BAB III

METODOLOGI DAN PERANCANGAN SISTEM

3.1 Metodologi Penelitian

Metodologi yang akan digunakan dalam penelitian ini adalah sebagai berikut.

1. Studi Literatur dan Studi Fisibilitas

   Dalam studi literatur, dilakukan pembelajaran dan pendalaman teori-teori terkait *Licence Plate Recognition* (LPR), algoritma *Labeling* (*connected-component analysis*), *Artificial Neural Network* (ANN), *F-measure*, dan *purposive sampling*. Dalam studi fisibilitas dilakukan pengumpulan data kuesioner dengan *purposive sampling* kepada sampel sejumlah tiga puluh karyawan Universitas Multimedia Nusantara dengan kendaraan yang masuk area parkir melalui portal parkir *lobby*.

2. Perancangan Sistem

   Secara garis besar, tahapan kerja sistem adalah sebagai berikut.

   a. Mendeteksi Pelat Nomor (melakukan deteksi pelat pada *frame* kamera secara *realtime* untuk melakukan pengambilan gambar yang mengandung pelat nomor). Proses ini akan menggunakan API TensorFlow untuk melakukan deteksi objek pelat secara *realtime*.
   b. Mengenali Pelat Nomor – *License Plate Recognition* dengan langkah-langkah sebagai berikut.
      i. *Preprocessing* (pengolahan citra digital menjadi bentuk yang dapat diolah dan diproses), proses ini akan menggunakan API OpenCV untuk mengolah citra digital.

ii. *Plate localization* (pendeteksian pelat nomor dan pengambilan gambar secara otomatis), proses ini akan menggunakan API OpenCV untuk mencari pola pelat nomor.

iii. *Plate Transformation* (mentransformasi gambar pelat menjadi sebuah gambar pelat nomor yang tegak – *plate orientation and seizing*), proses ini akan menggunakan API OpenCV untuk mengolah gambar pelat menjadi siap disegmentasi.

iv. *Character segmentation* (mengelompokan karakter-karakter pada pelat nomor ke dalam bentuk segmen-segmen), proses ini akan menggunakan algoritma *Labeling* (*connected-component analysis*).

v. *Character recognition* (mengklasifikasi karakter-karakter pada pelat nomor), proses ini akan menggunakan *Artificial Neural Network*.

vi. *Vehicle Identification* (identifikasi kendaraan), proses ini dilakukan dengan mengirim karakter-karakter pelat nomor hasil dari tahap *character recognition* ke *custom* API PHP untuk dicek apakah kendaraan sudah terdaftar dalam sistem. Jika sudah terdaftar, maka akan dikembalikan informasi kendaraan beserta pemiliknya, namun jika tidak maka akan dikembalikan informasi bahwa kendaraan tidak terdaftar.

Perancangan dimulai dengan membuat arsitektur *neural network, Data Flow Diagram* (DFD), kemudian *flowchart,* diikuti dengan rancangan struktur tabel, antarmuka untuk aplikasi Android dan sistem admin.

3. Implementasi

Pada tahap ini, akan dimulai dengan membuat *database* MySQL dan sistem admin berbasis web dengan bahasa pemrograman PHP. Kemudian akan dibangun aplikasi utama berbasis Android dengan bahasa pemrograman Java menggunakan *library* OpenCV untuk *preprocessing* dan TensorFlow untuk *realtime object detection*. Setelah itu akan dikumpulkan tiga puluh gambar foto pelat nomor agar *library* TensorFlow dapat mengenali objek pelat nomor. Selain itu, dikumpulkan juga data *training* untuk ANN berupa pelat nomor yang mencakup karakter-karakter A-Z dan 0-9 dengan masing-masing karakter sebanyak lima buah. Arsitektur ANN yang akan digunakan adalah tiga *layer*, dengan satu *input layer*, satu *hidden* layer, dan satu *output layer*. Jumlah *node* yang digunakan pada *input layer* adalah dua ratus *node*. Jumlah *node* pada *hidden layer* menjadi variabel penelitian ini untuk mendapatkan jumlah yang optimal. Jumlah *node* pada *output layer* adalah 36. ANN akan dilatih menggunakan *Back Propagation*.

4. Pengujian dan Evaluasi

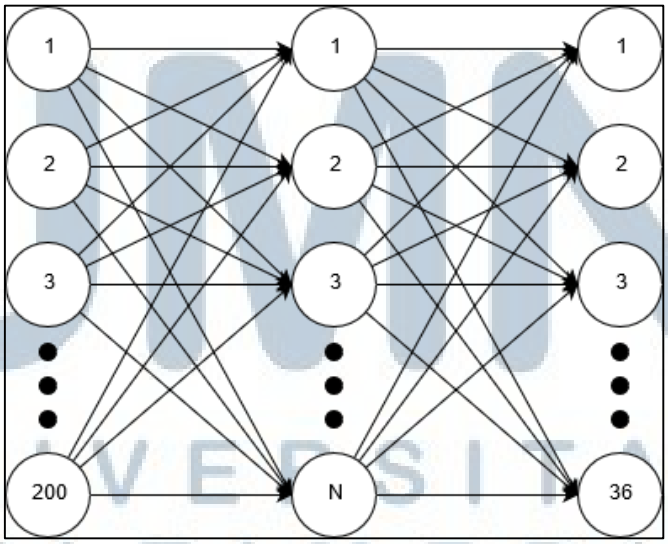
Pengujian dilakukan dengan melihat keberhasilan pengenalan pelat nomor dengan algoritma *labeling* dan *Artificial Neural Network*. Evaluasi dilakukan dengan pencatatan hasil dari pengujian akurasi dalam pendeteksian pelat nomor dengan menggunakan *F-measure*, juga mencari jumlah *neuron* pada *hidden layer neural network* yang cocok untuk sistem pengenal pelat nomor kendaraan ini.

5. Dokumentasi

Pada tahap ini, dibuat sebuah laporan sebagai dokumentasi dari penelitian dan pembuatan aplikasi secara bertahap mulai dari pendahuluan hingga kesimpulan dan saran.

3.2 Arsitektur Neural Network

Arsitektur *neural network* yang digunakan digambarkan pada Gambar 3.1. *Neural network* akan digunakan untuk pelatihan dan pengujian. Tahap pelatihan akan digunakan algoritma *Back Propagation* untuk mendapatkan bobot optimal yang akan disimpan menjadi suatu model latih yang akan disimpan dalam bentuk *file*. *Neural network* akan dilatih sampai dicapai nilai *error* yang lebih kecil atau sama dengan nilai maksimum *error* yang telah ditentukan. Tahap pengujian hanya akan dilakukan *feed forward* dengan bobot-bobot optimal yang telah disimpan hasil latihan dengan tujuan mendapatkan sebuah keluaran.



