BAB II

LANDASAN TEORI

2.1. Penyakit Tanaman Jagung

1. Hawar Daun

Hawar daun adalah penyakit yang dapat terjadi pada daun jagung di Indonesia karena serangan cendawan dengan nama *Helminthosporium* sp. Semangun (1991), (Sudjono, 2018), (Wakman dkk. 2016) menyebutkan, ciri daun jagung yang telah terserang cendawan ini adalah penampakkan bercak coklat keabuan bergaris dengan panjang 2,5 hingga 15 cm dan lebar 1 hingga 2 cm pada daun oleh *H. turcicum*, seperti yang dapat dilihat pada gambar 2.1.

Gambar 2.1. Penyakit Hawar pada Daun Jagung (Huda, 2018)

Penyakit ini dapat bertahan hidup pada tanaman yang sehat, sisa dari tanaman yang sudah sakit, maupun pada biji jagung. Penyebaran penyakit ini dilakukan melalui udara dan memiliki tingkat konsentrasi yang tinggi pada siang hari (Wakman dkk. 2016).

2. Bercak Daun

Bercak daun adalah penyakit yang terjadi karena serangan cendawan *Helminthosporium* sp, sama seperti hawar daun tetapi varian *H. maydis*. Semangun (1991), (Sudjono, 2018), (Wakman dkk. 2016) juga menyebutkan, ciri daun yang terkena serangan *H. maydis* yaitu penampakkan bercak coklat kemerahan dengan panjang 1,2 hingga 1,9 cm dan lebar 0,6 cm dengan posisi yang sejajar dengan pinggiran bercak berwarna kuning tua hingga coklat, seperti yang dapat dilihat pada gambar 2.2.

Gambar 2.2. Penyakit Bercak pada Daun Jagung (Huda, 2018)

Penyakit ini dapat bertahan hidup pada sisa-sisa tanaman hasil panen dan biji jagung yang telah terinfeksi. Penyebaran penyakit ini dapat dilakukan melalui udara dan air (Wakman dkk, 2016).

3. Karat Daun

Karat daun adalah penyakit yang terjadi karena serangan cendawan *Puccinia polysora*. Sudjono (2018), (Wakman dkk, 2016) menyebutkan, ciri daun yang terkena serangan adalah penampakkan bercak-bercak kecil bulat hingga oval berwarna coklat atau merah oranye dengan panjang 0,2 hingga 2 mm, seperti yang dapat dilihat pada gambar 2.3.

Gambar 2.3. Penyakit Karat pada Daun Jagung (Huda, 2018)

Penyakit ini hidup dan menyebarkan dirinya melalui udara dengan menyebarkan *aeciospora* yang dihasilkan oleh *spermatia* yang berfusi dengan *hipa* lentur yang kemudian membentuk *uredospora* (Wakman dkk, 2016).

2.2. Pre-Processing

Praproses adalah proses transformasi data citra menjadi data numerikal yang dapat diolah oleh komputer dan dinormalisasi dengan data lainnya yang ada di dalam dataset. Proses ini bertujuan untuk mencegah adanya data *outlier* yang dapat mengurangi tingkat akurasi dari model yang dilatih. Proses *pre-processing* yang digunakan dalam penelitan ini adalah konversi warna gambar menjadi tingkat keabuan menggunakan *gray level co-occurrence matrices* dan dinormalisasi menggunakan filter median.

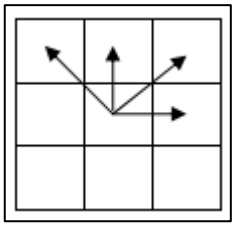
1. Gray Level Co-Occurrence Matrices

Gray Level Co-Occurrence Matrices (GLCM) adalah sebuah metode yang dapat untuk mengekstraksi karakteristik dari sebuah objek yang melibatkan relasi spatial dari pixel yang ada disekitarnya atau yang disebut sebagai *gray-level spatial dependence matrix* (Suresh, 2012).

GLCM merupakan basis dari teori Haralick untuk melakukan ekstraksi fitur berdasarkan tekstur dari sebuah gambar. Sebuah GLCM memiliki ukuran sebesar $p[_{i,j}]$, dengan besar i dan j mengikuti jumlah pixel yang ada pada rumus (2.1).

$$p[i,j] = \begin{bmatrix} p(0,0) & p(1,0) & \dots & p(i_n,0) \\ p(0,1) & p(1,1) & \dots & p(i_n,1) \\ \dots & \dots & \dots & \dots \\ p(0,j_n) & p(1,j_n) & \dots & p(i_n,j_n) \end{bmatrix} \quad \dots(2.1)$$

Haralick mengkalkulasi nilai dari *adjacency* yang ada di dalam matriks tersebut untuk mendapatkan *texture feature* dari gambar dengan merata-rata keempat *directional co-occurrence matrices* (horizontal, vertikal, diagonal kiri dan kanan), seperti yang dapat dilihat pada gambar 2.4.



