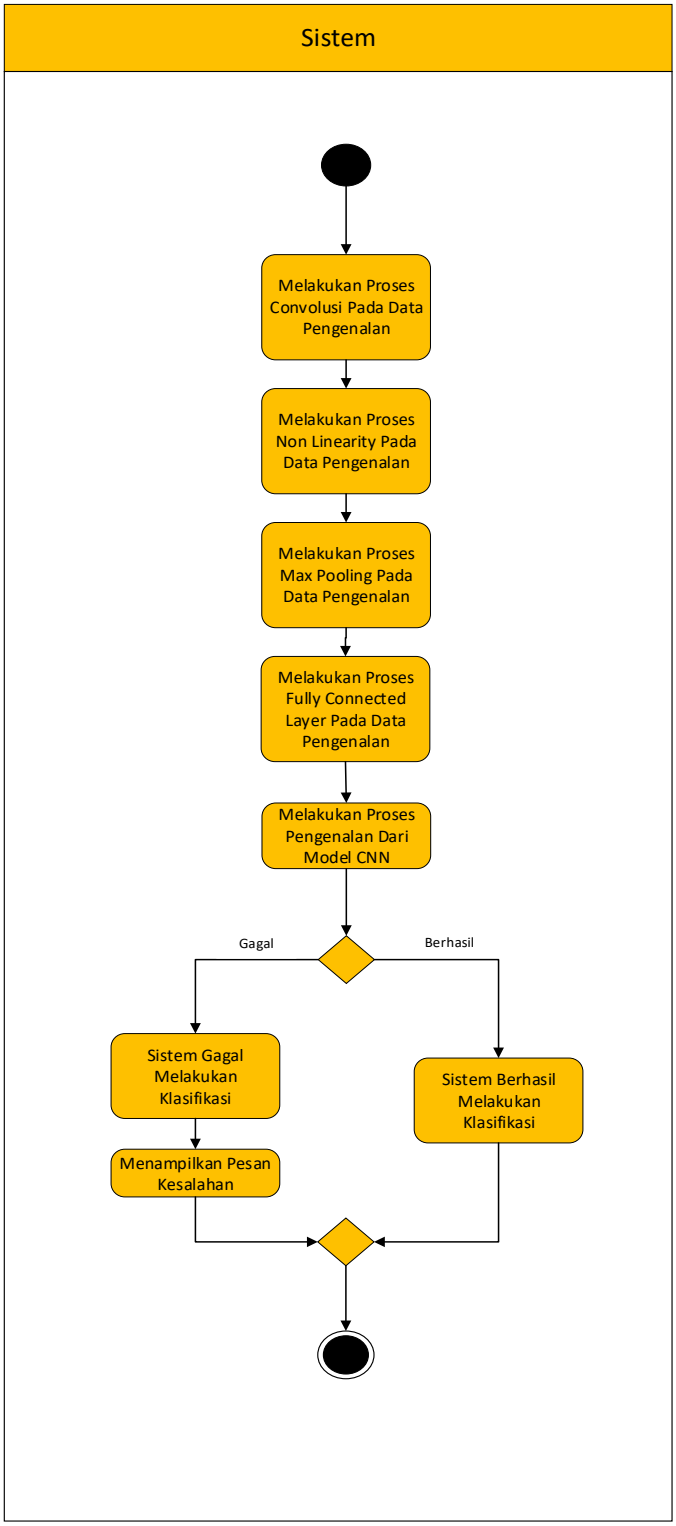
Gambar 3. 20 *Activity* Diagram Mengolah Data Masukan



