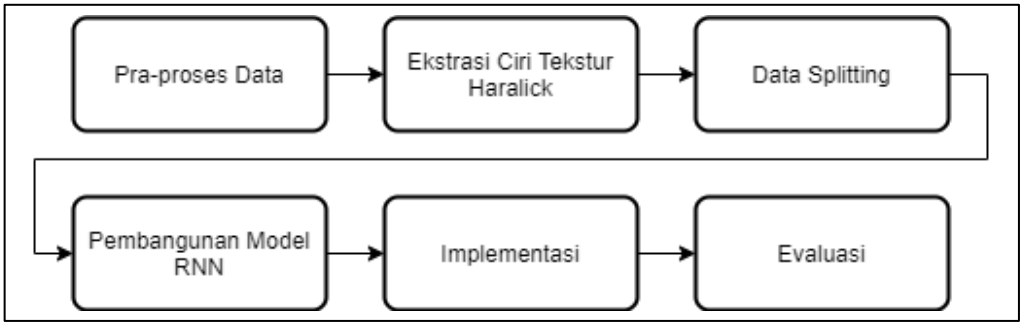
BAB III

METODOLOGI PENELITIAN DAN PERANCANGAN SISTEM

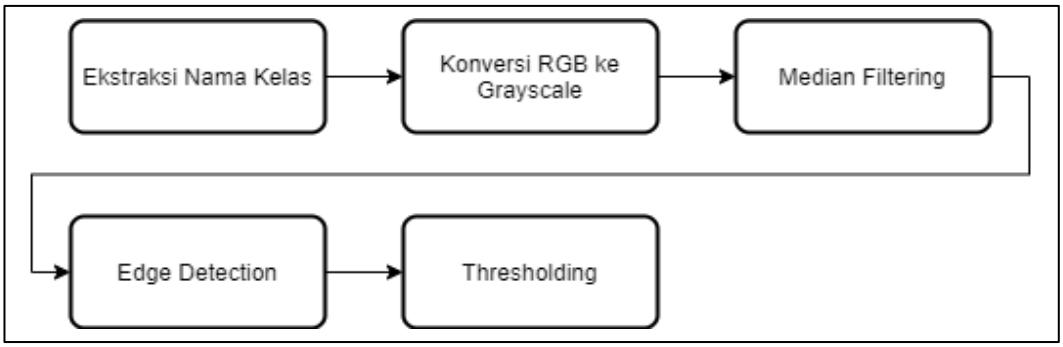
Pada penelitian ini terdapat beberapa tahapan yaitu praproses data, ekstraksi ciri tekstur, pembangunan model RNN, implementasi dan evaluasi. Tahapan akuisisi data tidak ada karena dataset yang digunakan sudah tersedia di domain publik github. Flowchart dari metodologi dapat dilihat pada gambar 3.1.



Gambar 3.1. Metode Penelitian

3.1. Pra-proses Data

Tahapan praproses terbagi menjadi 6 tahapan yaitu ekstraksi nama kelas dari masing-masing masukan, konversi warna rgb ke skalar keabuan, pemfilteran gambar menggunakan filter median, deteksi tepi, thresholding dan melakukan splitting data. Flowchart tahapan ini dapat dilihat pada gambar 3.2.