Gambar 3.1 Arsitektur Neural Network

Detil arsitektur yang akan digunakan adalah dua ratus *node input* yang terdiri dari setiap *pixel* gambar karakter yang di-*resize* menjadi 10x20 *pixel*, 85 *node*

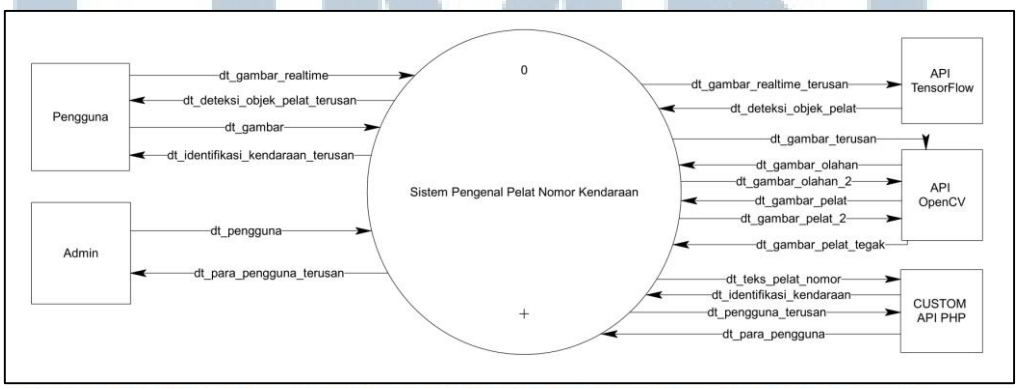
hidden yang didapat dari hasil perhitungan menentukan *node hidden*, dan 36 *node output* untuk karakter A-Z dan 0-9. *Learning rate* yang digunakan adalah 0,38 berdasarkan Rumus 2.2. Namun, jumlah *node hidden* dan *learning rate* akan disesuaikan jika memang terjadi kegagalan dalam *proses learning*.

3.3 Perancangan Sistem

Perancangan sistem ini terdiri dari *Data Flow Diagram* (DFD) dan *flowchart*, serta membuat perancangan struktur tabel, antarmuka aplikasi Android, dan antarmuka sistem admin.

3.3.1 Data Flow Diagram (DFD)

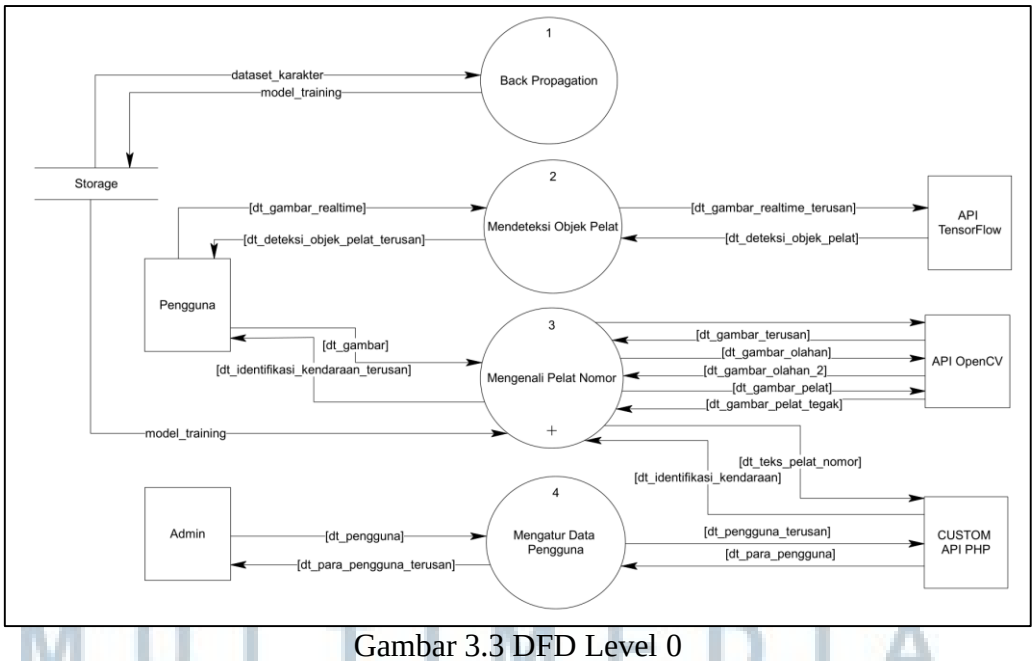
Gambar 3.2 merupakan *context diagram* dari sistem yang dibuat. Ada lima entitas pada sistem ini yaitu pengguna, admin, *Application Programming Interface* (API) TensorFlow, API OpenCV, dan *Custom* API PHP. *Custom* API PHP dibuat untuk mempermudah pemrosesan dari Android dan dari sistem admin. Penjelasan lebih lanjut mengenai sistem ini digambarkan dengan *context diagram*.



Gambar 3.2 Context Diagram

Gambar 3.3 merupakan DFD level 0, turunan dari sistem pengenal pelat nomor kendaraan pada *context diagram*. Secara garis besar, sistem ini akan

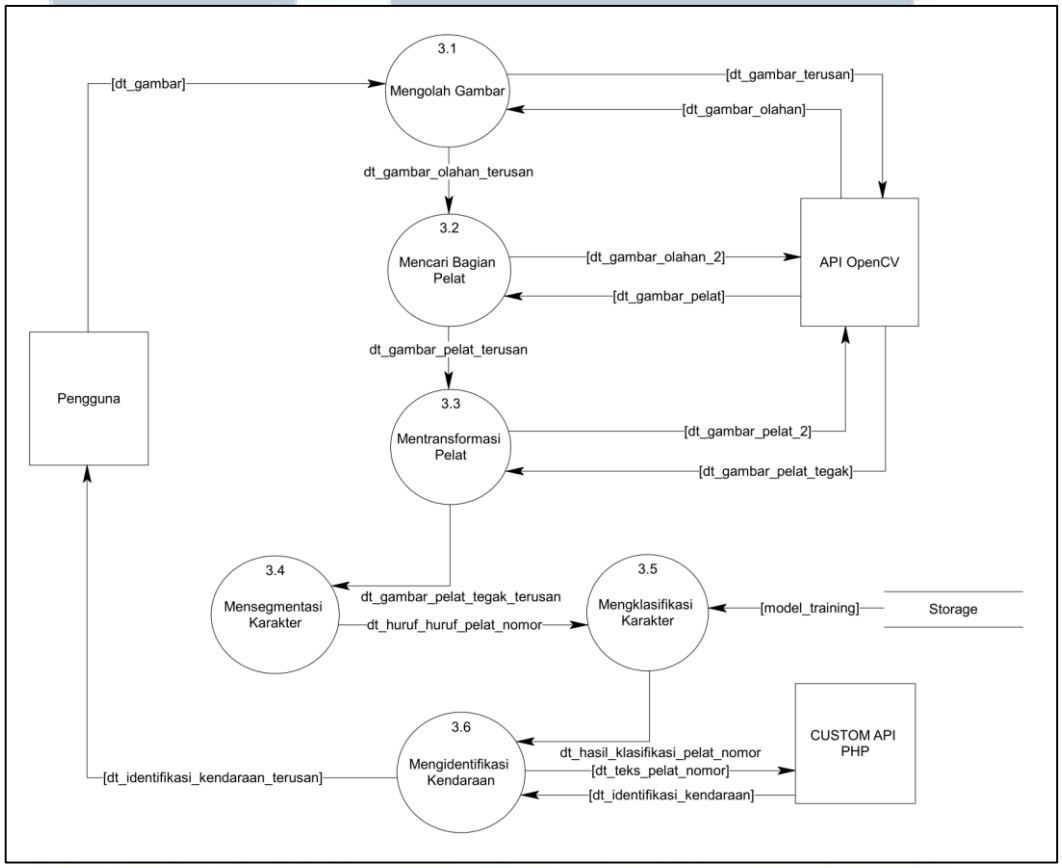
memiliki dua proses yaitu mendeteksi pelat nomor dan mengenali pelat nomor, namun akan didahului oleh proses *Back Propagation* untuk *training network*. *Back Propagation* akan mengambil dataset dari *storage* dan menyimpan model *training* ke *storage*. Proses mendeteksi pelat nomor menggunakan bantuan API TensorFlow untuk melakukan deteksi objek pelat nomor pada setiap *frame* gambar kamera secara *realtime*. Pengguna mengirimkan data berupa setiap *frame* gambar kamera kepada TensorFlow secara berulang. Ketika TensorFlow mendeteksi ada objek pelat nomor pada gambar tersebut, TensorFlow akan mengirimkan data berupa tanda ada objek pelat nomor pada *frame* tersebut. Kemudian pengguna akan melakukan pengambilan gambar tersebut untuk dikenali pada proses mengenali pelat nomor. Proses mengenali pelat nomor menggunakan konsep *License Plate Recognition* (LPR). Admin yang bertugas untuk mengatur atau melakukan *Create, Retrieve, Update,* dan *Delete* (CRUD) data pengguna melalui Custom API PHP.



