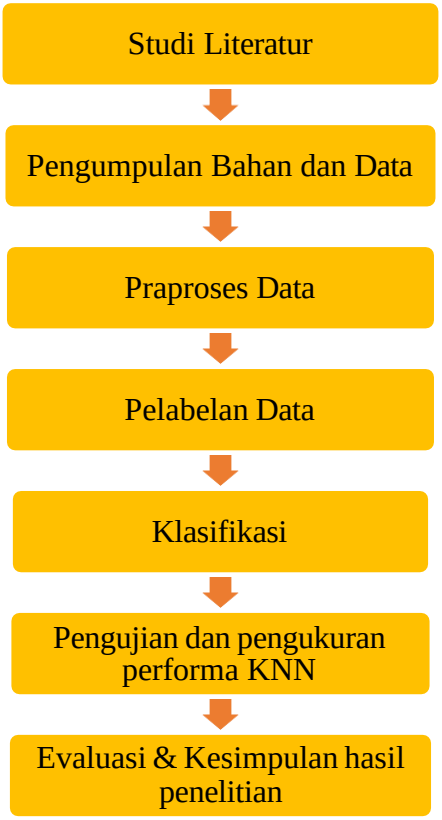
# BAB III

# METODOLOGI PENELITIAN

## 3.1 Metodologi Penelitian

Tahapan penelitian digambarkan secara lengkap dengan menggunakan *flow chart* menggunakan metode penelitian kuantitatif. *Flow chart* atau diagram alur digunakan untuk memudahkan penyampaian informasi terkait langkah-langkah yang akan dilakukan dalam penelitian ini. Tahapan penelitian secara keseluruhan disajikan pada gambar 3.1.




Gambar 3. 1 Tahapan Penelitian

## 3.2 Tahapan Penelitian

### 3.2.1 Pengumpulan Bahan dan Data

Data yang dibutuhkan untuk penelitian ini adalah gambar ikan gabus hias yang memiliki varian dan warna berbeda. Data ikan gabus hias akan diekstrak sehingga menghasilkan dataset yang dapat digunakan untuk proses klasifikasi dan memiliki kualitas data yang baik.

### 3.2.2 Praproses Data

1) Penghapusan Latar Belakang Gambar

Penghapusan latar belakang gambar ikan gabus hias dilakukan dengan menggunakan aplikasi pihak ketiga yaitu *background remover*. Proses penghapusan latar belakang gambar dimaksudkan agar mempermudah proses ekstraksi dan klasifikasi. Gambar yang diolah untuk proses ekstraksi adalah gambar objek utama saja berupa objek gambar ikan. Gambar yang akan diproses diupload ke dalam aplikasi. Proses penghapusan latar belakang dilakukan dengan menyeleksi objek utama dengan latar belakang.

2) Proses *Grayscaling*

Proses *grayscaling* merupakan proses untuk mengkonversi citra RGB menjadi citra *grayscale.* Proses tersebut dilakukan agar dapat dihitung nilai *contrast, homogeneity, ASM, energy* dan *correlation* masing–masing gambar. Citra *grayscale* merupakan citra yang nilai intensitas pikselnya didasarkan pada derajat keabuan. Pada citra *grayscale* 8-bit, derajat warna hitam sampai dengan

putih dibagi ke dalam 256 derajat keabuan di mana warna putih sempurna direpresentasikan dengan nilai 255 dan hitam sempurna dengan nilai 0. Persamaan yang umumnya digunakan untuk mengkonversi citra RGB truecolor 24-bit menjadi citra *grayscale* 8-bit adalah 0.2989*R+0.5870*G+0.1140*B,

3) Proses *Gray-Level-Cooccurance Matrix*

Praproses data dilakukan untuk mengolah data mentah dari setiap gambar agar dapat diproses pada tahap klasifikasi. Pada proses ini diperlukan data yang baik yaitu data yang valid dan tidak rusak. Untuk mendapatkan data yang valid dengan langkah pertama yaitu mengekstraksi citra dengan algoritma GLCM dan menyimpannya ke dalam sebuah file berekstensi *.csv. Praproses data juga memuat proses pembersihan data yang tidak sesuai dengan data yang dibutuhkan. Pembersihan dapat dilakukan pada pengurangan fitur sesuai kebutuhan. Berikut langkah pembuatan matrix GLCM.

a) Pembuatan *framework* matriks GLCM.
b) Mengisi *framework* matriks untuk pembuatan *co-occurance matrix.*
c) Penjumlahan antara *co-occurance matrix* dengan matriks hasil *transpose.*
d) Proses normalisasi matriks agar menghasilkan nilai antara 0–1.