Gambar 2.4. Directional Co-occurrence Matrices (Bino dkk, 2012)

Rumus perhitungan *directional co-occurrence matrices*:

$$P(i,j,d,0^{o}) =\neq \begin{Bmatrix} ((k,l),(m,n)) \in (L_y \times L_x) \times (L_y \times L_x)| \\ k-m=0, |l-n|=d, \\ I(k,l)=i, I(m,n)=j \end{Bmatrix} \quad \ldots(2.2)$$

$$P(i,j,d,45^{o}) =\neq \begin{Bmatrix} ((k,l),(m,n)) \in (L_y \times L_x) \times (L_y \times L_x)| \\ (k-m=d, l-n=-d)\ or \\ (k-m=-d, l-n=d), \\ I(k,l)=i, I(m,n)=j \end{Bmatrix} \quad \ldots(2.3)$$

$$P(i,j,d,90^{o}) =\neq \begin{Bmatrix} ((k,l),(m,n)) \in (L_y \times L_x) \times (L_y \times L_x)| \\ |k-m|=d, l-n=0, \\ I(k,l)=i, I(m,n)=j \end{Bmatrix} \quad \ldots(2.4)$$

$$P(i,j,d,135^{o}) =\neq \begin{Bmatrix} ((k,l),(m,n)) \in (L_y \times L_x) \times (L_y \times L_x)| \\ (k-m=d, l-n=d)\ or \\ (k-m=-d, l-n=-d), \\ I(k,l)=i, I(m,n)=j \end{Bmatrix} \quad \ldots(2.5)$$

Haralick dkk. (1973) yang kemudian dirangkum oleh Boland (1999) menjabarkan 14 metode perhitungan statistik yang dapat digunakan untuk mengekstraksi ciri tekstur dari sebuah gambar akan tetapi penelitian ini hanya menggunakan 9 metode seperti yang telah dinyatakan pada batasan masalah, antara lain:

a. Angular Second Moment (Enegy)

$$\sum_i \sum_j p(i,j)^2 \quad \ldots(2.6)$$

b. Contrast

$$\sum_{n=0}^{N_g-1} n^2 \left\{ \sum_{i=1}^{N_g} \sum_{j=1}^{N_g} p(i.j) \right\}, |i-j|=n \quad \ldots(2.7)$$

c. Correlation

$$\frac{\sum_i \sum_j (ij) p(i,j) - \mu_x \mu_y}{\sigma_x \sigma_y} \quad \ldots(2.8)$$

$\mu_x, \mu_y, \sigma_x, \sigma_y$ sebagai means dan standard deviations dari $p_x \; dan \; p_y$ dan fungsi partial probability density.

d. Sum of Squares: Variance

$$\sum_i \sum_j (i - \mu)^2 p(i,j) \quad \ldots(2.9)$$

e. Inverse Difference Moment

$$\sum_i \sum_j \frac{1}{(i - \mu)^2} p(i,j) = f_5 \quad \ldots(2.10)$$

f. Sum Average

$$\sum_{i=2}^{2N_g} i p_{x+y}(i) \quad ..(2.11)$$

x dan y sebagai koordinat co-occurrence matrix dari input dan $p_{x+y}(i)$ sebagai probabilitas dari koordinat co-occurrence matrix yang ditotalkan $x + y$.

g. Sum Variance

$$\sum_{i=2}^{2N_g} (i - f_5)^2 \, p_{x+y}(i) \quad ...(2.12)$$

h. Sum Entropy

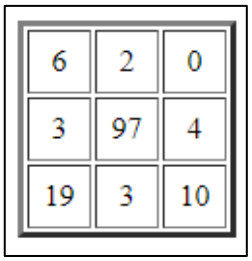
$$-\sum_{i=2}^{2N_g} p_{x+y}(i) \log\{p_{x+y}(i)\} \quad ..(2.13)$$

i. Entropy

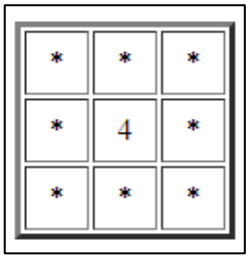
$$-\sum_i \sum_j p(i,j) \log\left(p_{x+y}(i)\right) \quad ..(2.14)$$

2. Median Filter

Adalah sebuah tahap pembersihan data pixel yang diterima dari *noise*. Hal ini dapat terjadi karena umumnya *noise* memiliki nilai warna yang jauh berbeda dengan nilai warna dari pixel tetangganya, sehingga filter ini ideal untuk digunakan. Filter ini mengambil sebagian dari gambar asli, umumnya dengan ukuran 3x3 pixel (Schulze, 2001), dengan isi dari pixel 3 x 3 tersebut diurutkan dan diambil nilai tengah yang kemudian akan merepresentasikan daerah dengan pixel 3 x 3 tersebut, seperti yang diilustrasikan pada gambar 2.5. dan 2.6.



