BAB III

METODOLOGI DAN PERANCANGAN SISTEM

3.1 Metodologi Penelitian

Metode penelitian yang digunakan dalam penelitian ini adalah sebagai berikut:

1. Telaah Literatur

   Pada tahap ini akan dilakukan pencarian dan pembelajaran terhadap pengetahuan mengenai metode-metode, teori, dan persamaan yang dilakukan dengan mengumpulkan sumber yang terdapat pada jurnal, buku, artikel, maupun referensi lain yang terkait dengan penelitian ini.

2. Pengumpulan Data

   Pada tahap ini akan dilakukan pengumpulan data yang digunakan untuk pelatihan jaringan saraf tiruan dari berbagai sumber seperti jurnal, buku, artikel, maupun referensi lain.

3. Perancangan Aplikasi dan Implementasi

   Pada tahap ini akan dilakukan analisis terhadap alur proses aplikasi, perancangan jaringan saraf tiruan untuk pelatihan dan pengenalan karakter, serta implementasi jaringan saraf tiruan *backpropagation*. Aplikasi ini akan dibuat agar bisa mengenali karakter yang ada pada dokumen tercetak. Seluruh pengolahan citra yang dilakukan dalam penelitian ini dilakukan dengan menggunakan *library* OpenCV.

4. Pengujian dan Evaluasi

   Pada tahap ini akan dilakukan pengujian terhadap aplikasi yang sudah dikerjakan dan evaluasi. Pengujian dilakukan untuk memastikan aplikasi berjalan sesuai dengan fungsi yang sudah ditetapkan. Sedangkan, evaluasi dilakukan untuk mengetahui tingkat akurasi dari implementasi jaringan saraf tiruan *backpropagation* untuk mengenali karakter pada dokumen tercetak. Penghitungan tingkat akurasi mengacu pada penelitian yang dilakukan oleh Apriyanti dan Widodo (2016) serta Afroge dkk. (2016) yaitu dengan membagi jumlah karakter uji yang dikenali dengan jumlah karakter uji.

5. Penulisan Laporan

   Pada tahap ini akan dilakukan penulisan laporan proses penelitian ini secara bertahap. Penulisan laporan dimulai dari latar belakang penilitian hingga kesimpulan yang didapatkan dari penelitian ini.

3.2 Perancangan Jaringan Saraf Tiruan

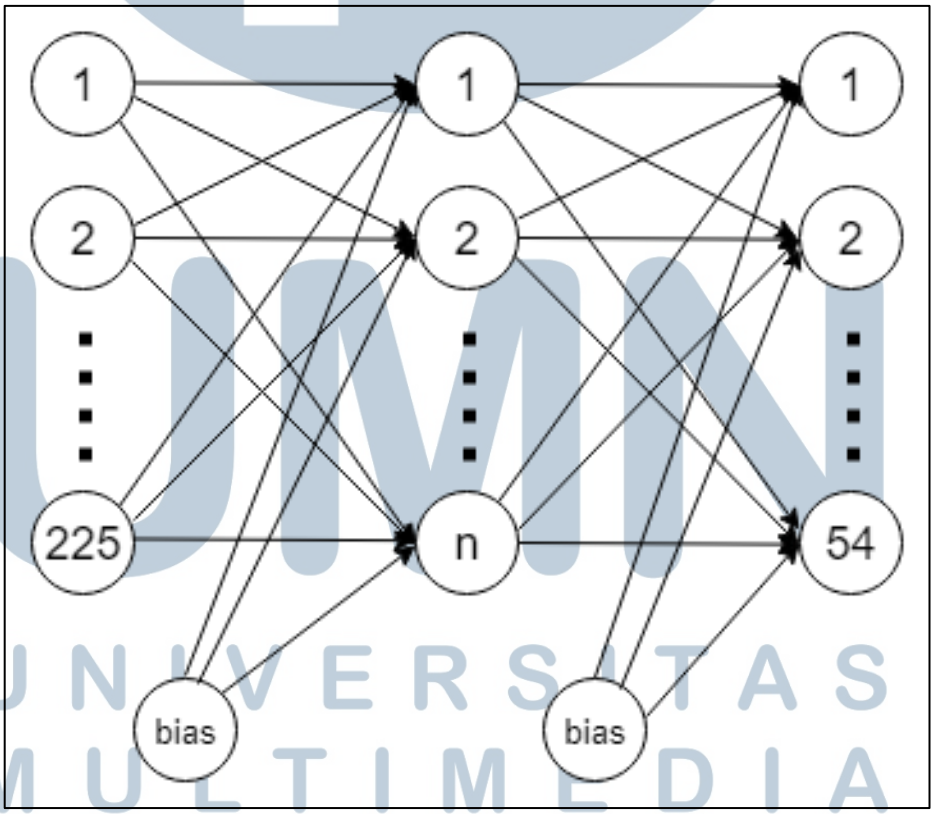
Arsitektur jaringan saraf tiruan yang digunakan dalam penelitian adalah jaringan lapisan banyak (*multilayer network*). Jaringan ini terdiri dari tiga lapisan yaitu lapisan input, lapisan tersembunyi, dan lapisan *output*. Lapisan input terdiri dari 225 *node* yang merupakan piksel-piksel dari citra input yang sudah dinormalisasi menjadi 15 × 15 piksel. Lapisan *output* terdiri dari 54 *node* yang merepresentasikan karakter A-Z, a-z, serta tanda baca titik (.) dan koma (,). Berdasarkan Rumus 3.1, jumlah *node* yang digunakan pada lapisan tersembunyi adalah 110. Sementara, *learning rate* yang digunakan berdasarkan pada Rumus