Langkah pertama adalah mendefinisikan matriks *framework* bernilai 0 dengan dimensi sesuai dengan *quantization level* yaitu 4x4 dan memakai perhitungan rumus 3.1, Berikut adalah tabel 3.1 matriks GLCM yang merupakan *quantization level* dan nilai *gray tone* 0-4.

Tabel 3. 1 Matriks GLCM dengan *Quantization Level 4*

| | | |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 2 | 3 |
| 2 | 3 | 2 |

*Quantization level = count(gray tone)* **(3.1)**

Sehingga nilai *quantization level* dari *gray tone* 0-3 adalah 4. Berikut *framework* dengan size 4x4 yang dapat dilihat pada tabel 3.3.

Tabel 3. 2 *Framework* Matriks 4x4

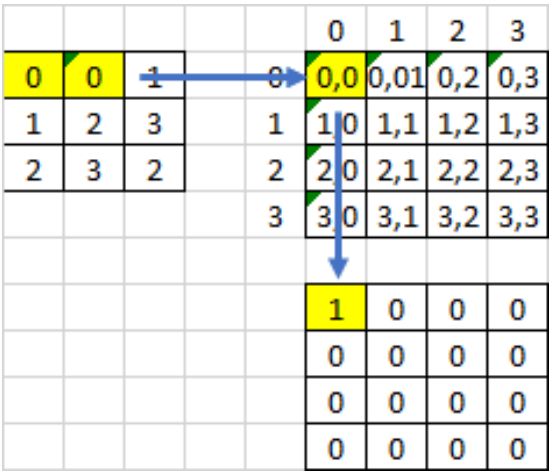
| | | | |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 |

Tiap posisi dari matriks *framework* tersebut adalah kombinasi nilai pixel pada matrix *gray tone* 3x4 yang dapat dilihat pada tabel 3.3.

Tabel 3. 3 Kombinasi Nilai Pixel 3x3

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 0,0 | 0,1 | 0,2 | 0,3 |
| 1 | 1,0 | 1,1 | 1,2 | 1,3 |
| 2 | 2,0 | 2,1 | 2,2 | 2,3 |
| 3 | 3,0 | 3,1 | 3,2 | 3,3 |

*Co-occurrence matrix* dibentuk dengan melakukan kombinasi nilai *pixel* yang berpasangan yang dipilih mulai dari kiri-atas sampai kanan-bawah. Lalu tambahkan nilai 1 pada lokasi yang sesuai dengan lokasi *pixel* pada contoh matriks seperti yang terlihat pada gambar 3.2.




Gambar 3. 2 Pengisian *Framework* Matriks

Setelah melakukan perulangan penambahan nilai satu untuk semua kombinasi pasangan pixel pada matrix input 3x3. Maka matriks yang sebelumnya 3x3 menjadi matriks 4x4. Hasilnya akan sesuai dengan pasangan pixel yang ada matriks input. Dibuat menjadi *symmetric matrix* dengan cara menjumlahkan hasil GLCM matriks dan *transpose* matriksnya. Sehingga hasil dari penjumlahan antara hasil GLCM matriks dengan transpose GLCM matriks.

Proses GLCM terdiri dari beberapa tahapan yaitu penjumlahan antara matriks GLCM dan transpose-nya kemudian hasil tersebut dinormalisasi dengan menggunakan rumus 3.2

$$glcmNorm = \frac{glcmValue}{\sum_i^N glcmValue} \quad \textbf{(3.2)}$$

Tujuan akhir dari proses GLCM adalah untuk memperoleh nilai tekstur diantaranya adalah nilai *contrast, dissimilarity, homogeneity, ASM, energi* dan *correlation.* Berikut adalah persamaan untuk menghitung nilai textur tersebut

$$Contrast = \sum_{i,j=0}^{levels-1} P_{i,j}(i-j)^2 \quad \textbf{(3.3)}$$

$$Dissimilarity = \sum_{i,j=0}^{levels-1} P_{i,j}|i-j| \quad \textbf{(3.4)}$$

$$Homogeneity = \sum_{i,j=0}^{levels-1} \frac{P_{i,j}}{1+(i-j)^2} \quad \textbf{(3.5)}$$

$$ASM = \sum_{i,j=0}^{levels-1} P_{i,j}^{\ 2} \quad \textbf{(3.6)}$$

$$Energy = \sqrt{ASM} \quad \textbf{(3.7)}$$

$$Correlation = \sum_{i,j=0}^{levels-1} Pi,j\left[\frac{(i-\mu_i)(j-\mu_j)}{\sqrt{(\sigma_i^2)(\sigma_j^2)}}\right] \quad \textbf{(3.8)}$$

Keterangan:

- *i,j* merupakan koordinat pixel pada matriks GLCM,
- *levels* merupakan rentang *gray tone*, pada citra digital 0–255 (level=256),
- Pi,j merupakan nilai *pixel* pada koordinat i,j GLCM matriks

4) Proses Pengecekan Data Rusak

Pengecekan data merupakan salah satu praproses data untuk menemukan dan menganalisa kembali data yang rusak karena data kosong atau tidak lengkap. Kerusakan data diakibatkan adanya pengiriman data yang tidak lengkap pada saat pemroresan ataupun transfer data. Kerusakan juga dapat diakibatkan oleh sistem dan masalah lainnya. Data yang mengalami kerusakan ataupun memiliki *infinity* harus diperbaiki dengan cara mengganti dengan nilai tengah pada satu kolom ataupun dengan dihilangkan agar proses selanjutnya dapat dilakukan.

### 3.2.3 Pelabelan Data

Hasil ekstraksi dengan algoritma ekstraksi citra akan menghasilkan sebuah dataset yang memiliki berbagai fitur namun belum secara otomatis diklasifikasikan ke dalam sebuah label. Setiap kelas data akan diberi label sesuai dengan ciri yang ada pada setiap data. Label ini menjadi acuan untuk proses pembagian kelas klasifikasi.

### 3.2.4 Klasifikasi KNN dan Weight-KNN

Data latih yang telah dikumpulkan akan digunakan untuk menentukan hasil klasifikasi dari data baru yang dimasukan. Klasifikasi dilakukan dengan menggunakan dua algoritma yaitu KNN dan *Weight*-KNN.

1) Normalisasi data

Klasifikasi memerlukan data yang memiliki rentang nilai kecil antara satu data dengan data lainnya. Oleh karena hal tersebut, diperlukan proses normalisasi data dengan metode *Z-Transformation* yaitu Persamaan *Z-transformation* dapat dilihat pada persamaan 3.9.

$$\text{Normalisasi}(xi) = \frac{xi - mean(x)}{stdev(x)} \quad \textbf{(3.9)}$$

Dimana $x_i$ merupakan nilai data ke-i pada dataset, *mean* merumapkan nilai rata-rata dari data yang ada pada dataset dan *stdev* adalah nilai standar deviasi dari data yang ada pada dataset. Perumusan perhitungan untuk menghitung nilai rata – rata fitur yang merupakan proses pertamanya dapat dilihat pada persamaan 3.10.

$$\text{Mean}(x) = \frac{\sum_{i=1}^{n} x_i}{n} \quad \textbf{(3.10)}$$

Hasil dari perhitungan nilai rata–rata akan digunakan pada proses selanjutnya. Proses berikutnya adalah menghitung deviasi standar dari fitur *total_fwd_packet*. Perumusan perhitungan untuk nilai standar deviasi dari data yang ada pada dataset setelah diperoleh nilai rata-rata dapat dilihat pada persamaan 3.11.

$$\text{Stdev}(x) = \frac{\sqrt{\sum_{i=1}^{n} (x_i - \bar{x})^2}}{n-1} \quad \textbf{(3.11)}$$

2) Klasifikasi dengan KNN

Klasifikasi KNN dilakukan dengan mengelompokan sejumlah data dan diurutkan berdasarkan tetangga terdekat dengan dibatasi oleh nilai K. Berikut proses klasifikasi KNN dapat dilakukan dengan langkah–langkah sebagai.

a) Menentukan nilai k misalnya k = 3 untuk mengelompokan data berdasarkan 3 tetangga terdekat.
b) Menghitung nilai *euclidean* data uji terhadap data latih. Perhitungan untuk nilai *euclidean* pada data uji ke–1 menggunakan rumus 3.12.
c) Setiap nilai jarak yang diperoleh diurutkan dari yang terkecil ke yang terbesar untuk menentukan nilai prediksi atau label yang sesuai.

$$dxy = \sqrt{\sum_{i=1}^{m} (x_i - y_i)^2} \quad \textbf{(3.12)}$$

d) Data latih dan data uji dimasukan ke dalam model dengan menggunakan operator *retrieve*. Data tersebut ditentukan fitur yang menjadi label dengan operator *set role*,

e) Proses normalisasi data dilakukan dengan bantuan operator *normalize*. Hasil normalisasi kedua data tersebut dapat diproses untuk proses klasifikasi.

f) Klasifikasi menggunakan algoritma KNN dengan nilai K = 5. Hasil performa dari proses klasifikasi dihitung dengan operator *performance*

3) Klasifikasi dengan *Weight*-KNN

*Weight-KNN* merupakan klasifikasi yang diadaptasi dari KNN yang memiliki perbedaan pada pemilihan label atau kelas hasil klasifikasi. Penentuan kelas di dalam KNN didasari pada banyaknya label yang sama pada hasil akhir klasifikasi berdasarkan nilai K yang ditentukan dapat dilihat pada persamaan 3.13.

$$Weight = \frac{1}{d} \quad \textbf{(3.13)}$$

Keterangan:

- *Weight* adalah nilai bobot data yang akan dihitung.
- D adalah jarak data yang dihitung sebelumnya.