Gambar 2.5. Matrix Sebelum di Filter



Gambar 2.6. Mean Filtered Matrix

3. Sobel Edge Detection

Adalah sebuah proses untuk gambar yang telah diubah ke dalam bentuk *greyscale* difilter menggunakan operator dari metode edge detection yang digunakan, dan operator yang digunakan pada penelitian ini adalah operator sobel. Sobel memiliki 2 operator yang masing – masing berfungsi untuk melakukan edge detection secara horisontal dan vertikal dengan sebutan Gx dan Gy. Operator sobel Gx dan Gy sebagai berikut.

$$Gx = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \text{...(2.15)}$$

$$Gy = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \quad \text{...(2.16)}$$

Edge detection dilakukan dengan mengambil sampel data matrix 3x3 dari gambar yang ingin di deteksi ujungnya, lalu matrix 3x3 itu dikalikan dengan operator Gx dan Gy. Hasil sum dari Gx dan Gy digunakan untuk menggantikan nilai yang berada di tengah dari matrix 3x3 tersebut, lalu prosesnya dilanjutkan dengan pengambilan sampel data matrix 3x3 pada pixel sebelahnya dan seterusnya hingga seluruh pixel dari gambar telah diambil sebagai sampel minimal 1 kali.

4. Otsu Thresholding

Thresholding merupakan sebuah proses normalisasi data dengan menggunakan sebuah nilai yang menandai batas bawah dari sekelompok nilai. Seluruh nilai yang berada di bawah nilai tersebut diubah nilainya menjadi 0 dan semua nilai yang ada di batas keatas nilainya dimaksimalkan menjadi 255.

Proses thresholding yang digunakan adalah algoritma threshold Otsu. Untuk menggunakan algoritma otsu, gambar terlebih dahulu dihaluskan menggunakan filter gaussian blur dengan menggunakan kernel 5x5 berikut (Fisher dkk, 2003).

$$Kernel\ Gaussian = \frac{1}{256}\begin{bmatrix}1 & 4 & 6 & 4 & 1\\ 4 & 16 & 24 & 16 & 4\\ 6 & 24 & 36 & 24 & 6\\ 3 & 16 & 24 & 16 & 4\\ 1 & 4 & 6 & 4 & 1\end{bmatrix} \quad ...(2.17)$$

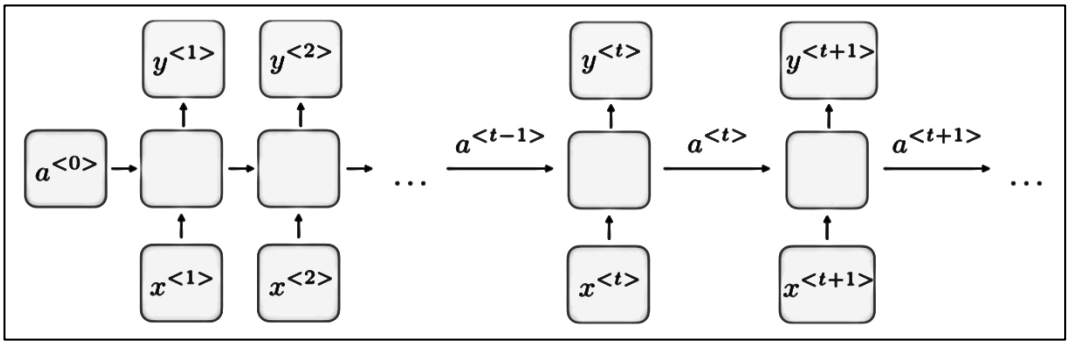
Otsu menghitung nilai yang dapat digunakan sebagai nilai ambang bawah yang optimal untuk gambar tersebut. Menurut Dr. Greensted (2010), otsu melakukan pemisahan pada data menjadi 2, yaitu latar depan dan latar belakang. Pemilihan latar depan dan latar belakang dilakukan dengan pencarian sekuensial dimulai dari nilai *greyscale* 0 menuju 255, dengan nilai yg berada di tengah 0 dan 255 dipilih untuk memisahkan nilai yg lebih kecil sebagai latar belakang dan yang sama besar atau lebih tinggi sebagai latar depan. Penentuan nilai thresholding yang optimal dilakukan dengan memilih nilai terkecil dari sum of variance. Rumus untuk mencari nilai ambang batas yang optimal sebagai berikut (Dr. Greensted, 2010).