Gambar 3. 21 *Activity* Diagram *Preprocessing*



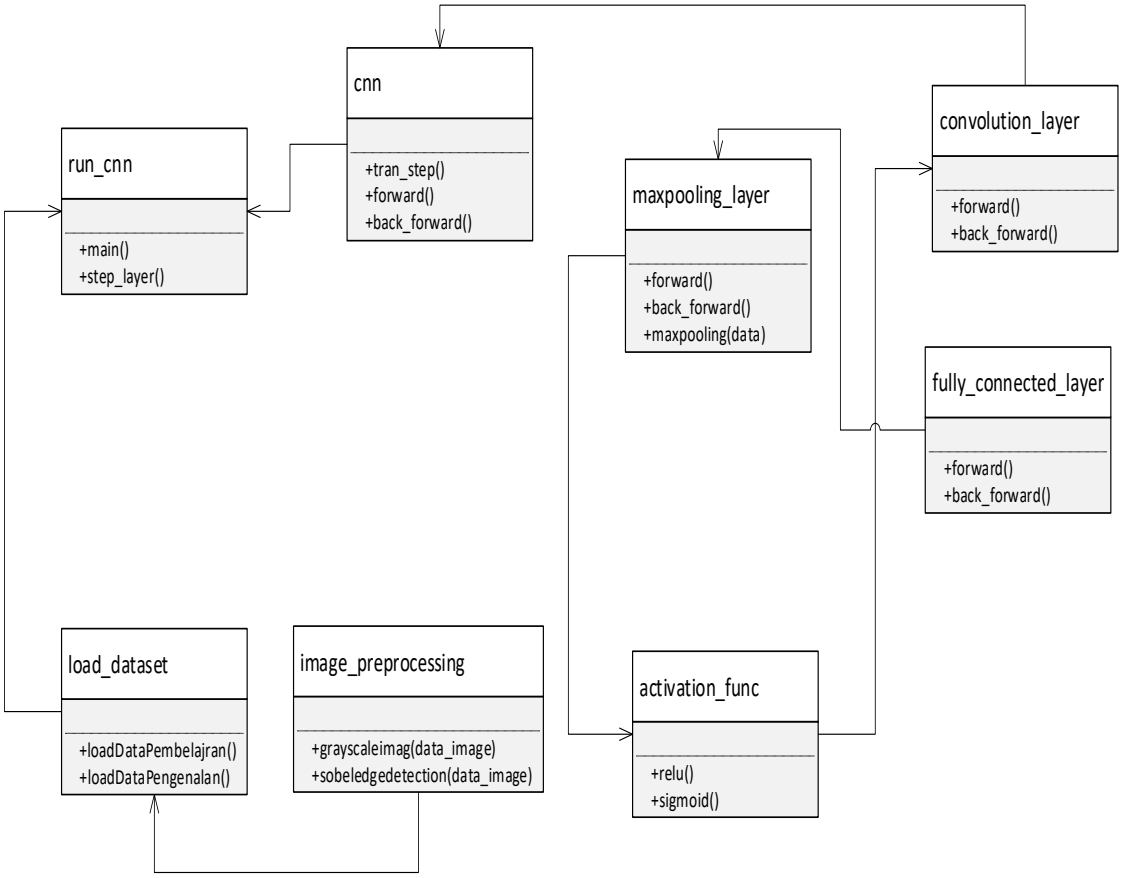
Gambar 3. 22 *Activity* Diagram Model CNN



