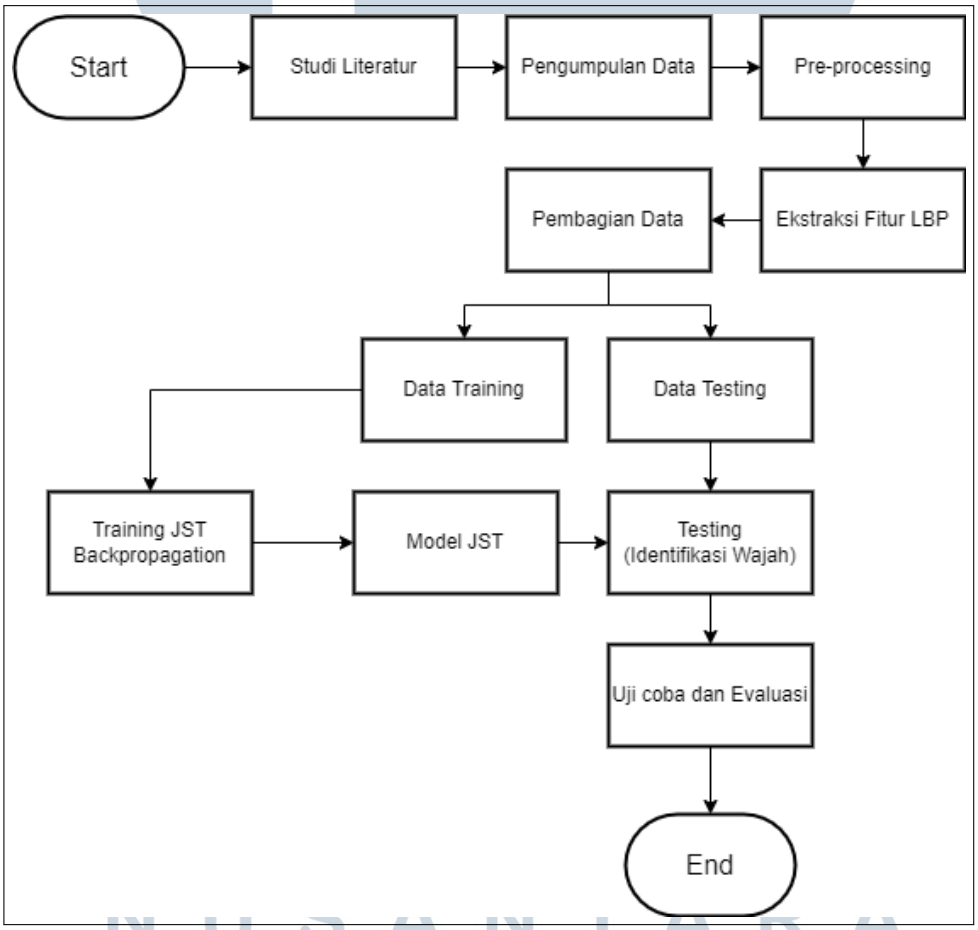
BAB 3
METODOLOGI PENELITIAN

3.1 Metodologi Penelitian

Metodologi yang digunakan dalam penelitian ini melewati beberapa tahap pelaksanaan agar metodologi dan perancangan sistem dapat berjalan dengan baik. Dalam proses perancangan yang dilakukan selama proses implementasi dan uji coba telah didukung oleh beragam sumber yang berguna untuk mempercepat proses dan memberikan hasil yang semakin baik. Tahap-tahap yang dilaksanakan antara lain sebagai berikut.



Gambar 3.1. Tahapan Metodologi Penelitian

1. Studi literatur

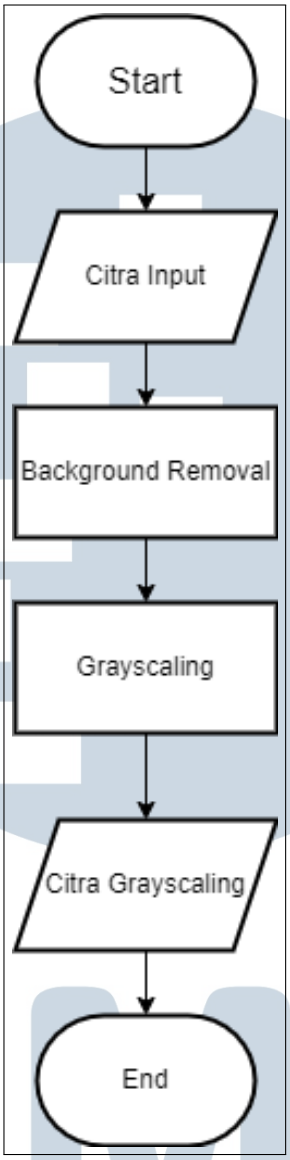
   Dalam tahap ini, dilakukan studi literatur untuk mengumpulkan data dan informasi terhadap teori dan konsep yang berhubungan dengan penelitian. Informasi yang dicari adalah mengenai LBP dan jaringan saraf tiruan *backpropagation*. Sumber pembelajaran yang digunakan seperti jurnal ilmiah, buku, internet, tesis, serta artikel-artikel terkait yang mendukung penelitian ini baik didapatkan dari konferensi, seminar ataupun ranah penelitian ilmiah sejenisnya yang membawa kepada arah ilmu pengetahuan yang dibutuhkan dalam proses penelitian ini.