Gambar 3.3 DFD Level 0

Gambar 3.4 merupakan DFD Level 1, turunan dari proses mengenali pelat nomor pada DFD level 0. Proses mengenali ini memiliki lima tahap utama yaitu

mengolah gambar (*preprocessing*), mencari bagian pelat dari gambar (*plate localization*), mentransformasikan gambar pelat menjadi gambar pelat yang tegak (*plate transformation*), mensegmentasi karakter (*character segmentation*), dan mengklasifikasikan karakter (*character recognition*). Proses mengolah gambar, mencari bagian pelat, dan mentransformasikan gambar pelat dilakukan dengan bantuan API OpenCV. Setelah kelima proses tersebut selesai, proses mengenali ini diakhiri dengan mengidentifikasi kendaraan tersebut, apakah kendaraan tersebut terdaftar dalam sistem untuk dikembalikan ke pengguna.



Gambar 3.4 DFD Level 1

3.3.2 Flowchart

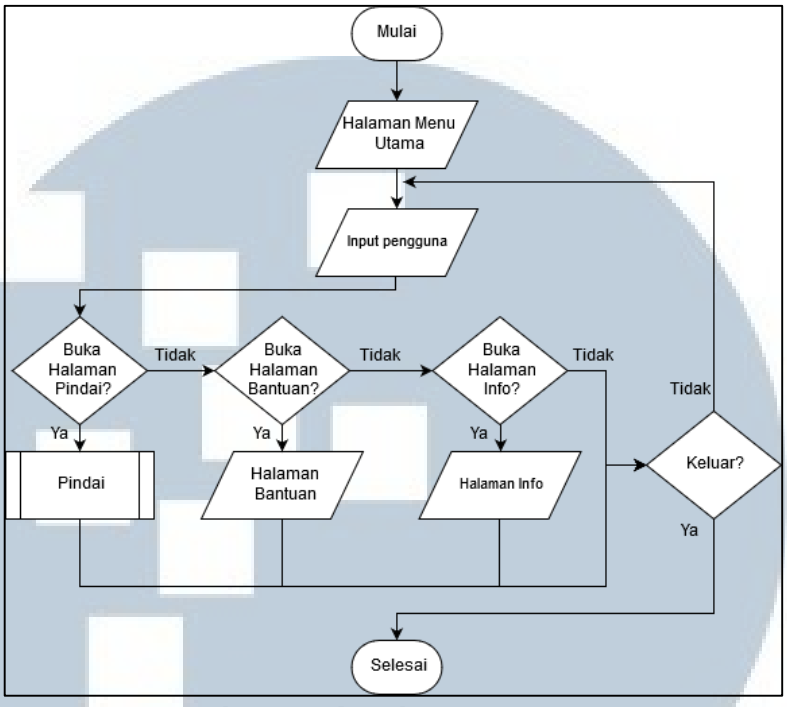
Flowchart digunakan untuk menjelaskan alur dari aplikasi yang dibuat. *Flowchart* sistem terdiri dari dua bagian besar yaitu *flowchart* aplikasi Android dan *flowchart* sistem admin. *Flowchart* aplikasi Android terbagi menjadi sebelas bagian lebih spesifik yaitu menu utama, pindai, mengenali pelat nomor, mensegmentasi karakter, *find pattern, get neighbor labels, aggregate pattern, sort pattern,* mengklasifikasikan karakter, *Artificial Neural Network,* dan *softmax*. *Flowchart* mensegmentasi karakter, *find pattern, get neighbor labels, aggregate pattern, sort pattern,* mengklasifikasikan karakter, *Artificial Neural Network,* dan *softmax* digambarkan dalam bentuk *flowchart* program sedangkan sisanya digambarkan dalam bentuk *flowchart* sistem. *Flowchart* admin digambarkan dalam bentuk *flowchart* sistem.

A. Flowchart Aplikasi Android

Aplikasi Android ini adalah aplikasi utama yang akan digunakan oleh pengguna. Aplikasi ini akan terdiri dari halaman menu utama, pindai, bantuan, info, dan hasil. Halaman-halaman tersebut akan ditampilkan dalam *flowchart* menu utama, pindai, mengenali pelat nomor, mensegmentasi karakter, *find pattern, get neighbor labels, aggregate pattern, sort pattern,* dan mengklasifikasikan karakter.

A.1 Menu Utama

Gambar 3.5 merupakan *flowchart* menu utama. *Flowchart* menu utama hanya memberikan *output* berupa tampilan halaman berisi navigasi ke halaman-halaman lainnya sesuai *input* pengguna.