Gambar 3. 23 *Activity* Diagram Klasifikasi

3.8.4 *Class Diagram*

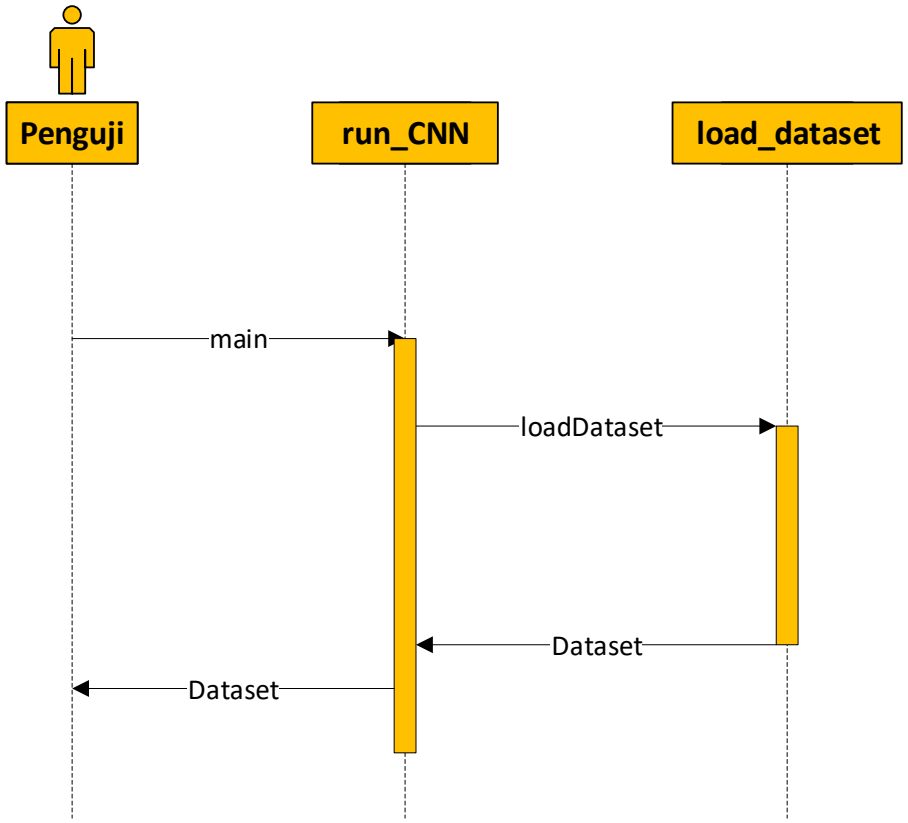
Class diagram adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari desain berorientasi objek. Berikut adalah penjelasan bentuk *class diagram* pada gambar 3.24 berikut:



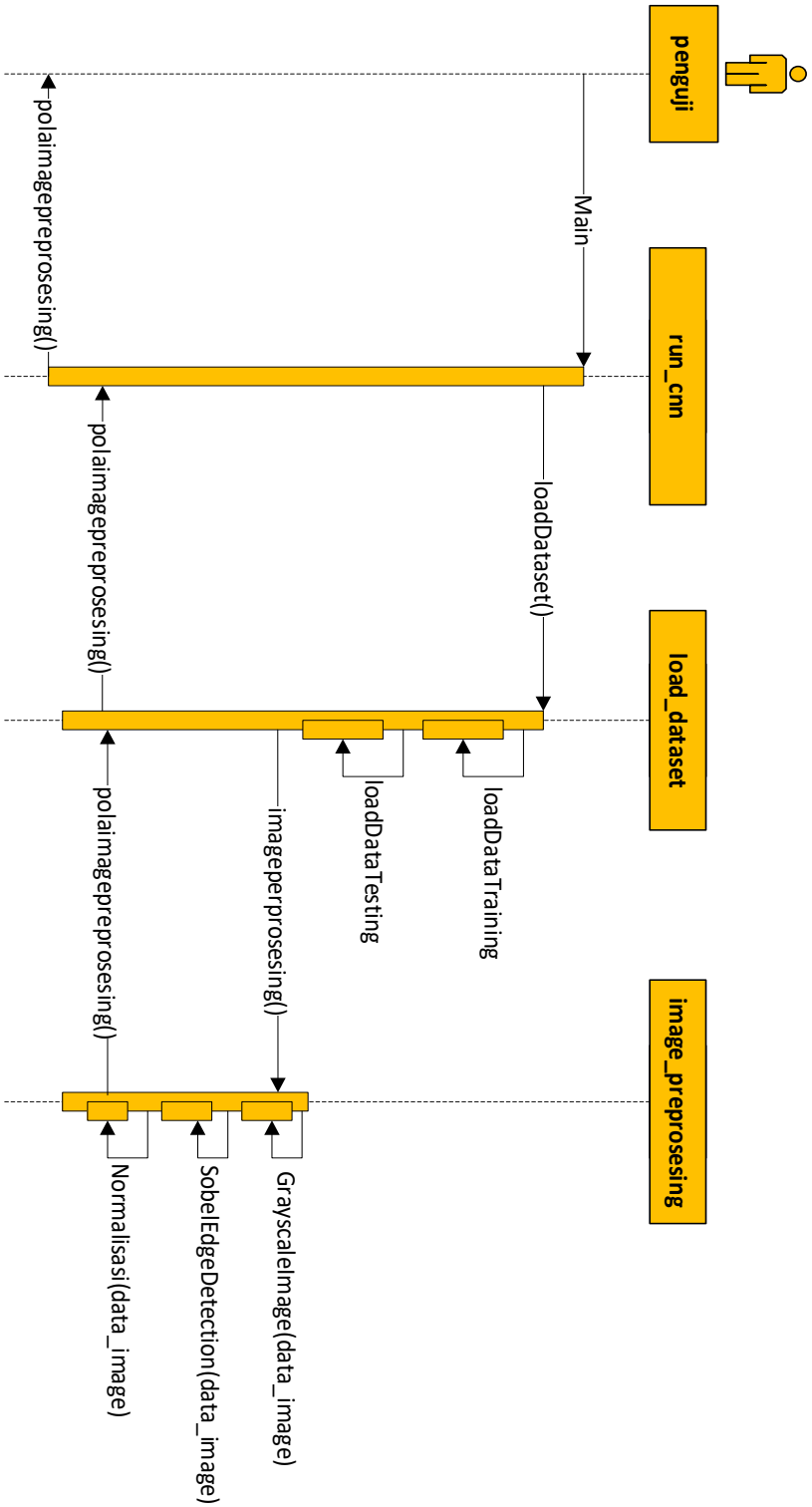
Gambar 3. 24 *Class Diagram*

3.8.5 *Sequence Diagram*

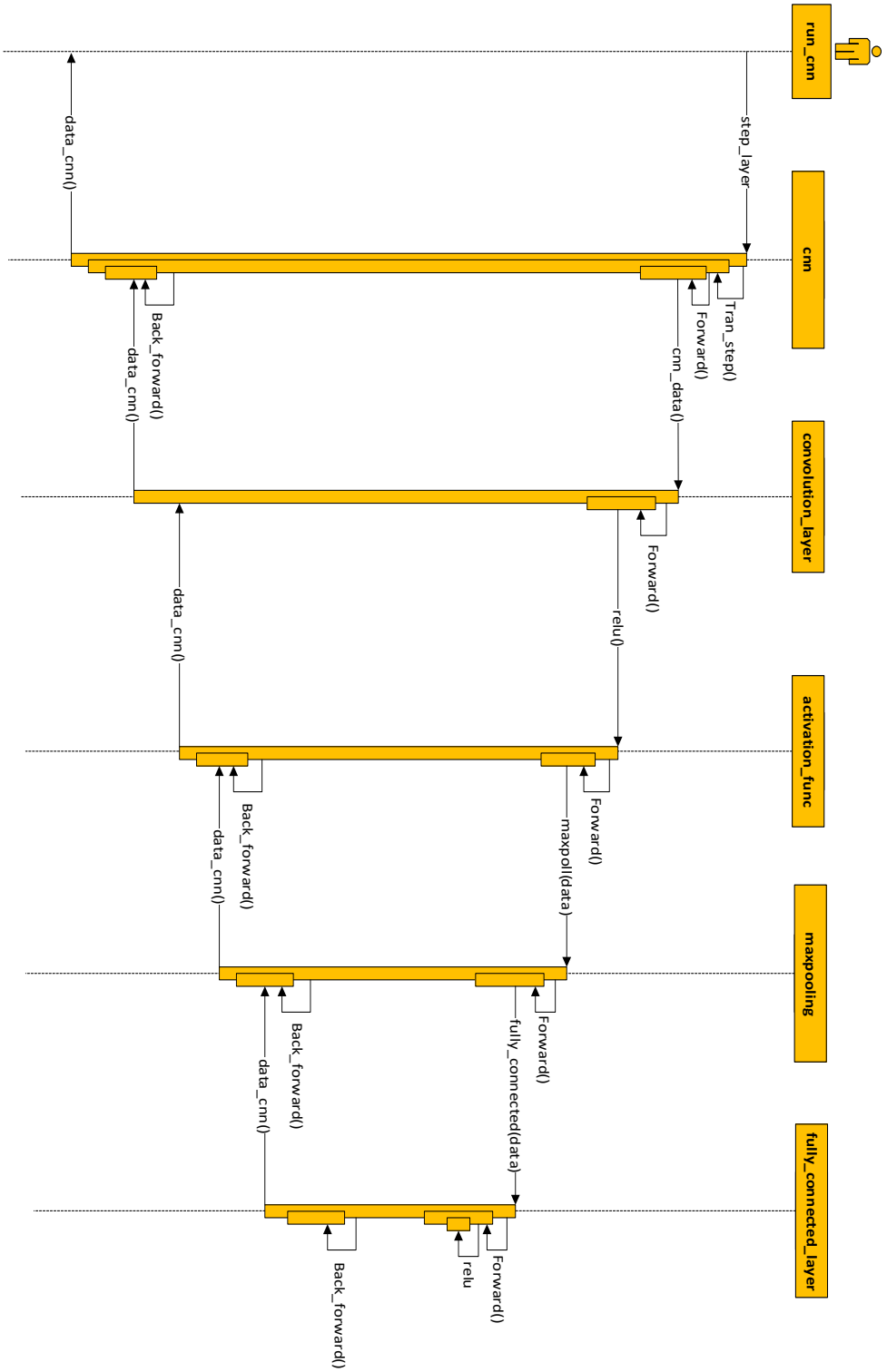
Berikut ini adalah *sequence diagram* yang terdapat pada pembangunan sistem yang dibangun sebagai berikut:



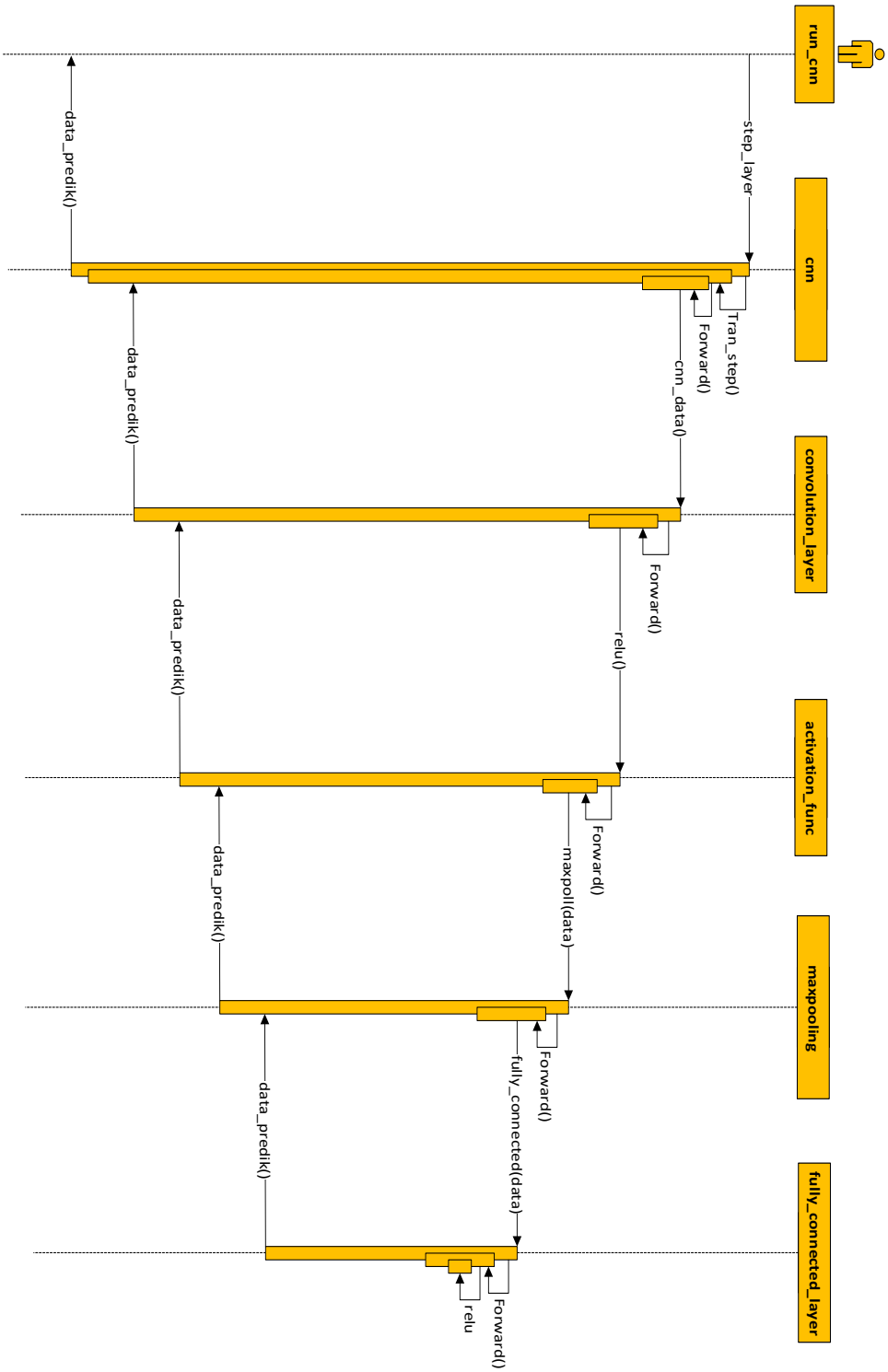
Gambar 3. 25 *Sequence* Diagram Mengolah Data Masukan