2. Pengumpulan Data

   Pada tahap ini, dilakukan pengumpulan data yang berisi kumpulan *image* wajah, dengan menggunakan citra database yang sudah tersedia diambil dari *ORL Dataset* yang bersumber dari Kaggle.com terdiri atas 200 citra wajah yang berasal dari 20 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.

3. *Pre-processing*

   Preprocessing merupakan tahap awal dalam proses pengenalan objek dengan menggabungkan konsep citra digital dari data mentah untuk diolah sebelum dilakukan tahap ekstraksi fitur LBP. Tujuan dilakukan *preprocessing* agar dapat menghasilkan citra yang lebih baik, sehingga citra tersebut kemudian dapat dilanjutkan pada tahap ekstraksi fitur LBP. Proses *preprocessing* ditunjukan dalam Gambar 3.2.



Gambar 3.2. Flowchart Preprocessing

Pada Gambar 3.2 terdapat 2 (dua) tahapan *preprocessing* yang dilakukan. Berikut ini merupakan tahapan *preprocessing*:

(a) *Background Removal*

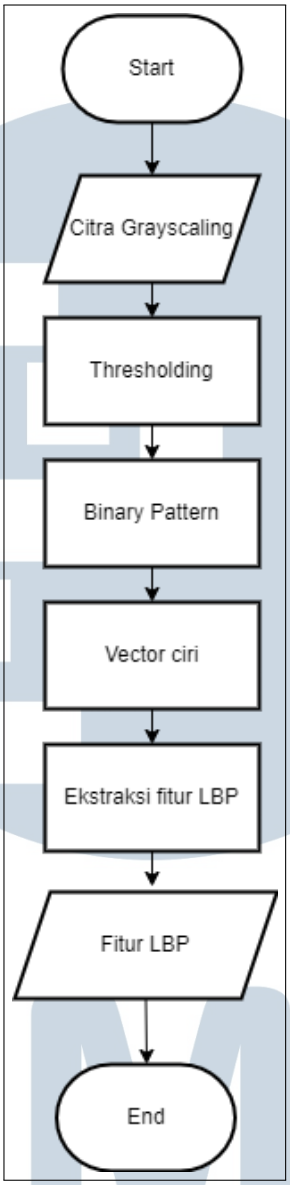
Melakukan *Background Removal* pada dataset wajah dengan menggunakan remove.bg. *Background Removal* digunakan untuk menghilangkan *background* sehingga hanya bagian wajah saja yang diproses, proses ini juga bertujuan untuk meminimalisir *noise* yang ada pada *background* sehingga sistem deteksi *object* dapat terfokus pada pengambilan area yang penting.

(b) *Grayscaling*

Setelah proses *Background Removal*, selanjutnya dilakukan proses *grayscaling* dengan tujuan agar citra dapat diproses pada tahap ekstraksi fitur. Proses *grayscaling* yaitu proses mengubah citra berwarna menjadi bentuk *grayscaling* atau tingkat keabuan dari 0-255. Konversi ruang warna citra dari RGB menjadi *grayscaling*, bertujuan untuk mempermudah dan mempercepat proses ekstraksi fitur *Local Binary Patterns*, hal tersebut dikarenakan ketika citra berada pada ruang warna *grayscaling* maka citra hanya terdiri dari satu lapisan layer warna. Selain itu, pada metode LBP diperlukan menggunakan citra *grayscaling*, sehingga citra RGB dikonversikan menjadi citra *grayscaling*.

4. Ekstraksi Fitur LBP

Setelah dilakukan tahap *pre-processing*, data yang telah diolah kemudian dilakukan proses untuk mengekstraksi fitur wajah menggunakan metode *Local Binary Pattern*. Proses ekstraksi fitur LBP ditunjukan dalam Gambar 3.3



Gambar 3.3. Flowchart Alur Ekstraksi Fitur LBP

Pada Gambar 3.3 citra hasil *preprocessing* diproses ke tahap selanjutnya yaitu proses ekstraksi fitur menggunakan metode LBP. Dalam proses perancangan, proses LBP menggunakan perhitungan Matlab dengan menetapkan parameter yang digunakan yaitu radius. Proses ekstraksi fitur dengan metode LBP dibagi menjadi dua tahap, yaitu pembentukan matriks LBP dan perhitungan nilai 6 fitur ciri LBP dalam ekstraksi fitur LBP, dengan tahapan yaitu:

(a) Citra yang diolah menggunakan citra hasil *preprocessing* berupa citra *grayscale*.