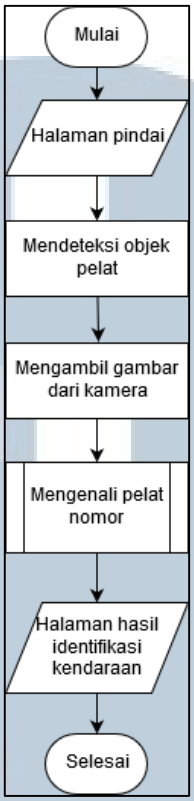


Gambar 3.5 Flowchart Menu Utama

Pertama kali ditampilkan halaman menu utama. Jika ingin ke halaman pindai, maka akan menuju ke submodul pindai. Jika ingin ke halaman bantuan, maka akan terbuka halaman bantuan. Jika ingin ke halaman info, maka akan terbuka halaman info. Jika tidak ketiganya, maka pengguna diberi pilihan untuk mengakhiri aplikasi.

A.2 Pindai

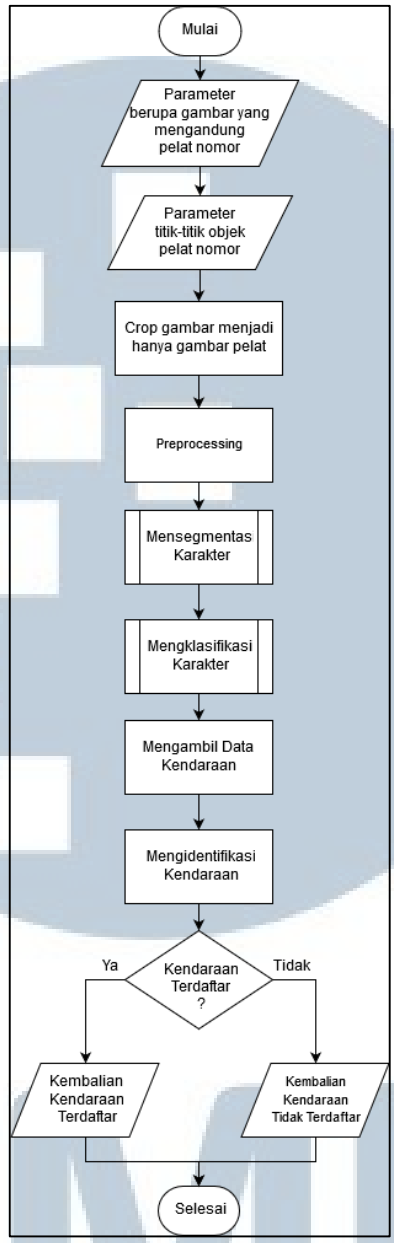
Halaman pindai ini akan menampilkan gambar yang dipindai dari perangkat kamera. Setelah itu, aplikasi akan memberikan respon berupa tampilan identifikasi kendaraan, apakah kendaraan terdaftar atau tidak. *Flowchart* ini merupakan *flowchart* rujukan dari DFD level 0. Berikut Gambar 3.6 merupakan *flowchart* Pindai.



Gambar 3.6 Flowchart Pindai

A.3 Flowchart Mengenali Pelat Nomor

Gambar 3.7 adalah *flowchart* modul mengenali pelat nomor pada *flowchart* utama. *Flowchart* ini merupakan rujukan dari DFD level 1 pada Gambar 3.4. Pada modul ini, akan diterima parameter berupa gambar yang mengandung pelat nomor dan empat titik sudut objek pelat nomor dari *Tensorflow Object Detection* API. Kemudian gambar yang mengandung pelat akan di-*crop* menjadi hanya gambar pelat saja, di-*preprocessing* sampai gambar pelat terolah. Proses tersebut akan menggunakan bantuan API OpenCV.

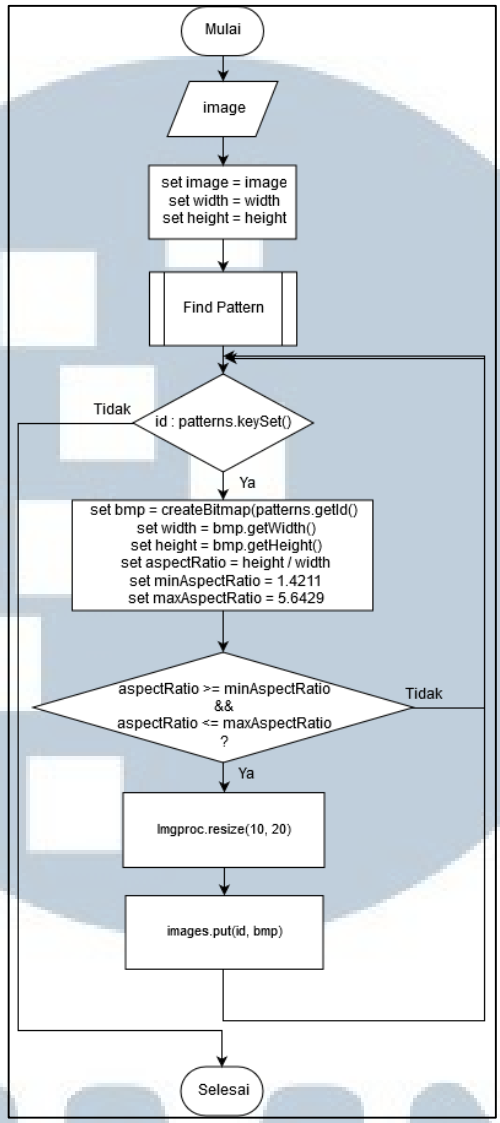


Gambar 3.7 Flowchart Mengenali Pelat Nomor

Setelah ketiga proses tersebut, akan dilanjutkan dengan submodul segmentasi karakter menggunakan algoritma *Labeling* (*connected-component analysis*) dan mengklasifikasikan karakter-karakter tersebut untuk dibaca menggunakan *Artificial Neural Network* (ANN). Hasil klasifikasi karakter tersebut berupa teks pelat nomor akan dibandingkan dengan data-data pada sistem melalui PHP *microservice* untuk mengembalikan data kendaraan terdaftar atau tidak.

A.4 Flowchart Mensegmentasi Karakter

Gambar 3.8 adalah *flowchart* modul mensegmentasi karakter pada Gambar 3.7. *Flowchart* ini dimulai dengan menerima parameter image dan menyimpan parameter image tersebut ke dalam variabel image. Dari variabel image tersebut, disimpan nilai *width* dari image ke dalam variabel width dan nilai *height* dari *image* ke dalam variabel height. Variabel patterns didapat dari hasil modul *Find Pattern*. Setelah itu akan dilakukan perulangan berdasarkan keySet() dari variable patterns yang disimpan ke dalam id. Di dalam perulangan, dilakukan deklarasi variabel lokal perulangan bmp, width, height, aspectRatio, minAspectRatio, dan maxAspectRatio. Setelah deklarasi, bmp diinisialisasi dengan hasil dari fungsi createBitmap gambar patterns.getId(), width diinisialisasi dengan nilai bmp.getWidth(), height diinisialisasi dengan nilai bmp.getHeight(), aspectRatio diinisialisasi dengan hasil dari height dibagi width, minAspectRatio diinisialisasi dengan nilai 1,4211, dan maxAspectRatio diinisialisasi dengan nilai 5,6429. Setelah itu, dicek apakah nilai aspectRatio ada di antara nilai minAspectRatio dan maxAspectRatio. Jika ya, maka gambar akan di-*resize* dengan fungsi dari OpenCV menjadi gambar 10x20 px dan gambar akan dimasukan ke dalam variabel images.