3.2 adalah 0,29. Perhitungan jumlah *node* pada lapisan tersembunyi dan *learning rate* mengacu pada penelitian yang dilakukan oleh Shibata dan Ikeda (2009).

$$N^{(h)} = \sqrt{N^{(i)}N^{(o)}} \quad \ldots(3.1)$$

$$\eta = 32/\sqrt{N^{(i)}N^{(o)}} \quad \ldots(3.2)$$

Fungsi aktivasi yang digunakan pada lapisan tersembunyi adalah ReLU dan fungsi aktivasi yang digunakan pada lapisan *output* adalah sigmoid. Pada tahap pelatihan, digunakan seluruh algoritma *backpropagation* untuk mendapatkan bobot dan bias optimal. Sedangkan, pada tahap pengenalan hanya dilakukan *forward propagation* dengan bobot dan bias yang sudah didapat pada tahap pelatihan. Arsitektur jaringan saraf tiruan yang digunakan dalam penelitian ini dapat dilihat pada Gambar 3.1.



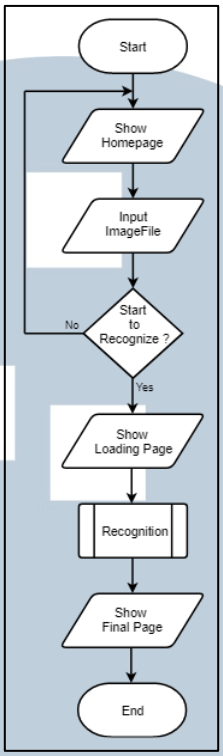
Gambar 3.1 Arsitektur Jaringan Saraf Tiruan

3.3 Flowchart

Alur dari aplikasi implementasi jaringan saraf tiruan *backpropagation* untuk pengenalan karakter pada dokumen tercetak digambarkan dalam *flowchart* yang dapat dilihat pada Gambar 3.2 hingga Gambar 3.13. *Flowchart* dibagi menjadi sembilan yaitu *flowchart* aplikasi, *flowchart recognition*, *flowchart line extraction*, *flowchart word extraction*, *flowchart letter extraction*, *flowchart normalize*, *flowchart* jaringan saraf tiruan, *flowchart forward propagation*, dan *flowchart backpropagation*.

3.3.1 Flowchart Aplikasi

Gambar 3.2 menunjukan alur proses aplikasi secara keseluruhan. Ketika aplikasi dijalankan, halaman utama ditampilkan. Pada halaman utama terdapat tombol *choose file* dan *start to recognize*. Ketika *image file* sudah dipilih dan tombol *start to recognize* ditekan maka halaman *loading* ditampilkan dan proses *recognition* dijalankan. Setelah proses *recognition* selesai dijalankan maka halaman akhir ditampilkan.

