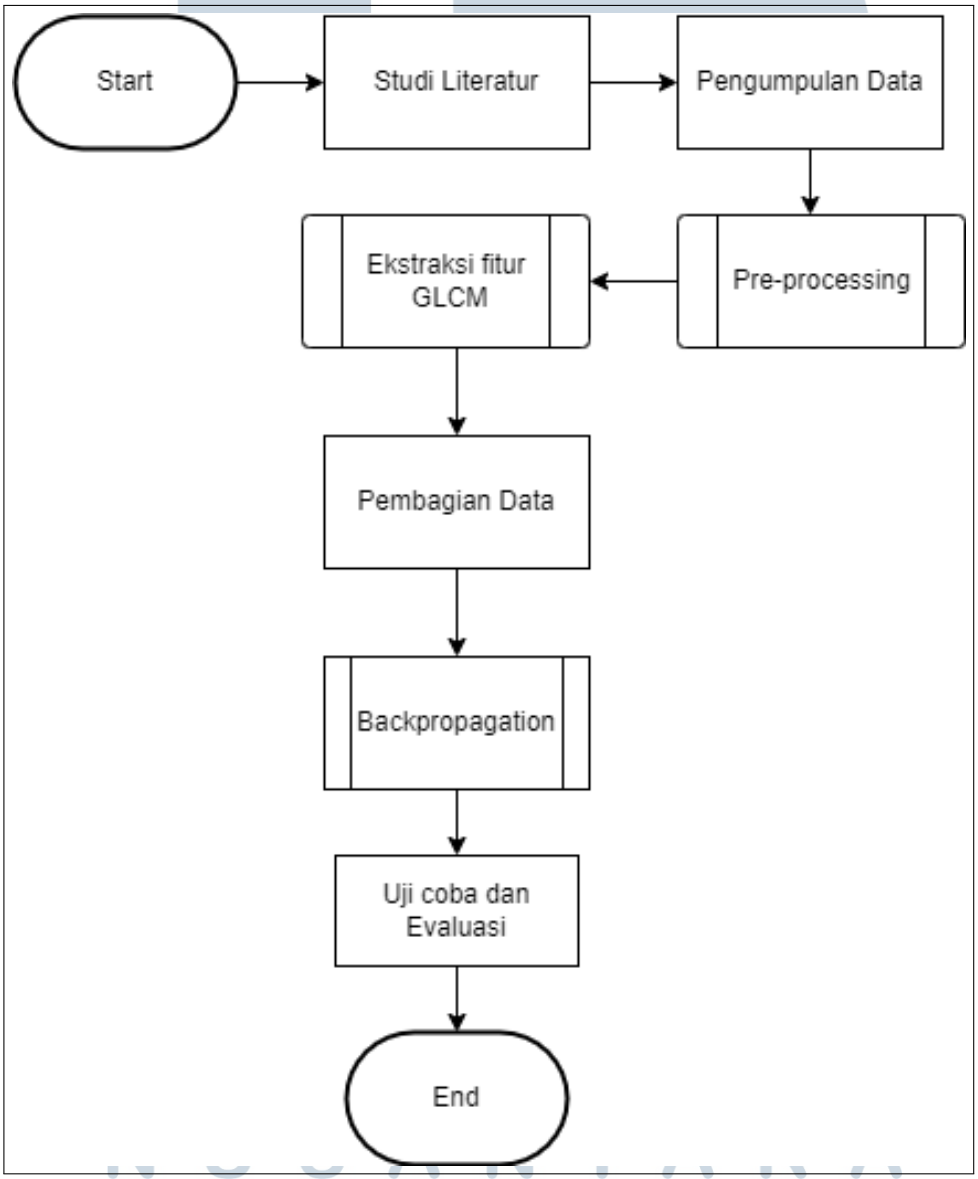
BAB 3
METODOLOGI PENELITIAN

3.1 Metodologi Penelitian

Metode penelitian yang digunakan dalam penelitian ini melewati beberapa tahap pelaksanaan yang secara garis besar ditunjukkan pada Gambar 3.1.