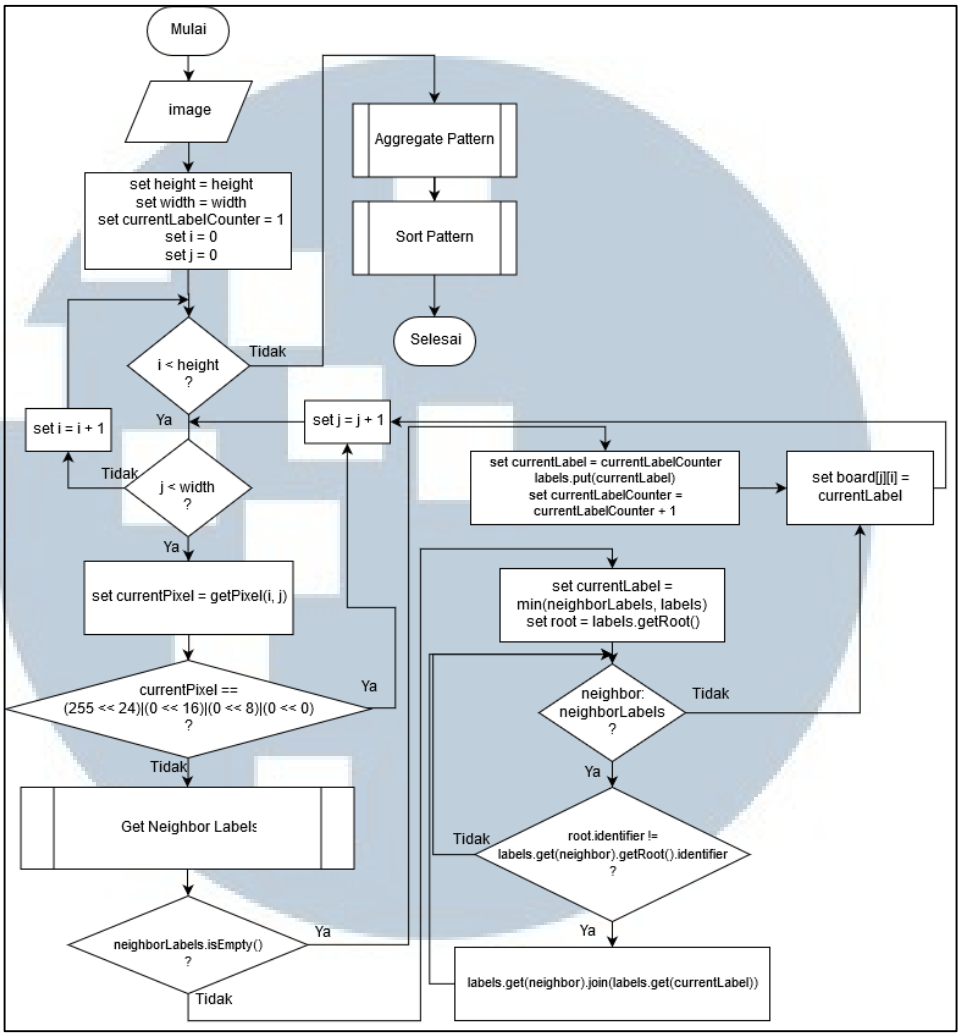


Gambar 3.8 Flowchart Mensegmentasi Karakter

A.5 Flowchart Find Pattern

Gambar 3.9 adalah *flowchart* modul *Find Pattern* pada Gambar 3.8. Proses ini bertujuan untuk memberi label awal pada setiap *pixel* gambar dilihat dari *pixel* tetangganya. *Flowchart* ini dimulai menerima parameter image, kemudian variabel image diinisialisasi dengan parameter image. Setelah itu, variabel height diisi dengan *height* dari image, variabel width diisi dengan *width* dari image, currentLabelCounter diisi dengan nilai satu, i diisi dengan nilai nol, dan j diisi

dengan nilai nol. Setelah itu dilakukan perulangan sebanyak jumlah *pixel* dari image. Di awal perulangan, variabel currentPixel diisi dengan *pixel* pada kolom ke j dan baris ke i. Setelah itu, nilai currentPixel akan dicek apakah currentPixel memiliki nilai yang sama dengan $(255 \ll 24) \mid (0 \ll 16) \mid (0 \ll 8) \mid (0 \ll 0)$ atau warna *background* (hitam). Jika ya maka akan dilanjutkan ke *pixel* kolom berikutnya, jika tidak maka akan dicari *pixel* tetangga dari currentPixel dengan fungsi getNeighborLabels(currentPixel). Dari semua nilai label pada *pixel* tetangga akan dicek apakah semua tetangga belum memiliki label, jika ya maka nilai currentLabel akan diisi dengan nilai currentLabelCounter, nilai currentLabel tersebut akan dimasukkan ke dalam sebuah *list* labels, dan nilai currentLabelCounter akan di-*increment*. Jika tidak maka nilai currentLabel akan diisi dengan nilai terkecil antara label-label tetangga dengan nilai dari *list* labels lalu nilai root akan diisi dengan nilai root dari label tersebut. Setelah itu, akan dilakukan perulangan sebanyak jumlah *pixel* tetangga, untuk memberi root pada semua *pixel* tetangga yang bukan *background* menjadi currentLabel. Setelah pengecekan sudah atau belumnya nilai *pixel* tetangga memiliki label, maka variabel board[i][j] akan diisi dengan nilai currentLabel dan perulangan setiap *pixel* berlanjut. Jika semua proses perulangan setiap *pixel* telah selesai maka akan dilanjutkan dengan pemanggilan modul Aggregate Pattern dan Sort Pattern.