Gambar 3. 26 *Sequence* Diagram Preprocessing



Gambar 3. 27 *Sequence* Diagram Model CNN



Gambar 3. 28 *Sequence* Diagram Klasifikasi

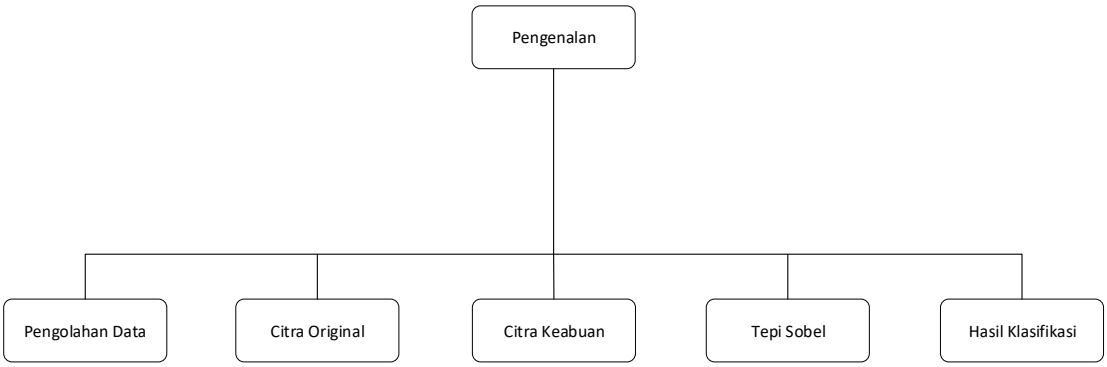
3.9 Perancangan Sistem

Perancangan sistem merupakan penggambaran dan perancangan sistem untuk mengimplementasikan hasil analisis yang dilakukan sebelumnya.

Perancangan system yang telah dibuat terdiri dari struktur menu dan perancangan antarmuka.