Gambar 3.1. Tahapan Metodologi Penelitian

1. Studi Literatur

   Dalam tahap ini, dilakukan pencarian dan pembelajaran terhadap pengetahuan mengenai metode-metode, teori, dan persamaan yang dibutuhkan, terutama mengenai GLCM dan jaringan saraf tiruan *Backpropagation*. Sumber pembelajaran yang digunakan adalah jurnal, buku, paper, artikel, tesis, *website* dan referensi lain yang terkait penelitian ini.

2. Pengumpulan Data

   Dalam tahap ini, dilakukan pengumpulan data berupa *folder* yang berisi kumpulan citra wajah. Dataset citra wajah yang digunakan bersumber dari kaggle.com yaitu *ORL dataset* yang terdiri atas 410 citra wajah yang berasal dari 41 orang dengan masing-masing 10 citra wajah. Data yang diperoleh dari dataset berupa data berwarna (RGB) dengan format *file* JPG yang memiliki ukuran 80x70 piksel dan tiap *file* berukuran 2,56 KB.

3. Pre-processing

   Tahap *pre-processing* untuk pengolahan data mentah sebelum masuk ke proses selanjutnya yaitu ekstraksi fitur. *Pre-processing* yang dilakukan adalah *background removal* dan *grayscaling*.

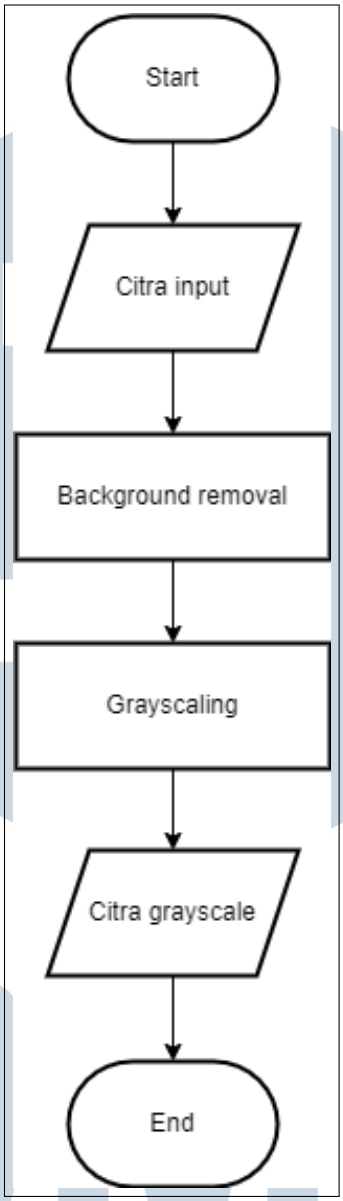
   - *Background removal*

     Background removal untuk menghilangkan *background* pada citra sehingga hanya bagian wajah saja yang diproses dan mengurangi *noise* pada *background* agar sistem dapat fokus pada area yang diperlukan saja. *Background removal* dilakukan dengan menggunakan pihak ketiga yaitu remove.bg.

   - *Grayscaling*

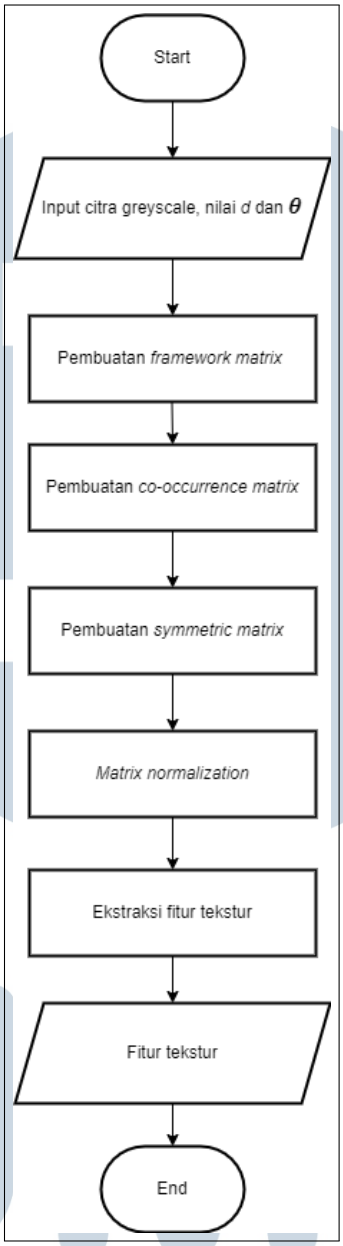
     Proses *grayscaling* dengan mengubah citra wajah berwarna(RGB) menjadi citra yang memiliki tingkat warna abu-abu(*grayscale*) yang digunakan sebagai citra input pada ekstraksi fitur GLCM.