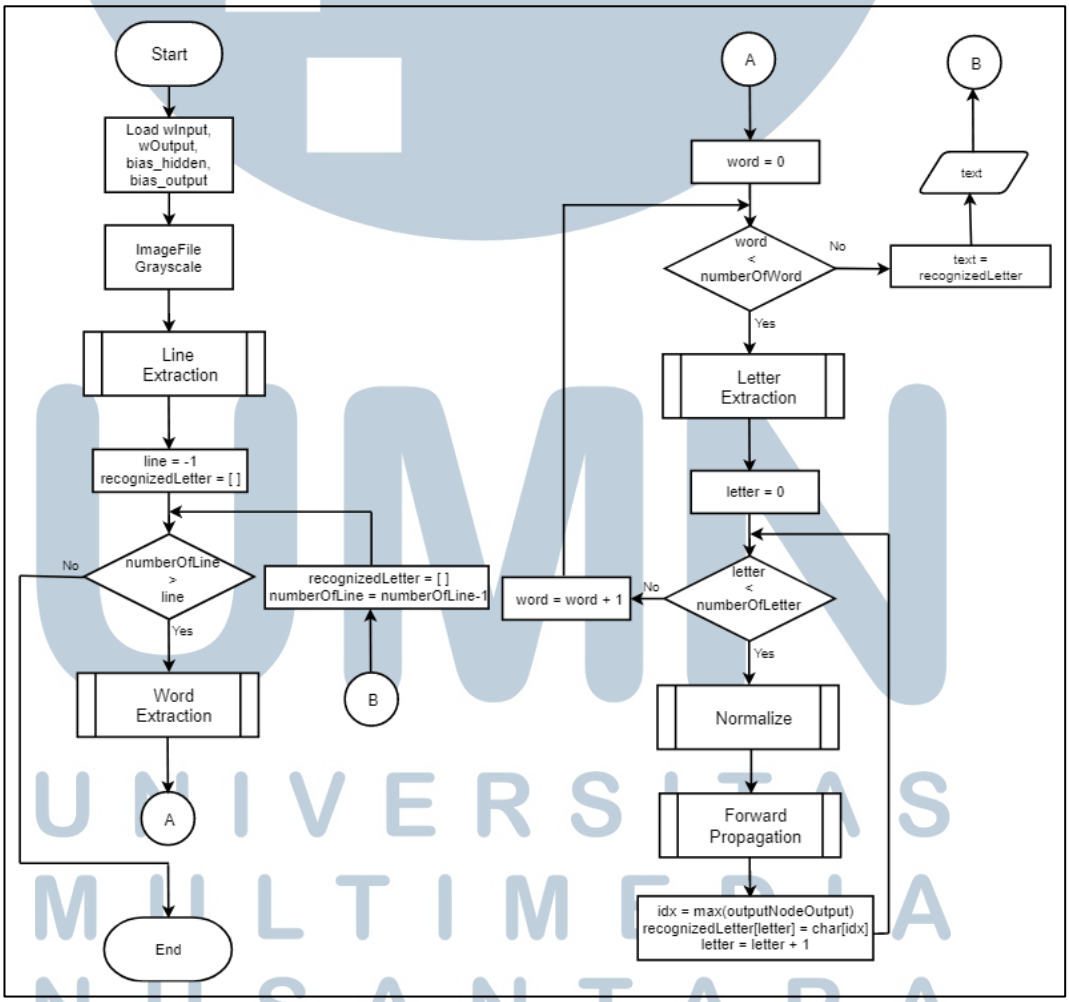


Gambar 3.2 *Flowchart* Aplikasi

3.3.2 Flowchart Recognition

Gambar 3.3 menunjukan alur dari proses *recognition*. Ketika proses dijalankan weight dan bias dimuat. Lalu, *image file* yang menjadi input diubah menjadi *grayscale image*. Setelah input diubah menjadi *grayscale image*, proses *line extraction* dijalankan. Proses *line extraction* mengembalikan jumlah baris dan *array* yang berisi *image file* yang sudah dipotong per baris. Setelah proses *line extraction* dilakukan, dijalankan iterasi sebanyak jumlah baris. Selama iterasi dilakukan, proses *word extraction* dijalankan. Proses *word extraction* mengembalikan jumlah kata dan *array* yang berisi *image file* yang sudah dipotong per kata. Selanjutnya, dijalankan iterasi sebanyak jumlah kata. Selama iterasi dilakukan, proses *letter extraction* dijalankan. Proses *letter extraction* mengembalikan jumlah huruf dan *array* yang berisi *image file* yang sudah

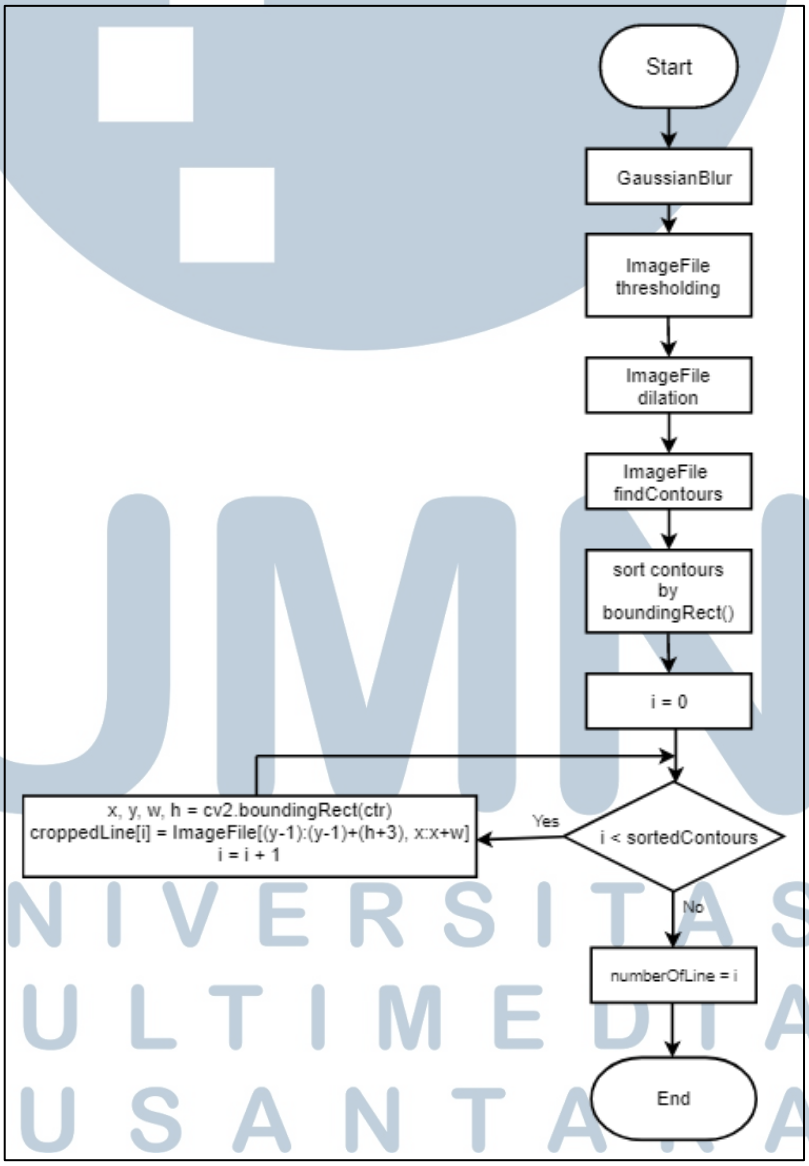
dipotong per huruf. Setelah itu, dijalankan iterasi sebanyak jumlah kata. Selama iterasi dilakukan, dijalankan proses *normalize* dan *forward propagation*. Proses *normalize* akan mengembalikan *image file* yang sudah dinormalisasi menjadi 15 × 15 piksel. *Image file* yang dikembalikan setelah proses *normalize* dijadikan input pada proses *forward propagation* untuk dikenali. Proses *forward propagation*, mengembalikan *array* yang berisi *output* dari lapisan *output* dan lapisan tersembunyi. Setelah proses *forward propagation* selesai dilakukan, nilai maksimum dari *array* yang berisi *output* dari lapisan *output* dijadikan indeks untuk diklasifikasikan sebagai karakter.