3.9.1 Struktur Menu

Perancangan struktur menu merupakan gambar alur pemakaian sistem, sehingga sistem yang dibangun mudah dipahami dan mudah digunakan. Berikut ini adalah perancangan struktur menu yang akan diterapkan pada sistem. Adapun perancangan struktur menu dapat dilihat pada gambar 3.29 berikut:



Gambar 3. 29 Struktur Menu

3.10 Perancangan Antarmuka

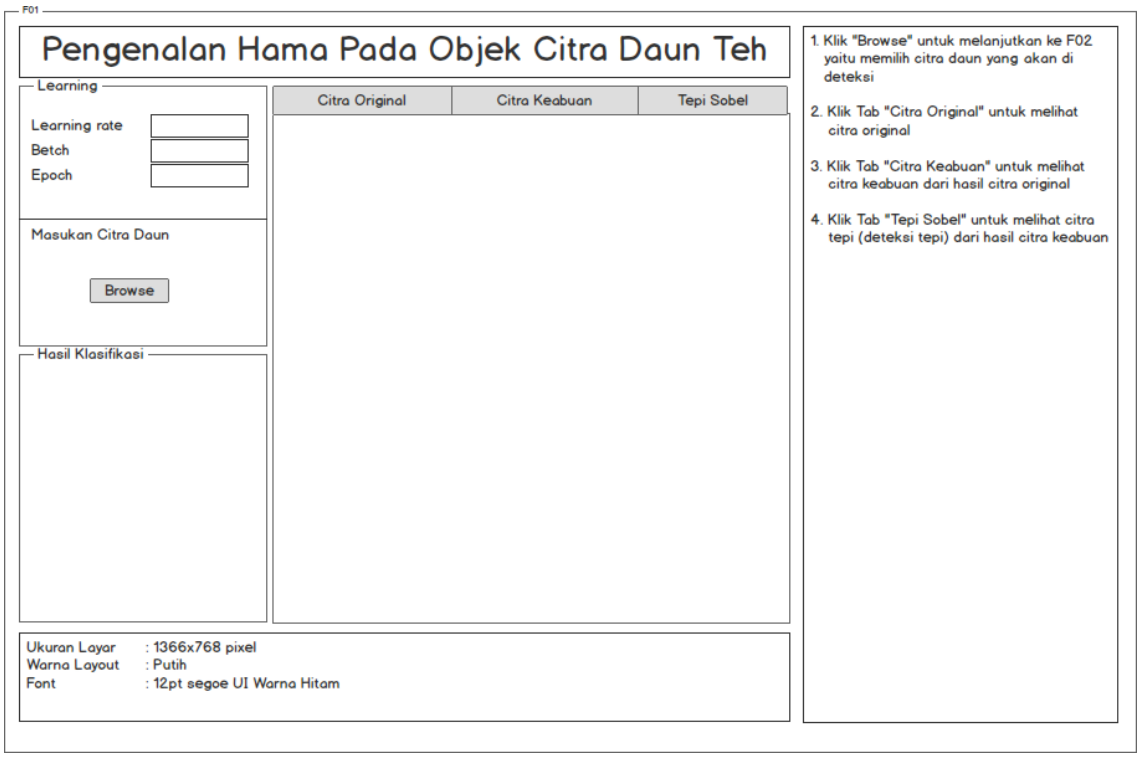
Perancangan antarmuka mendeskripsikan rencana tampilan yang akan digunakan pada sistem yang akan dibangun. Perancangan antarmuka terdiri dari perancangan form, perancangan pesan dan jaringan semantik.

3.10.1 Perancangan *Form*

Perancangan *form* dalam menentukan tata letak dari tampilan sistem yang akan dibuat. Rancangan tersebut dibuat sebagai perancangan *form* pada sistem. Berikut adalah perancangan *form* pada aplikasi yang akan dibangun.