   Tahap *pre-processing* pada penelitian ini ditunjukkan pada Gambar 3.2.



Gambar 3.2. Flowchart Pre-processing

4. Ekstraksi Fitur GLCM
   Setelah proses *pre-processing*, citra wajah diekstraksi fitur tekturnya menggunakan GLCM. Proses ekstraksi fitur GLCM ditunjukkan dalam Gambar 3.3.



Gambar 3.3. Flowchart Ekstraksi fitur GLCM

Ekstraksi fitur GLCM menggunakan *library skimage* yang dibagi menjadi 2 tahap yaitu pembentukan *matrix* GLCM menggunakan fungsi greycomatrix dan perhitungan fitur tekstur dari matrix GLCM menggunakan fungsi greycoprops. Tahapan ekstraksi fitur GLCM adalah sebagai berikut:

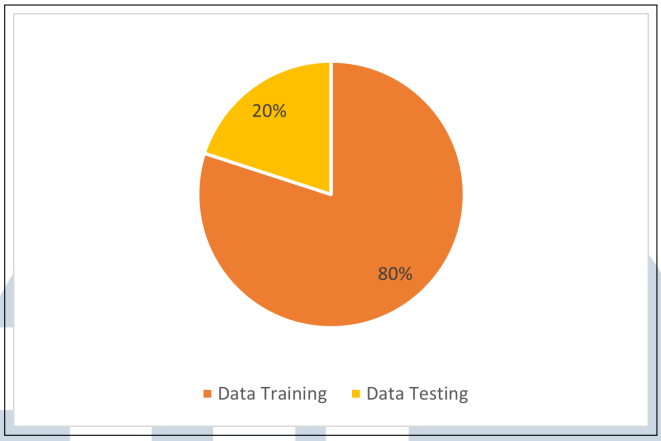
- Dalam proses ekstraksi fitur GLCM diperlukan *input* berupa citra *grayscale* dan menentukan jarak(d) dan sudut(θ) yang digunakan.
- Pembuatan *framework matrix* dengan dimensi sesuai dengan jumlah

quantization level yaitu jumlah *gray tone* pada citra *grayscale*, dan tiap posisi pada *framework matrix* merupakan kombinasi nilai piksel pada *matrix input*.

- Pembuatan *co-occurrence matrix* atau lebih tepatnya mengisi *framework matrix* dengan menghitung jumlah *co-occurrence* antara piksel referensi dan piksel tetangga berdasarkan sudut dan jarak yang telah ditentukan.
- Pembuatan *symmetric matrix* dengan menjumlahkan *co-occurrence matrix* dengan hasil transposnya.
- Tahap selanjutnya, *symmetric matrix* dinormalisasi dengan menghitung probabilitas setiap elemen *matrix*.
- Selanjutnya dari matriks GLCM yang sudah dinormalisasi dapat diperoleh fitur teksturnya. Fitur yang digunakan adalah 7 fitur utama GLCM, yaitu: ASM, *contrast, correlation, homogeneity, dissimiliarity, entropy,* dan *autocorrelation*.