Gambar 3.3 *Flowchart Recognition*

3.3.3 Flowchart Line Extraction

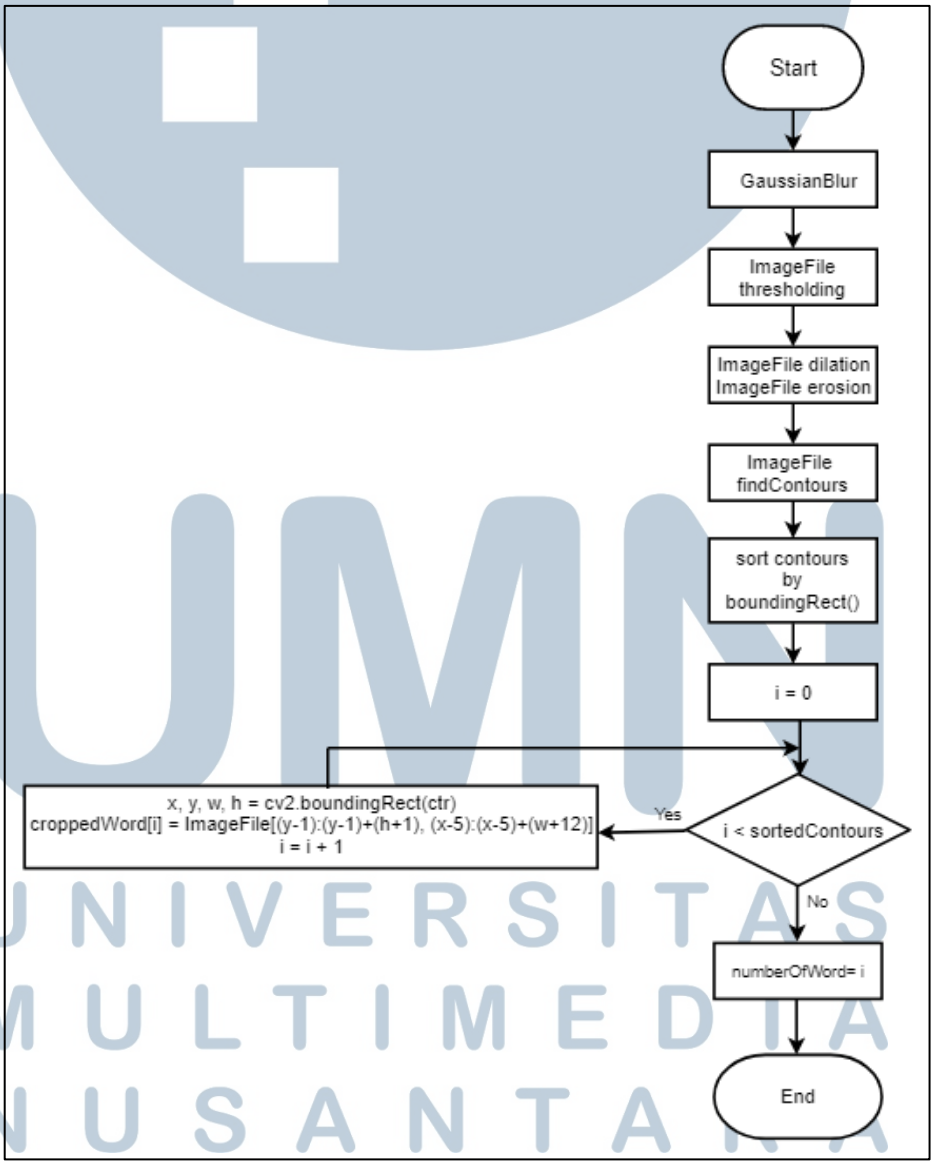
Gambar 3.4 menunjukan alur dari proses *line extraction*. Proses dimulai dengan menghaluskan citra dengan *gaussian blur*. Kemudian, ditentukan nilai *threshold* dari *image file* dan dilakukan pelebaran piksel dengan melakukan dilasi pada *image file*. Kontur pada *image file* dicari dan diurutkan berdasarkan koordinat x dan y dari *bounding box* pada *image file*. Setelah itu, dilakukan iterasi sebanyak kontur dan *image file* dipotong berdasarkan pada koordinat dari *bounding box*. *Image file* yang dipotong disimpan ke dalam *array*.



Gambar 3.4 *Flowchart Line Extraction*

3.3.4 Flowchart Word Extraction

Gambar 3.5 menunjukan alur dari proses *word extraction*. *Image file* akan diblur menggunakan *gaussian blur*. *Image file* yang sudah diblur ditentukan nilai *threshold*-nya, kemudian dilakukan pelebaran piksel dengan dilasi dan penipisan piksel dengan erosi. Setelah itu, kontur dari *image file* dicari dan diurutkan berdasarkan koordinat x dan y dari *bounding box* pada *image file*. Iterasi dilakukan sebanyak kontur dan *image file* dipotong berdasarkan pada koordinat dari *bounding box*. *Image file* yang dipotong disimpan ke dalam array.