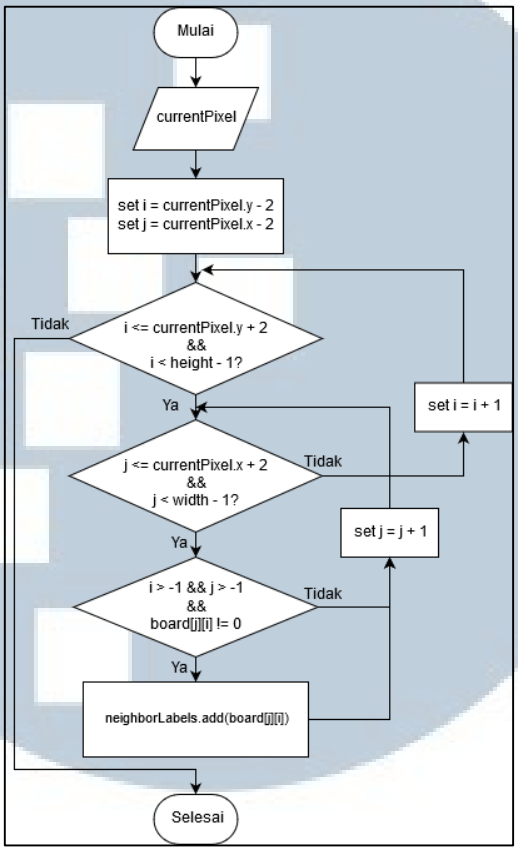


Gambar 3.9 Flowchart Find Pattern

A.6 Flowchart Get Neighbor Labels

Gambar 3.10 adalah *flowchart* modul *Get Neighbor Labels* pada Gambar 3.9. Proses ini bertujuan untuk menghasilkan label-label yang dimiliki *pixel* tetangga dari *pixel* sekarang. *Flowchart* ini diawali dengan menerima parameter currentPixel. Variabel i diisi dengan $currentPixel.y - 2$ dan variabel j diisi dengan $currentPixel.x - 2$. Setelah itu, dilakukan perulangan sampai nilai i menjadi $currentPixel.y + 2$ tetapi tidak lebih dari $width - 1$ dan nilai j menjadi $currentPixel.x + 2$ tetapi tidak lebih dari $height - 1$. Perulangan itu selalu

melakukan pengecekan nilai i dan j lebih besar dari minus satu dan nilai $board[j][i]$ tidak sama dengan nol.

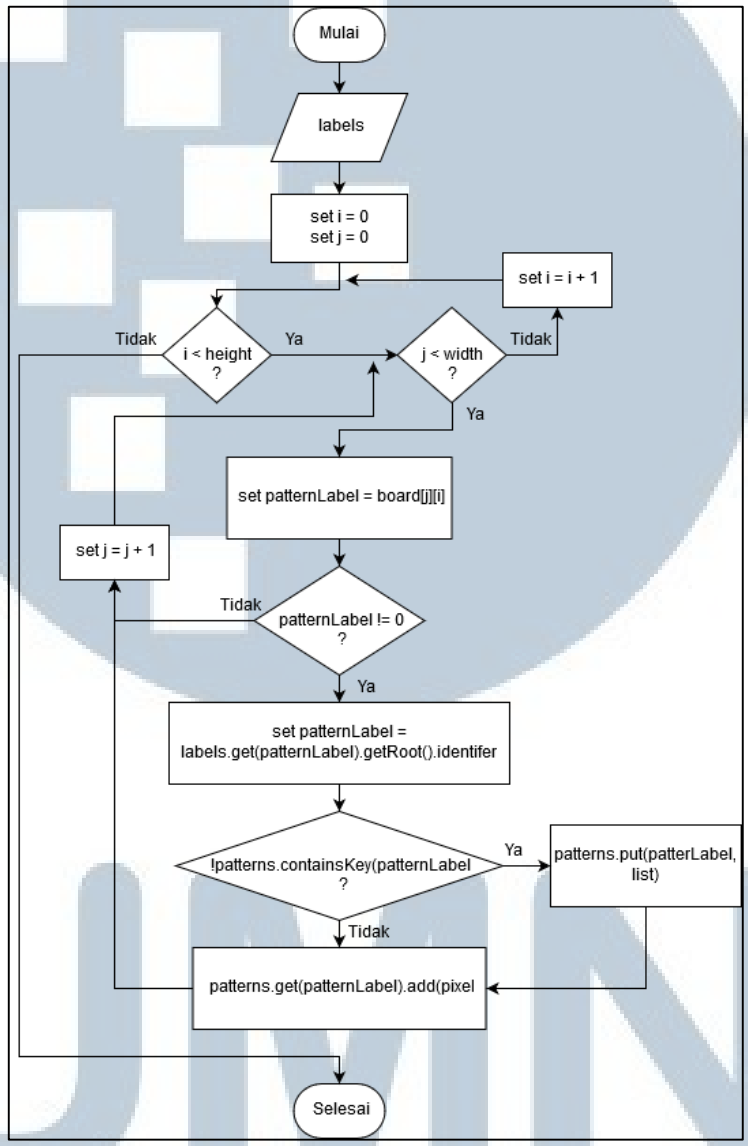


Gambar 3.10 Flowchart Get Neighbor Labels

A.7 Flowchart Aggregate Pattern

Gambar 3.11 merupakan *flowchart* dari modul *aggregate pattern* pada Gambar 3.9. Proses ini bertujuan untuk mengaggregasi pola yang sudah diberi label pada tahap pertama, dikarenakan mungkinnya ada satu komponen yang memiliki lebih dari satu label. *Flowchart aggregate pattern* ini dimulai dari menerima parameter labels, kemudian menginisialisasi variabel i dan j dengan nilai nol untuk iterasi setiap *pixel* gambar dari kiri atas. Di dalam iterasi akan diambil patternLabel dari variabel board[j][i]. Kalau ternyata patternLabel bernilai nol atau merupakan warna *background* maka lanjutkan iterasi, namun jika tidak nol maka akan dilihat

nilai root pada *pixel* tersebut. Kalau label ada di pola, maka akan disimpan, jika tidak maka akan dibuat baru. Lalu, iterasi dilanjutkan sampai semua *pixel* sudah diagregasi.

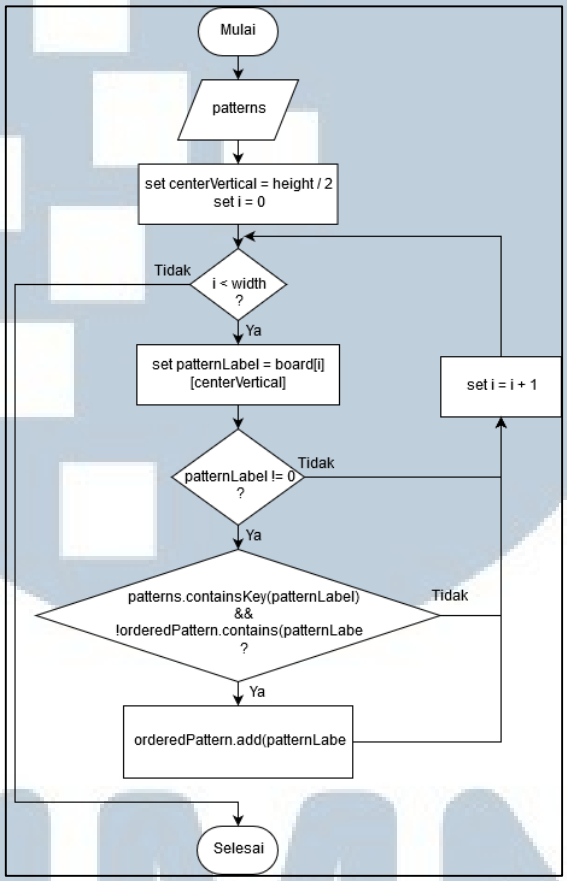


Gambar 3.11 Flowchart Aggregate Pattern

A.8 Flowchart Sort Pattern

Gambar 3.12 merupakan *flowchart* dari modul *sort pattern* pada Gambar 3.9. Proses ini merupakan iterasi tambahan pada proses *Labeling* untuk mengurutkan komponen karakter yang sudah didapat, terhitung dari setengah dari tinggi pelat.