5. Pembagian Data

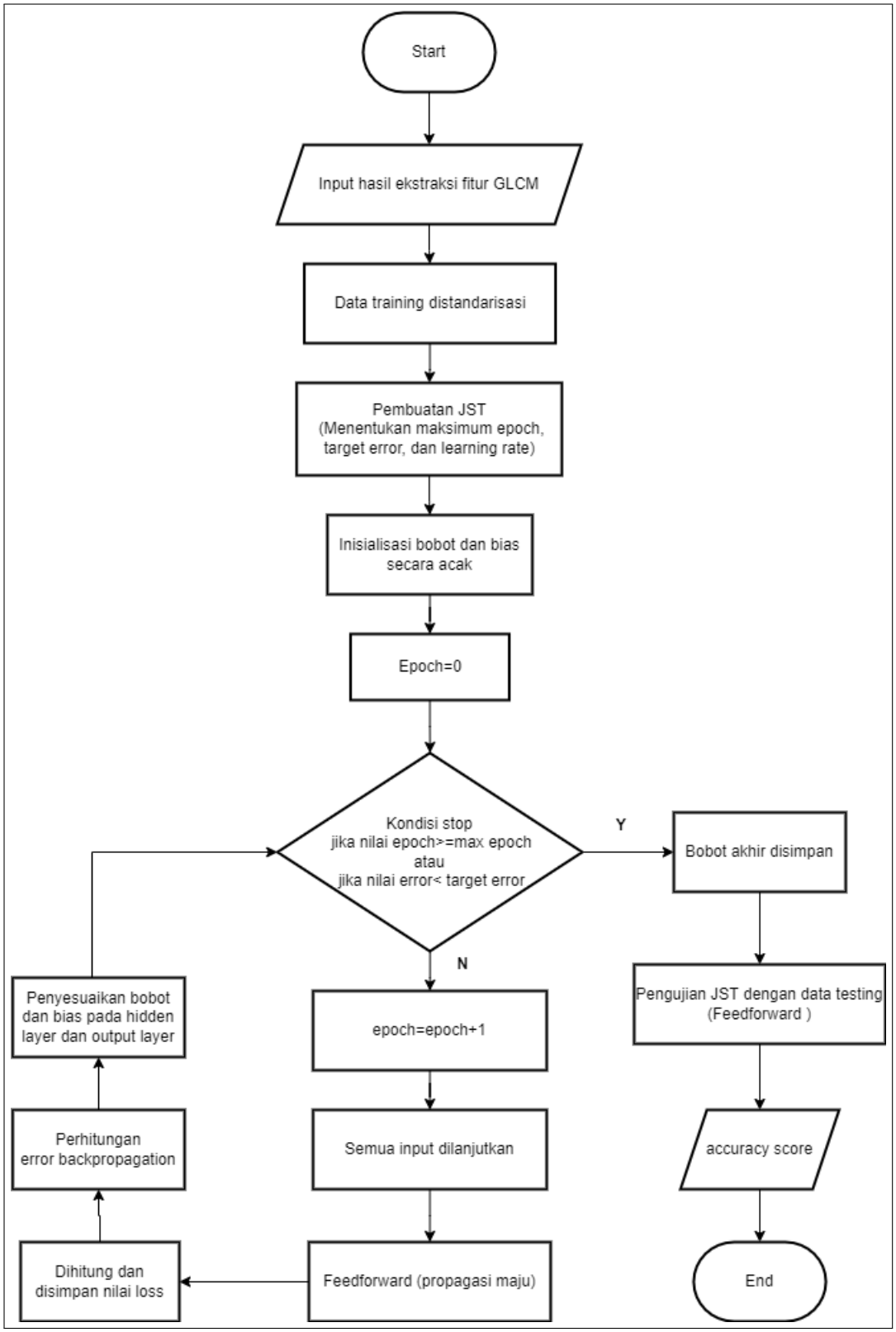
   Data hasil ekstraksi fitur GLCM sebelumnya dibagi manjadi data pelatihan (*training*) dan data pengujian(*testing*). Data pelatihan digunakan untuk melatih algoritma untuk menemukan model yang sesuai, sedangkan data pengujian digunakan untuk mengetahui kinerja algoritma yang telah dilatih sebelumnya saat menemukan data baru. Sesuai dengan teori Pareto menyatakan bahwa banyak kejadian sekitar 80% daripada efeknya disebabkan oleh 20% dari penyebabnya. Sehingga untuk melatih model sebaik mungkin maka rasio partisi data pada *training* harus lebih besar daripada rasio partisi data *testing*. Pada penelitian ini menggunakan rasio pembagian data 80:20, yaitu 80% sebagai data *training* dan sisanya 20% sebagai data *testing*, yang diilustrasikan pada Gambar 3.4.



Gambar 3.4. Perbandingan Data Training dan Data Testing

6. Jaringan Saraf Tiruan(JST) Backpropagation

   Implementasi JST *Backpropagation* menggunakan *class* MLP Classifier yang disediakan dalam library sklearn. Multilayer Perceptron (MLP) adalah salah satu model JST dengan bobot acak dari pelatihan *Backpropagation*. Pelatihan ini melakukan optimasi dengan menggunakan algoritma genetika sebagai pelatihan untuk memperoleh bobot yang lebih baik. Selain itu, MLP mampu mengenali suatu pola berdasarkan pengetahuan yang diperoleh saat proses pelatihan yang disebut *Backpropagation*. Setiap kali MLP melakukan proses pelatihan, maka bobot-bobot jaringan yang menghubungkan *neuron* pada tiap *layer* diperbarui.