Gambar 3.5 *Flowchart Word Extraction*

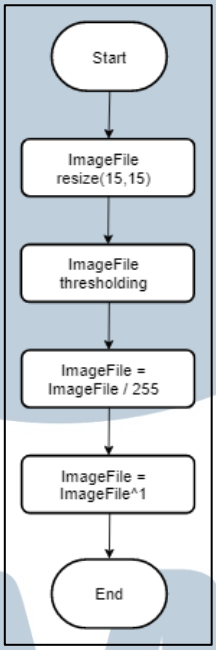
3.3.5 Flowchart Letter Extraction

Gambar 3.6 menunjukan alur dari proses *letter extraction*. Dilakukan deteksi tepi dengan operator *canny*. Kemudian, nilai *threshold* dari *image file* ditentukan. Piksel didalam tepi akan diisi dengan *flood fill* setelah deteksi tepi dilakukan. Kemudian, dilakukan pelebaran piksel dengan dilasi dan dilakukan iterasi sebanyak kontur yang sudah dicari dan diurutkan sebelumnya. Selama iterasi dilakukan *image file* dipotong berdasarkan pada koordinat dari *bounding box* dan disimpan ke dalam array.

Gambar 3.6 *Flowchart Letter Extraction*

3.3.6 Flowchart Normalize

Gambar 3.7 menunjukan alur dari proses *normalize*. Pada proses ini, *image file* akan di-*resize* menjadi 15 × 15 piksel. Setelah itu, akan ditentukan *thresholding* pada *image file*. Kemudian, *image file* diubah menjadi citra biner dengan membagi *image file* dengan 255 dan XOR *image file* dengan 1. Hasil dari proses *normalize* ini langsung dijadikan input pada jaringan saraf tiruan tanpa dilakukan *feature extraction* terlebih dahulu.