1. Perancangan Form Antar Muka Utama (F01)

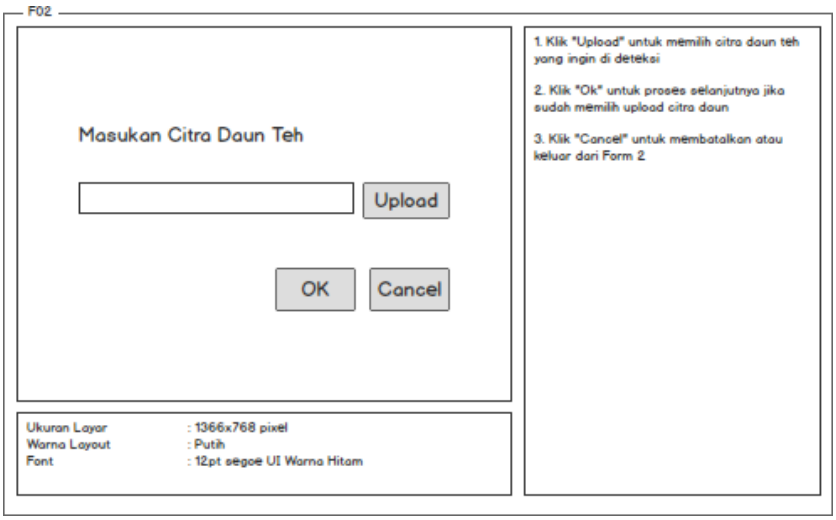
Perancangan F01 merupakan halaman utama pada saat pertama kali menjalankan program. Tampilan *default* awal program diawali tampilan pengenalan citra daun dan masuk ke F02. Perancangan F01 dapat dilihat pada gambar 3.30.



Gambar 3. 30 *Form* Mengolah Citra Daun

2. Perancangan Form Antar Muka F02

Perancangan F02 merupakan halaman untuk memilih citra daun yang akan di deteksi pada directory. Perancangan F02 dapat dilihat pada gambar 3.31.

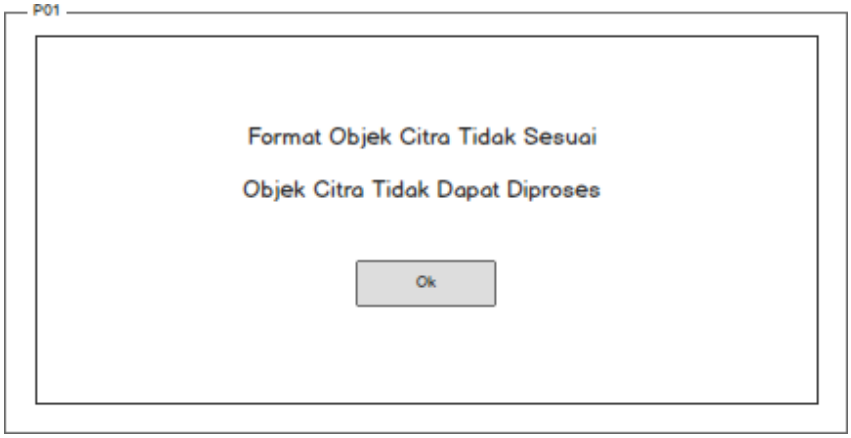


Gambar 3. 31 *Form* Memilih Citra Daun

3.10.2 Perancangan Pesan

Perancangan pesan merupakan validasi dan pesan yang akan disampaikan oleh sistem dalam keadaan-keadaan tertentu.

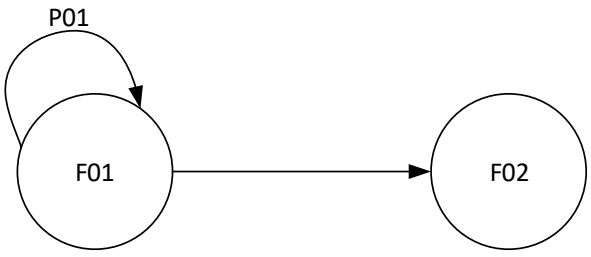
1. Perancangan pesan pada saat memilih citra berformat selain JPG, JPEG atau PNG ketika menekan tombol Ok pada upload citra daun teh, maka akan tampil pesan P01. Adapun perancangan pesan P01 dapat dilihat pada gambar 3.32.



Gambar 3. 32 Perancangan Pesan P01

3.10.3 Jaringan Semantik

Jaringan semantik menggambarkan keterhubungan navigasi menu dari satu antarmuka ke antarmuka lain. Jaringan semantik yang terbentuk pada sistem ini dapat dilihat pada gambar 3.33 sebagai berikut:



Gambar 3. 33 Jaringan Semantik