Flowchart Sort Pattern ini dimulai dari menerima parameter *patterns* yang sudah teragregasi, kemudian melakukan set nilai centerVertical dengan tinggi pelat dibagi dua dan set nilai i sama dengan nol untuk iterasi. Iterasi sebanyak lebar pelat, pola diurutkan berdasarkan urutan ditemukannya label dari kiri ke kanan.

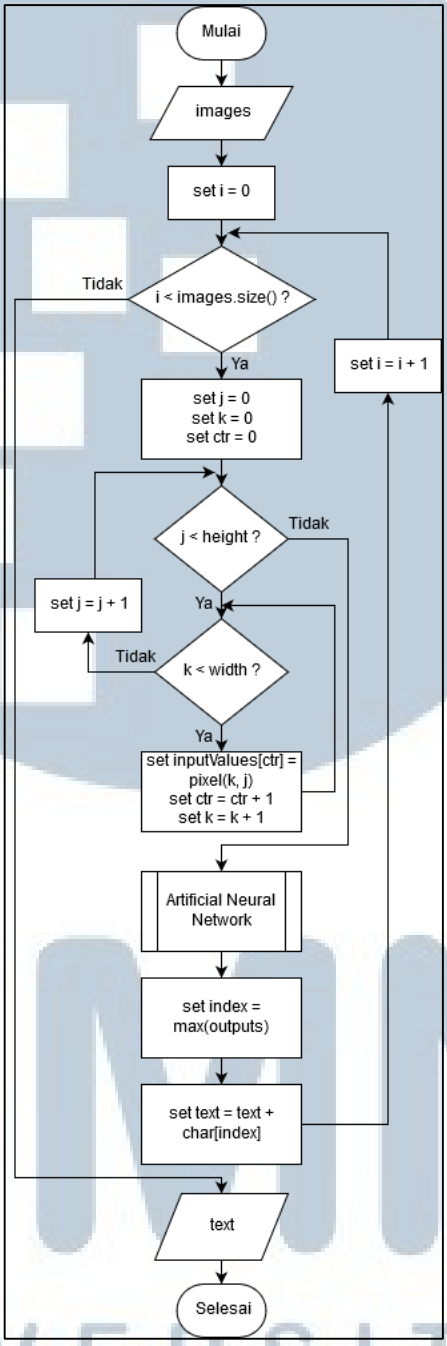


Gambar 3.12 Flowchart Sort Pattern

A.9 Flowchart Mengklasifikasikan Karakter

Gambar 3.13 merupakan *flowchart* mengklasifikasikan karakter. Dimulai dari menerima input images dan set variabel i menjadi nol untuk iterasi. Setelah itu akan dilakukan iterasi sebanyak jumlah gambar komponen dari hasil Gambar 3.8. Setiap gambar akan dijadikan satu variabel inputValues yang terdiri dari nilai satu atau nol bergantung pada nilai *pixel* baris ke-k dan kolom ke-j. Setiap inputValues, akan dimasukan ke modul *Artificial Neural Network*. Dari modul tersebut akan diperoleh

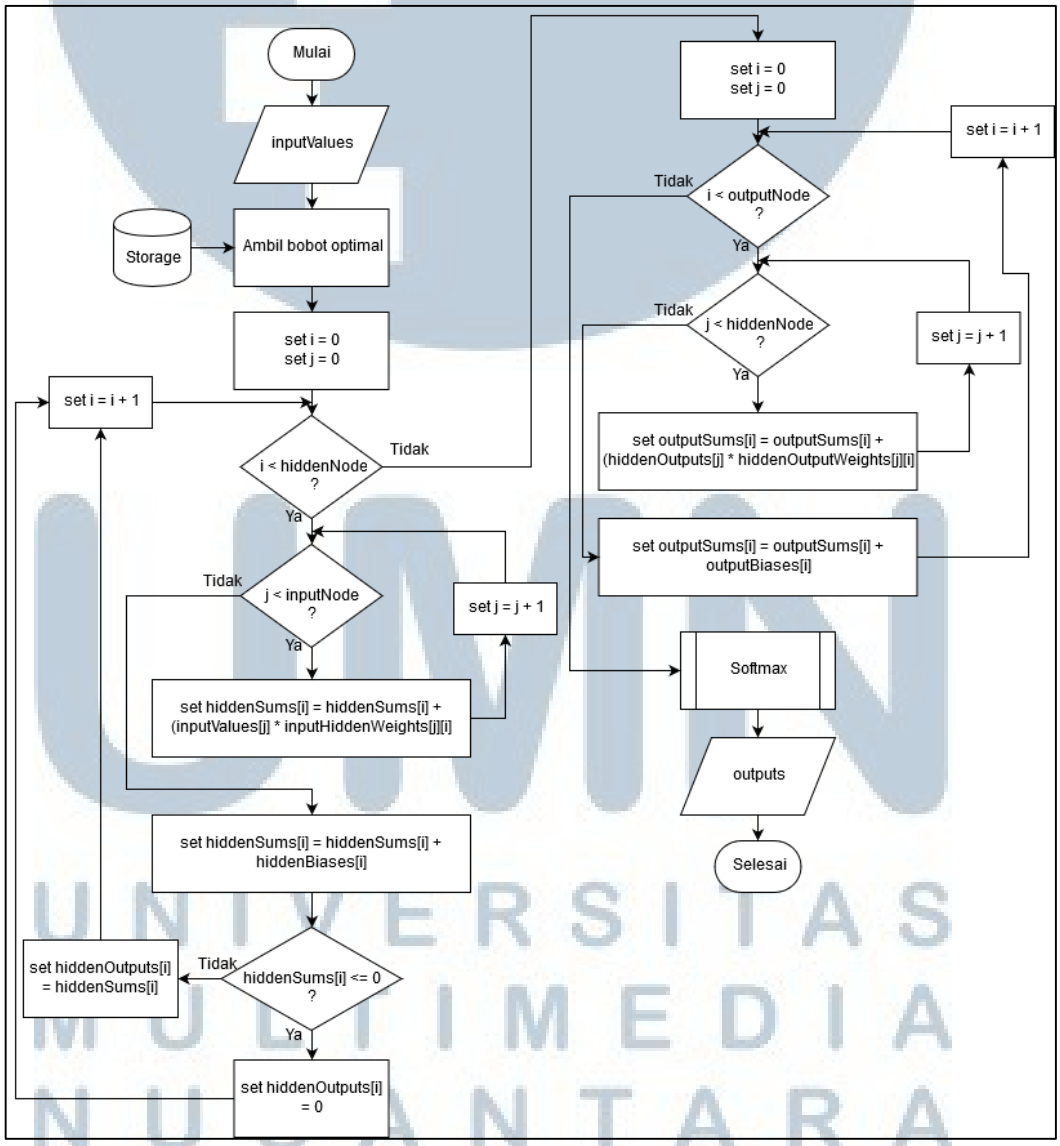
output yang berbentuk *array*. Output tersebut akan dicari index dengan nilai output maksimum untuk diklasifikasikan sebagai sebuah karakter.