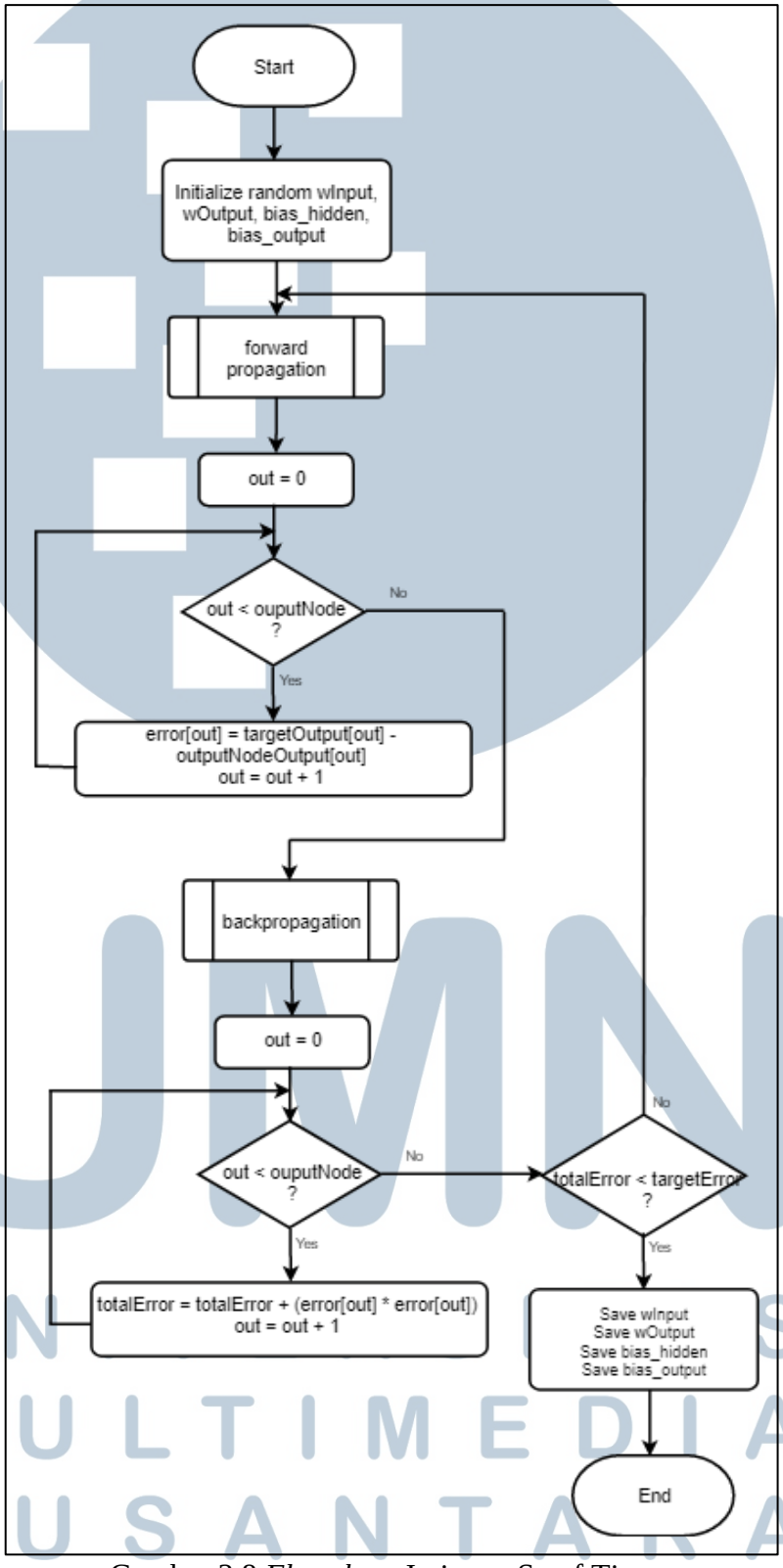


Gambar 3.7 *Flowchart Normalize*

3.3.7 Flowchart Jaringan Saraf Tiruan

Gambar 3.8 menunjukan alur dari proses jaringan saraf tiruan. Proses ini dilakukan untuk pelatihan jaringan saraf tiruan. Pada proses ini nilai bobot dan bias akan diinisialisasi secara acak. Kemudian, proses *forward propagation* akan dijalankan. Setelah itu, dilakukan perhitungan nilai *error* dari setiap *node* pada lapisan *output* dan dilakukan proses *backpropagation*. Setelah proses *backpropagation* selesai, total *error* akan dihitung dan dicek. Proses pada

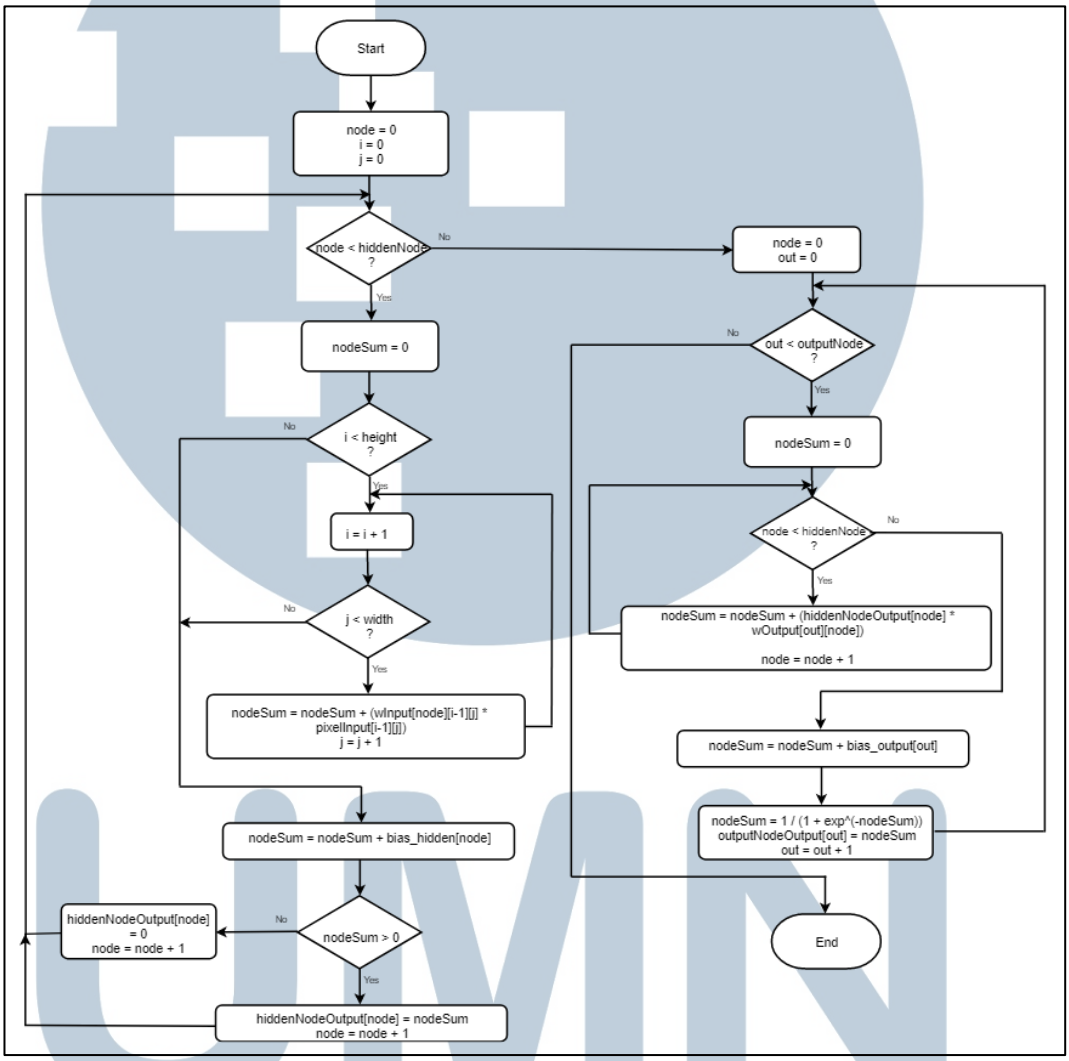
pelatihan jaringan saraf tiruan akan terus dilakukan hingga total *error* lebih kecil dari target *error*.



Gambar 3.8 *Flowchart* Jaringan Saraf Tiruan

3.3.8 Flowchart Forward Propagation

Gambar 3.9 menunjukan alur dari proses *forward propagation*. Pada proses ini dilakukan tahap perambatan maju (*forward propagation*) dari algoritma *backpropagation*. Rumus 2.1 sampai Rumus 2.4 diterapkan dalam proses ini.