$$Wihtin\ Class\ Variance = \sigma_W^2 = W_b\sigma_b^2 + W_f\sigma_f^2 \quad ..(2.18)$$

$$Between\ Class\ Variance = \sigma_B^2 = W_bW_f(\mu_b - \mu_f)^2 \quad ..(2.19)$$

5. Recurrent Neural Network

Sedikit berbeda dengan Convolutional Neural Network yang setiap simpulnya hanya dilewati sebanyak satu kali menuju *output layer*, Recurrent Neural Network (RNN) adalah sebuah jaringan yang setidaknya memiliki sebuah *feed-back connection* yang menerima hasil dari masukan sebelumnya digunakan sebagai masukan di dalam neuron itu kembali atau masukan di dalam neuron selanjutnya sehingga memungkinkan untuk sebuah *neural network* untuk mempelajari suatu rangkaian atau pola dari sebuah masukan (Bullinaria, 2015), seperti yang dapat dilihat pada gambar 2.7.



Gambar 2.7. Arsitektur Tradisional RNN (Amidi dkk, 2018)

Setiap layer RNN melakukan perhitungan *hidden state* yang kemudian digunakan sebagai masukan untuk layer selanjutnya dengan rumus berikut (PyTorch.org, n.d).

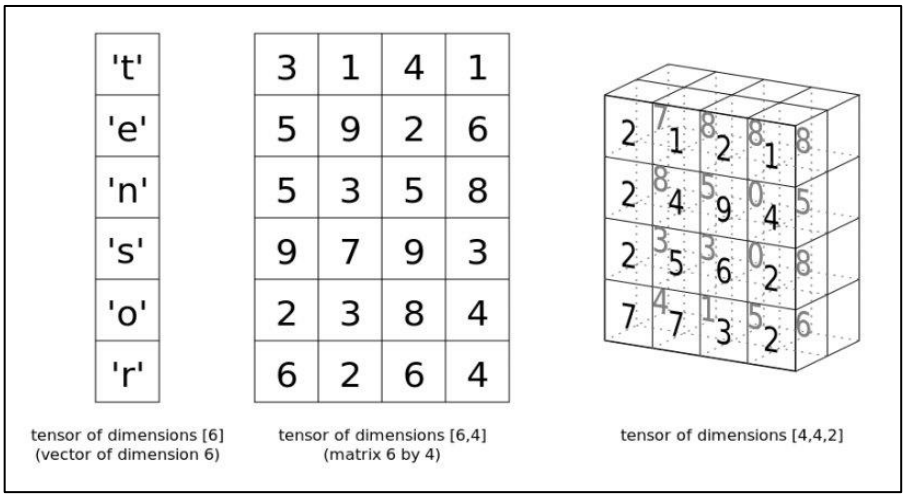
$$h_t = \tanh\left(W_{ih}X_t + b_{ih} + W_{hh}h_{(t-1)} + b_{hh}\right) \quad \text{...(2.20)}$$

Keterangan:

- W_{ih}, W_{hh}, bobot untuk masukan
- X_t, masukan
- $h_{(t-1)}$, *hidden state* dari t-1

6. PyTorch

Pytorch adalah sebuah framework pembelajaran mesin yang dapat digunakan untuk membangun sebuah model *deep learning* (Stevens, 2019). Pytorch dapat mengekspresikan model deep learning secara *idiomatic*. Idiomatic disini menyatakan bahwa pytorch dalam pengembangan model deep learningnya memiliki cara penulisan code yang harfiah dan *high level* sehingga penulisan codenya fleksibel karena tidak bergantung dengan fungsi yang ada, melainkan fungsi yang dibutuhkan dapat dibentuk sendiri. Pytorch memiliki sebuah struktur data yang bernama *tensor* yang bentuknya mirip seperti *array* multidimensional yang ada pada library numpy tetapi memiliki kapabilitas untuk menyimpan sebuah matrix 3 dimensi di dalam sebuah array yang selanjutnya array tersebut dapat di tambahkan menjadi matrix array 3 dimensi (Paul, 2018). Contoh ilustrasi tensor dapat dilihat pada gambar 2.8.



Gambar 2.8. Tensor (Mayo, 2018)