(b) Membagi matriks citra menjadi bagian yang lebih kecil atau disebut kernel yaitu matriks 3×3. Selanjutnya mencari nilai *threshold* pada piksel tengah dengan membandingkan nilai piksel pada titik pusat dengan nilai piksel pada 8 titik tetangga sekitarnya. Jika nilai piksel tetangga lebih besar atau sama dengan dari titik pusat maka diberi nilai 1, namun apabila nilai piksel tetangga lebih kecil dari titik pusat maka diberi nilai 0.

(c) Selanjutnya menyusun 8 nilai biner searah jarum jam, kemudian diubah kedalam nilai desimal untuk menggantikan nilai piksel pada pusat citra.

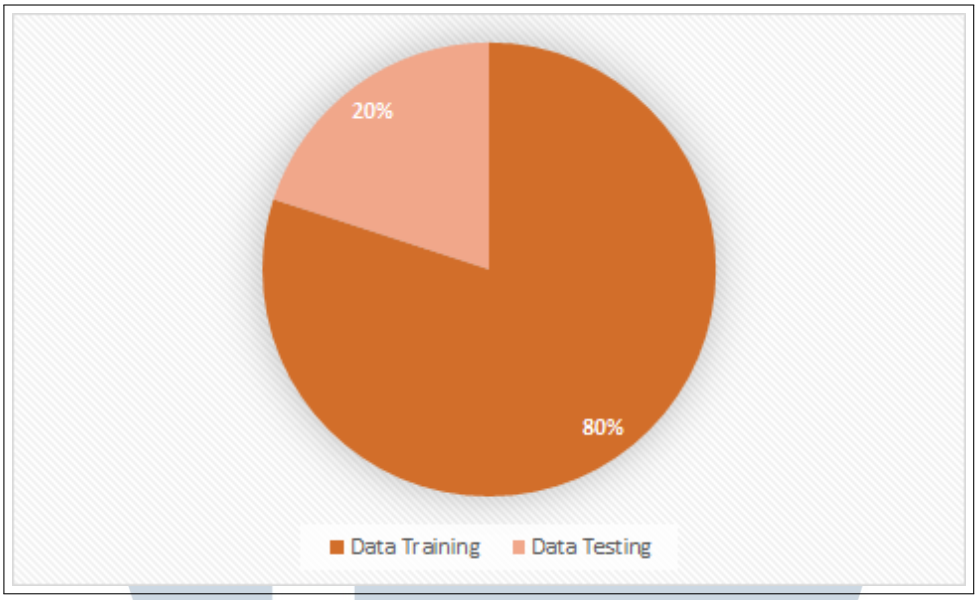
(d) Mengubah bilangan biner ke bilangan desimal dengan mengalikan nilai biner dengan bilangan eksponensial dari 2 Nilai desimal yang diperoleh menjadi nilai LBP pada satu kernel.

(e) Kemudian diperoleh hasil ekstraksi fitur berupa vektor ciri dari citra.

(f) Setelah mendapatkan hasil dari masing-masing nilai LBP, selanjutnya mencari nilai ciri dalam ekstraksi fitur LBP dengan menggunakan metode pengambilan ciri yang didasarkan pada karakteristik vektor citra. Dari nilai-nilai pada yang dihasilkan, dapat dihitung beberapa parameter ciri, antara lain *mean*, *skewness*, *variance*, *kurtosis*, *entropy* dan *standar deviasi*.