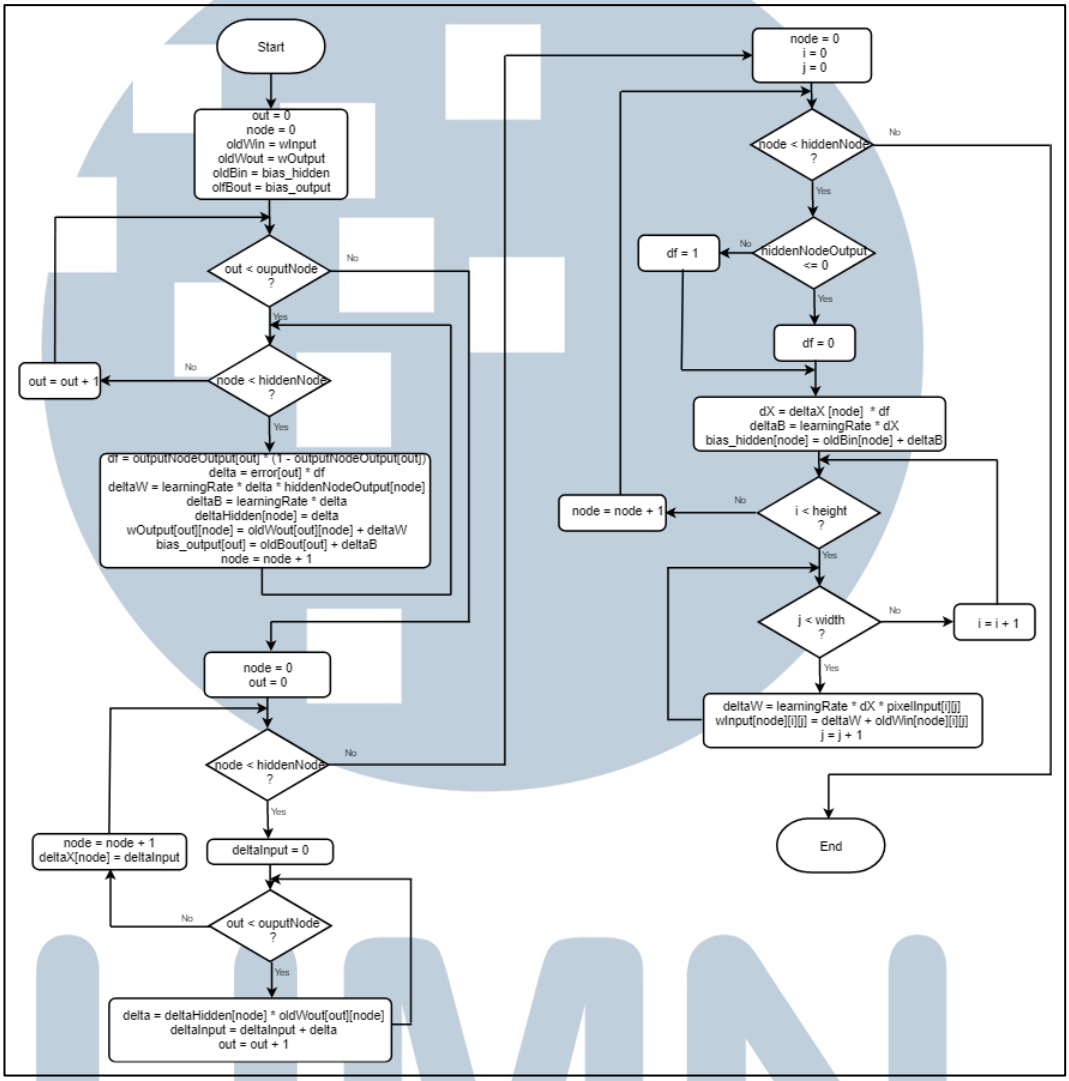


Gambar 3.9 *Flowchart Forward Propagation*

3.3.9 Flowchart Backpropagation

Gambar 3.10 menunjukan alur dari proses *backpropagation*. Pada proses ini dilakukan tahap perambatan balik (b*ackpropagation*) sekaligus perubahan bobot

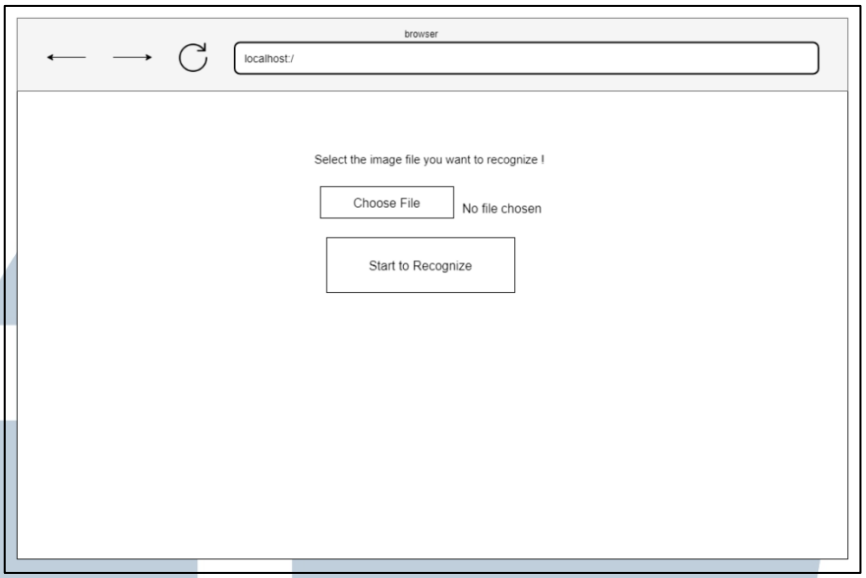
dan bias dari algoritma *backpropagation*. Rumus 2.5 sampai Rumus 2.13 diterapkan dalam proses ini.