Gambar 3.5. Flowchart Backpropagation

Pada Gambar 3.5 menunjukkan proses *training* dan *testing* dengan *Backpropagation*, berikut ini merupakan tahapan proses tersebut:

- Sebelum data *training* dilatih dengan *Backpropagation*, perlu dilakukan standarisasi sehingga semua data dapat dievaluasi secara seragam. Standarisasi hanya dilakukan pada data *training* dan bukan pada data *testing*. Hal ini dikarenakan dalam dunia nyata, data tidak diskalakan dan tujuan akhir dari jaringan saraf tiruan adalah untuk membuat prediksi pada data dunia nyata.
- Pada tahap *training*, dibuat jaringan saraf tiruan (JST) terlebih dahulu menggunakan *class* MLP Classifier dengan menentukan nilai parameter yang digunakan seperti *hidden_layer_sizes, activation, solver, learning_rate, learning_rate_init,* dan *max_iter*-nya. Kemudian *class* MLP Classifier menginisialisasi bobot dan *bias* secara acak serta memilih secara acak data *training* sebagai *input*.
- Kemudian dilakukan *feed-forward* atau komputasi maju untuk menghitung nilai aktivasi yang ada pada semua *neuron* baik yang ada di *hidden layer* ataupun *output layer*.
- Sinyal ouput yang dihasilkan saat *feed-forward* dicocokkan dengan target yang diharapkan dengan melakukan perhitungan selisih antara target dengan sinyal keluaran, dan disimpan nilai *loss*-nya.
- Setelah itu dilakukan *backward propagate* atau komputasi mundur dengan menggunakan nilai *loss* yang telah disimpan untuk menyesuaikan bobot hubungan antara *output layer* dengan semua *neuron* yang berada pada *hidden layer*. Selanjutnya, sinyal kesalahan atau *loss* dikirmkan ke dalam *hidden layer* sehingga tiap *neuron* pada *hidden layer* dapat menyesuaikan dengan nilai *output* yang memiliki nilai mendekati target.
- Setelah tahap *backward propagate*, dilakukan penyesuaian bobot(*weight*) dan *bias* pada lapisan tersembunyi(*hidden layer*) dan lapisan keluaran(*output layer*).
- Kondisi penghentian pada proses *training* yang digunakan adalah jumlah iterasi ataupun kesalahan (*error*). Ketika iterasi yang dilakukan sudah melebihi jumlah maksimum iterasi yang ditentukan, atau kesalahan

yang terjadi sudah lebih kecil dari target *error* yang ditentukan, maka iterasi dihentikan.

- Setelah proses *training* selesai, *Backpropagation* dapat digunakan untuk proses pengujian(*testing*) jaringan. Pada tahap *testing*, proses yang dilakukan hanya sampai *feed-forward* atau tahap maju saja. Seluruh bobot input diambil dari nilai bobot terakhir yang diperoleh dari proses *training*. Pada tahap pengujian(*testing*), jaringan diharapan dapat mengenali pola berdasarkan data baru yang diberikan. Kemudian, hasil pengujian dibandingkan dengan label sebenarnya sehingga diperoleh *accuracy score*-nya.

7. Uji Coba dan Evaluasi

   Pada tahap ini, dilakukan pengujian terhadap hasil penelitian yang diperoleh dari hasil program yang telah dirancang sebelumnya. Untuk mengetahui tingkat kesuksesan uji coba maka diperlukan evaluasi menggunakan *Backpropagation* dengan *hidden node* dan *learning rate* berbeda untuk mengetahui tingkat akurasi. Setelah itu, dilakukan evaluasi untuk menilai performa dari hasil permodelan berdasarkan kriteria keberhasilan tertentu. Dalam melakukan evaluasi, digunakan *confusion matrix* untuk memperoleh nilai *recall, precision, accuracy,* dan *F1-score*. *Confusion matrix* dipilih karena merupakan metode yang populer digunakan untuk memvalidasi sejumlah *learning model* [28].