5. Pembagian Data

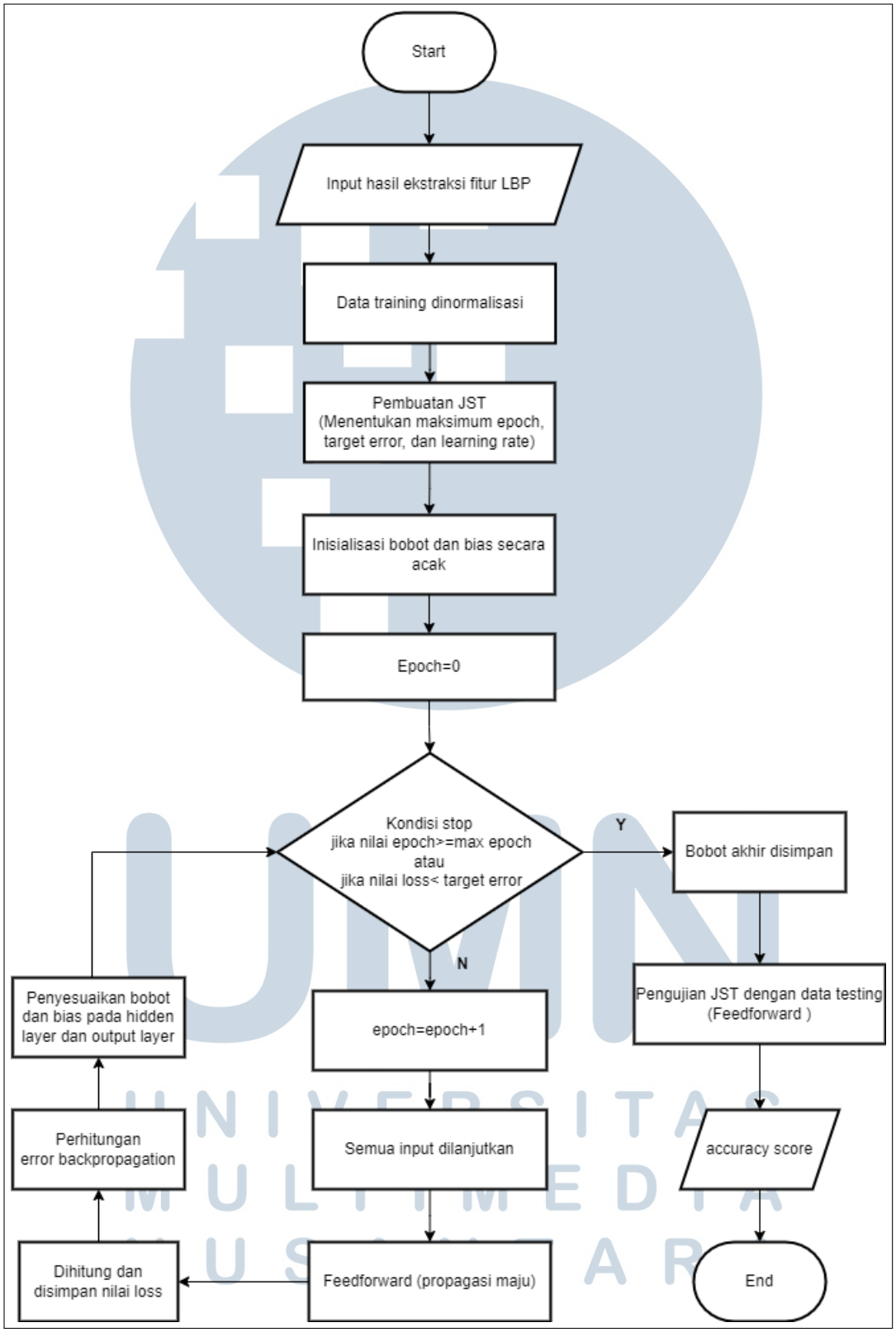
Setelah melakukan proses ekstraksi fitur dengan LBP, dilakukan pembagian data menjadi data *testing* dan data *training*. Data *training* digunakan untuk melatih suatu algoritma untuk menemukan model yang sesuai, sedangkan data *testing* digunakan untuk mengetahui kinerja algoritma yang telah dilatih sebelumnya. 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*, seperti yang ditunjukkan pada Gambar 3.4.



Gambar 3.4. Pembagian Data Testing dan Data Training

6. Jaringan Saraf Tiruan Backpropagation

Implementasi jaringan saraf tiruan (JST) *backpropagation* dilakukan menggunakan class MLP Classifier yang disediakan dalam library sklearn. Multilayer perceptron (MLP) adalah salah satu model jaringan syaraf tiruan (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 lapisan diperbarui.



Gambar 3.5. Flowchart Algoritma Backpropagation

Gambar 3.5 menunjukkan proses *training* dan *testing* pada *backpropagation*, berikut ini merupakan tahapan proses tersebut :

- Sebelum data *training* dilatih dengan *Backpropagation*, perlu dilakukan normalisasi sehingga semua data dapat dievaluasi secara seragam. Normalisasi 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 diharapkan 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 didapat dari hasil program yang telah dirancang sebelumnya, untuk memperoleh data dan fungsi yang optimal. Selain itu, Untuk mengetahui tingkat kesuksesan uji coba maka diperlukan evaluasi menggunakan *backpropagation* dengan *hidden node* dan *learning rate* berbeda untuk mengetahui tingkat akurasi. Dari beberapa parameter yang diuji fokus pengujian kepada akurasi. Setelah itu, dilakukan evaluasi untuk menilai performa dari hasil permodelan berdasarkan kriteria keberhasilan tertentu. Dalam melakukan evaluasi, digunakan *confusion matrix* untuk memperoleh nilai *precision, recall, accuracy* dan *F1-score*. *Confusion matrix* dipilih karena merupakan metode yang populer digunakan untuk memvalidasi sejumlah *learning model* [32].