Gambar 3.10 *Flowchart Backpropagation*

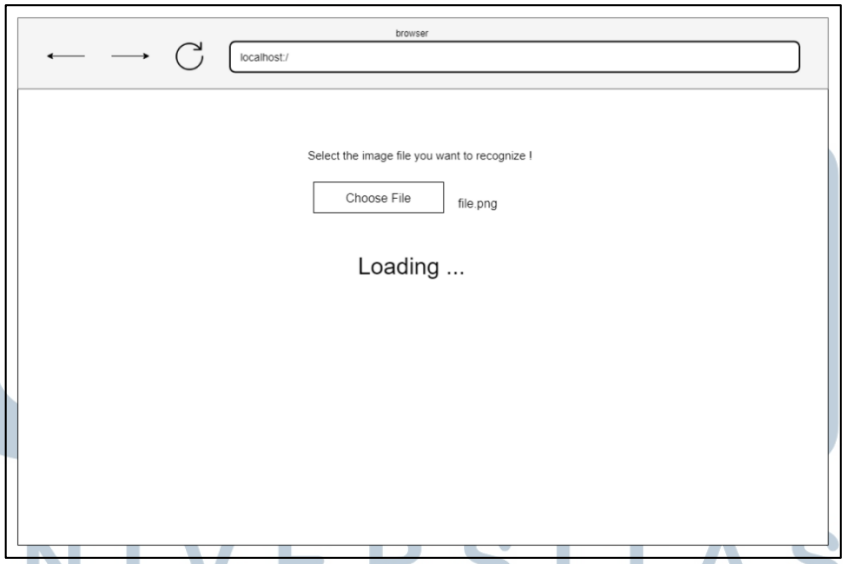
3.4 Rancangan Antarmuka Aplikasi

Rancangan antarmuka untuk implementasi jaringan saraf tiruan *backpropagation* untuk pengenalan karakter pada dokumen tercetak terdiri dari halaman awal, halaman *loading*, dan halaman akhir. Berikut ini merupakan rancangan antarmuka untuk implementasi jaringan saraf tiruan *backpropagation* untuk pengenalan karakter pada dokumen tercetak.



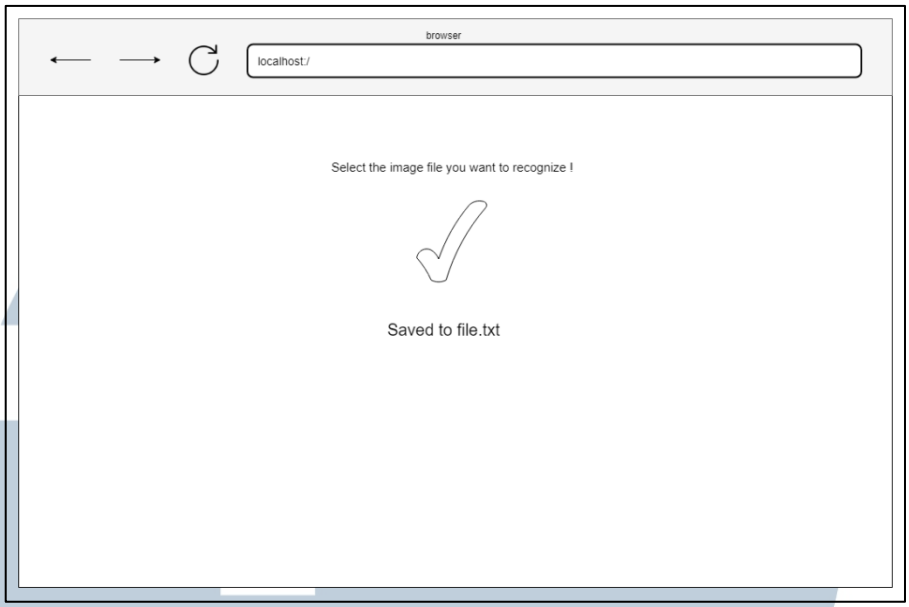
Gambar 3.11 Rancangan Halaman Awal

Gambar 3.11 merupakan rancangan antarmuka halaman awal aplikasi. Pada halaman ini terdapat tombol *choose file* dan *start to recognize*. Tombol *choose file* digunakan untuk memilih *image file* yang ingin dikenali. Sedangkan, tombol *start to recognize* digunakan untuk memulai pengenalan pada *image file*.



Gambar 3.12 Rancangan Halaman *Loading*

Gambar 3.12 merupakan rancangan antarmuka halaman *loading* aplikasi. Halaman ini ditampilkan saat proses pengenalan sedang dilakukan.



Gambar 3.13 Rancangan Halaman Akhir

Gambar 3.13 merupakan rancangan antarmuka halaman akhir aplikasi. Halaman ini ditampilkan saat proses pengenalan sudah selesai dilakukan.