Sama halnya dengan proses KNN secara manual, pada *weight-KNN* juga tidak dapat dilakukan perhitungan satu persatu dikarenakan data yang cukup banyak. Pemodelan klasifikasi *weight-KNN* sama dengan pemodelan klasifikasi KNN, namun ada perbedaan diantara keduanya yaitu pada bagian parameter untuk klasifikasi *weight-KNN* menggunakan *weighted vote*. *Weighted vote* merupakan proses pemilihan label data dengan memperhatikan *weight* setiap jarak data.

### 3.2.5 Pengujian dan Pengukuran Performa KNN dan Weight-KNN

Hasil data yang telah diseleksi akan dilakukan pelatihan dan pengujian dengan menggunakan algoritma KNN untuk melihat nilai performa dari dataset yang digunakan. Hasil pengujian dan pengukuran tersebut dijadikan untuk dilakukan perbandingan performa algoritma KNN dan *Weight*-KNN

1) Pengujian dengan nilai K

Nilai maksimal akurasi pada KNN dipengaruhi oleh nilai K yang dipilih untuk menentukan kelas data pada label. Rentang nilai K yang diujikan pada model yang telah diseleksi fitur dalam penelitian ini adalah nilai ganjil dari 1–27. Nilai ganjil diperlukan untuk menghindari adanya jumlah label yang sama dari hasil uji. Hasil klasifikasi terbaik ditentukan berdasarkan hasil nilai akurasi terbesar dari setiap nilai K yang diujikan.

2) Pengujian dengan nilai K pada KNN

Pengujian nilai K pada KNN memiliki proses yang sama dengan klasifikasi KNN sebelumnya, perbedaannya hanya pada proses klasifikasi yang dilakukan berulang–ulang sesuai dengan range nilai K yang ditentukan pada operator perulangan. Di dalam operator perulangan terdapat proses KNN sehingga performa setiap nilai K dapat dihitung akurasinya.

3) Pengujian dengan nilai K pada *Weight*-KNN

Sama halnya dengan algoritma KNN pada *Weight-KNN* juga diperlukan penentuan nilai K terbaik berdasarkan dari proses klasifikasi. Hasil akurasi yang didapatkan akan dijadikan acuan untuk melakukan perbandingan proses klasifikasi KNN.

### 3.2.6 Evaluasi

Evaluasi performa terbaik dilakukan dengan menggunakan metode *Confussion Matrix* untuk menghitung nilai akurasi, presisi, *recall* dan *f-1 score* dari model yang akan dievaluasi yaitu nilai K terbaik dari algoritma *weight-KNN*. Berikut beberapa rumus c*onfusion matrix* untuk menghitung *accuracy, precision, recall,* dan *f1 – score* menurut Han, Kamber dan Pei, 2012.

$$\text{Akurasi} = \frac{TP+TN}{TP+TN+FP+FN} x\ 100\% \quad \textbf{(3.14)}$$

Akurasi berfungsi untuk menghitung jumlah ukuran proporsi dari klasifikasi yang benar dalam model dengan menggunakan persamaan 3.14.

$$\text{Presisi} = \frac{TP}{TP+FP} x\ 100\% \quad \textbf{(3.15)}$$

Presisi berfungsi untuk mengetahui perbandingan jumlah klasifikasi yang benar dengan yang salah dengan menggunakan rumus pada persamaan 3.15.

$$\textit{Recall} = \frac{TP}{TP+FN} x\ 100\% \quad \textbf{(3.16)}$$

*Recall* berfungsi untuk menghitung jumlah klasifikasi yang benar dengan melakukan perbandingan terhadap jumlah entri yang terlewat dengan menggunakan rumus pada persamaan 3.16.

$$\textit{F-1 score} = 2\ x\ \frac{Presisi\ x\ Recall}{Presisi+Recall} x\ 100\% \quad \textbf{(3.17)}$$

*F-1 score* berfungsi untuk menghitung rata – rata dari presisi dan *recall* sehingga dapat diketahui efektivitas dari model yang dibuat dengan menggunakan rumus pada persamaan 3.17. Hasil penelitian akan disimpulkan dengan memperhatikan hasil pengukuran algoritma KKN dengan fitur lengkap dan fitur terpilih setelah dilakukan seleksi fitur.