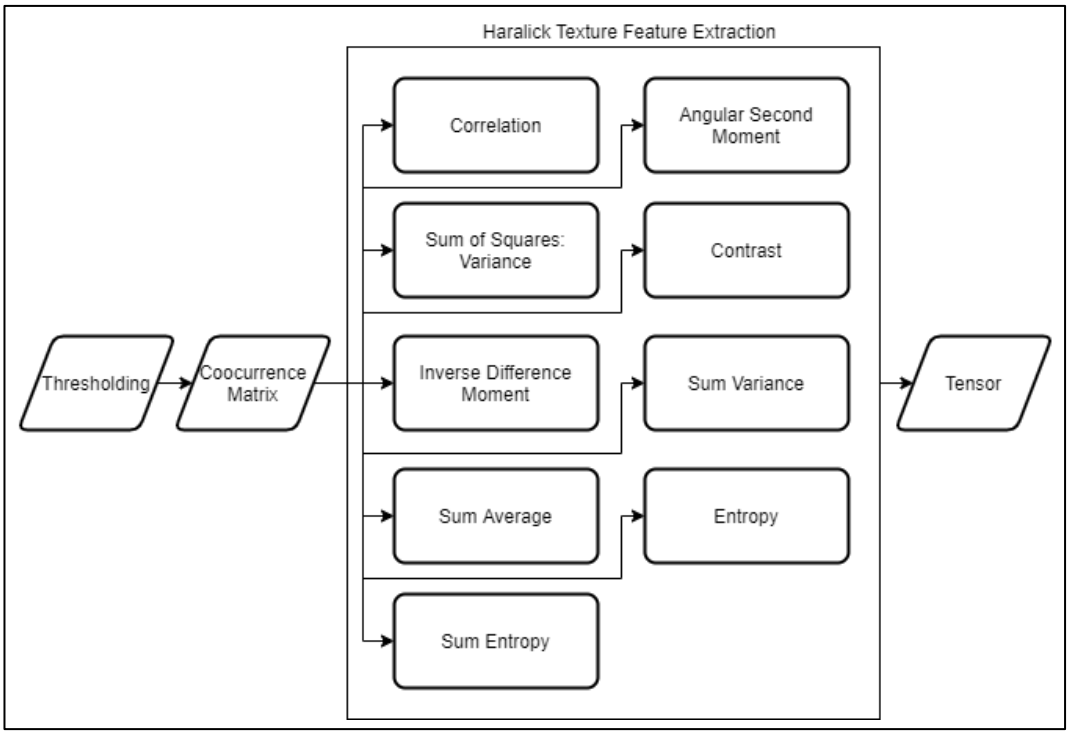


Gambar 3.2. Tahapan Pra-proses Data

Pada tahapan ini, nama kelas dari setiap masukan di ekstraksi dan disimpan untuk proses pembangunan model. Lalu masukan citra dikonversi warnanya dari rgb ke skalar keabuan. Hasil citra yang warnanya telah diubah di filter menggunakan filter median untuk mengurangi jumlah noise yang ada di dalam gambar. Citra yang telah difilter, dideteksi tepinya yang kemudian filter dengan filter gaussian lalu difilter lagi menggunakan ambang batas otsu. Citra yang telah di filter dengan ambang batas otsu kemudian dibagi dengan persentasi 90 persen untuk data *training* dan 10 persen untuk data test yang kemudian akan digunakan pada proses ekstraksi ciri tekstur dengan menggunakan metode GLCM.

3.2. Ekstraksi Ciri Tekstur Haralick

Ektraksi ciri tekstur menggunakan metode GLCM pada penelitian ini dihitung dengan arah 0^o, 45^o, 90^o, 135^o dan dengan jarak 1 pixel pada matriks coocurrencenya. Dengan menggunakan matriks coocurrence dihitung 9 ciri tekstur untuk setiap gambar yang disimpan di dalam sebuah matrix tensor untuk digunakan dalam proses pembangunan model maupun proses klasifikasi. Proses ekstraksi ciri menggunakan GLCM dapat dilihat pada gambar 3.3.



Gambar 3.3. Proses Ekstraksi Ciri Tekstur

3.3. Data Splitting

Seluruh data yang di dapatkan dari proses ekstraksi ciri dibagi menjadi 2 bagian yaitu data train, dan data test dengan menggunakan library dari Scikit-learn. Data train digunakan untuk melatih model dengan harapan model menyesuaikan parameternya, sedangkan data test digunakan untuk mengukur tingkat akurasi dari model dengan menggunakan data yang belum pernah dilihat oleh model.

3.4. Pembangunan Model

Pembangunan model RNN dibagi menjadi 3 yaitu pelatihan model, pengujian model, dan optimasi parameter. Optimasi parameter menggunakan library Skorch. Data *training* masuk ke dalam kumpulan unit kecil dari RNN yang bernama *RNN Cell*.

Model dibangun dengan menggunakan nilai batch sebesar 64, epoch dengan nilai 100, step dengan nilai pembulatan dari pembagian jumlah data *training* dengan nilai batch yaitu 54, input sebanyak 4, jumlah neuron terbaik hasil hyperparameter yang bernilai 40 dan output sebanyak 4. Data *training* dan kelas dibentuk menjadi sebuah array tensor 3 dimensi. Data *training* memiliki bentuk 9x1 yang isinya berbentuk 1x4, 9 data ciri haralick yang masing-masing ciri memiliki 4 data dari setiap derajat *coocurrence matrix*.

3.5. Evaluasi Model

Evaluasi model didapatkan dengan pengujian menggunakan data uji yang dimasukkan kedalam model yang telah dibangun. Model yang telah dilatih memberikan hasil yang memiliki nilai *true positive*, *false positive*, *true negative*, *false negative*. *True positive* berarti hasil klasifikasi menyatakan hasil positif, sesuai dengan hasil yang diketahui oleh penguji yaitu positif. Sedangkan *false positive* memberikan hasil klasifikasi yang positif namun kenyataannya negatif, demikian juga dengan *true negative* dan *false negative*.

Tabel 3.1. Tabel Confusion Matrix

		Kelas Hasil Prediksi	
		Positif	Negatif
Kelas Asli	Positif	True Positive (TP)	False Negative (FN)
	Negatif	False Positive (FP)	True Negative (TN)

Keterangan:

- True Positive: Teridentifikasi secara benar
- True Negative: Tertolak secara benar

- False Positive: Teridentifikasi secara salah
- False Negative: Tertolak secara salah

Nilai presisi dari sebuah sistem klasifikasi memanfaatkan nilai *true positive* dan *false positive*, sedangkan nilai *recall* memanfaatkan nilai *true positive* dan *false negative*. Rumus presisi, recall dan f-score sebagai berikut (Koehrsen, 2018), (DeepAI.org, n.d).

$$\text{Akurasi} = \frac{True\ Positive + True\ Negative}{True\ Positive + True\ Negative + False\ Positive + False\ Negative} \quad \ldots(3.1)$$

$$\text{Presisi} = \frac{True\ Positive}{True\ Positive + False\ Positive} \quad \ldots(3.2)$$

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative} \quad \ldots(3.3)$$

$$F1Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad \ldots(3.4)$$

Akurasi menunjukan kemampuan model untuk mengklasifikasikan sebuah kelas secara umum sedangkan presisi dan *recall* menilai model secara detil karena memfokuskan perhitungan terhadap hasil prediksi untuk setiap kelas dan tidak menggunakan nilai *True Negative* yang dihasilkan dari hasil identifikasi kelas lain. *f1-score* menurut scikit-learn.org dan deepai.org merupakan sebuah nilai yang didapatkan dari perhitungan rata-rata harmonik antara presisi dan *recall*. Semakin besar nilai presisi dan nilai recall, semakin besar *f1-score* dari sistem klasifikasi tersebut.

3.6. Implementasi

Implementasi dilakukan untuk mengidentifikasi daun jagung yang tidak terdapat di dalam dataset. Bahasa pemrograman yang digunakan adalah Python 3 dengan menggunakan Jupyter Notebook dan PyCharm.