Gambar 3.13 *Flowchart* Mengklasifikasikan Karakter

A.10 Flowchart Artificial Neural Network

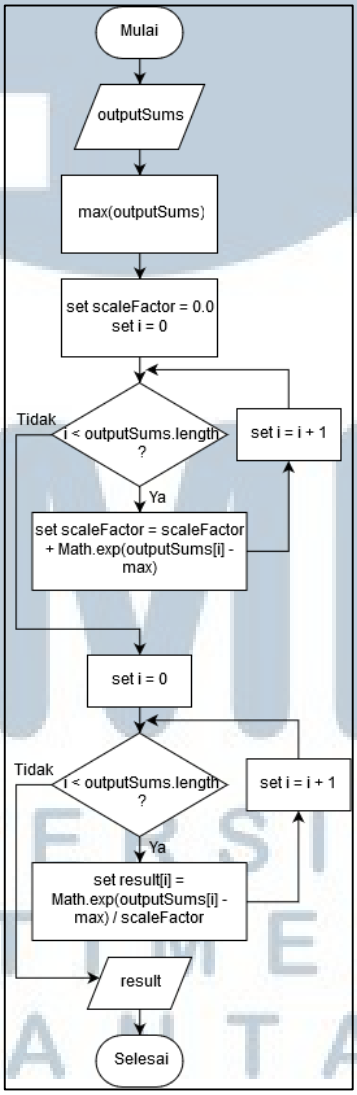
Gambar 3.14 merupakan *flowchart Artificial Neural Network* dari modul pada Gambar 3.13. Dimulai dari menerima parameter inputValues, mengambil bobot optimal dari *file* pada *storage* aplikasi, dan melakukan iterasi. Iterasi pertama dilakukan sebanyak jumlah *input node* dan *hidden node* untuk dihitung nilai hiddenSums dan diaktivasi dengan ReLU menjadi hiddenOutputs. Iterasi kedua dilakukan sebanyak jumlah *hidden node* dan *output node* untuk dihitung nilai outputSums dan diaktivasi dengan Softmax menjadi outputs.



Gambar 3.14 Flowchart Artificial Neural Network

A.11 Flowchart Softmax

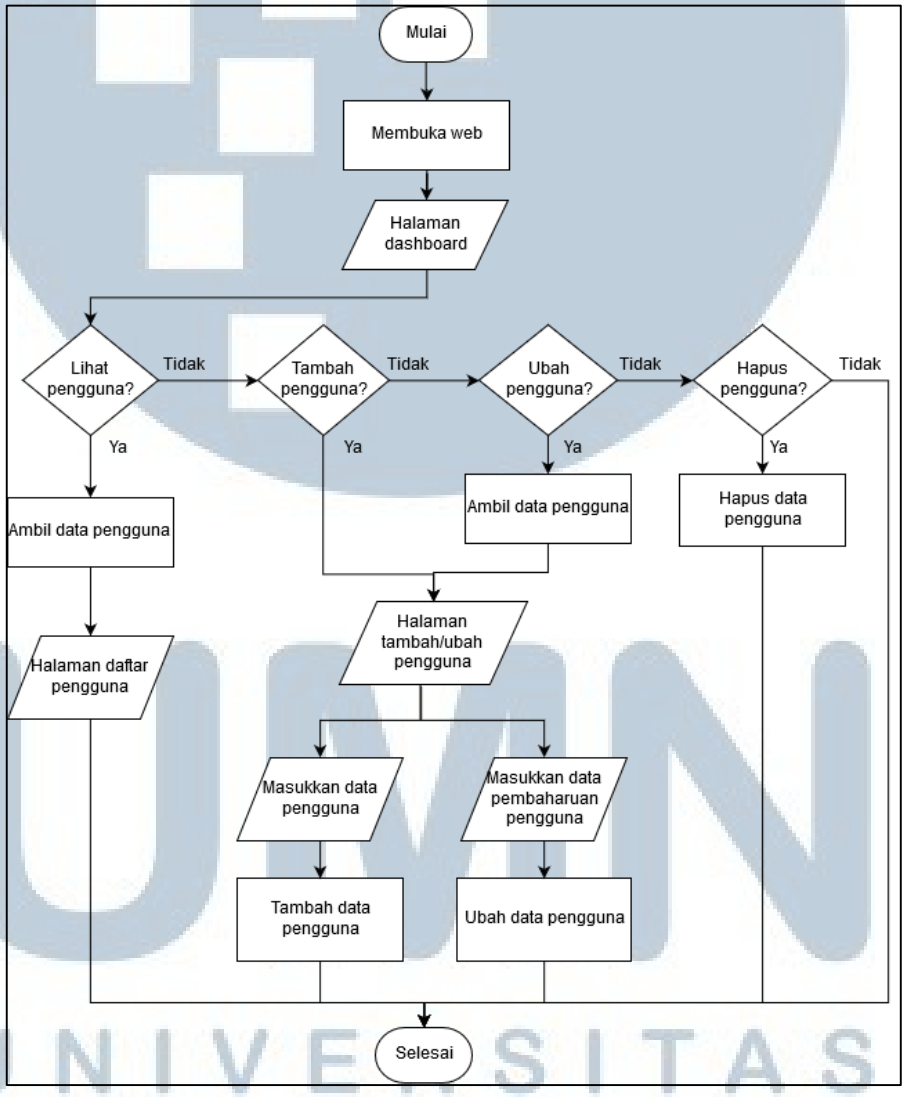
Gambar 3.15 merupakan *flowchart* fungsi aktivasi *softmax*. Proses dari *softmax* ini mendistribusikan node untuk setiap *output* node secara merata. Dimulai dengan menerima parameter outputSums, kemudian mencari nilai terbesar dari outputSums ke dalam variabel max, dan mencari nilai scaleFactor. Nilai scaleFactor didapat dari iterasi sebanyak jumlah outputSums dan menambahkan outputSums dengan nilai eksponen dari outputSums[i] dikurangi nilai max. Lalu baru bisa dihitung nilai *output* dari setiap *node* dari nilai eksponen dari outputSums[i] – max dibagi dengan scaleFactor.



Gambar 3.15 Flowchart Softmax

B. Flowchart Aplikasi Admin

Gambar 3.16 merupakan *flowchart* sistem admin. Sistem admin ini berbasis web dibuat menggunakan bahasa pemrograman PHP. Dimulai dengan tampilan *dashboard* dengan navigasi ke halaman list pengguna, tambah pengguna, dan ubah pengguna. Semua aktivitas (tambah, ubah, dan hapus) pengguna dilakukan melalui HTTP *request* ke Custom API PHP.



Gambar 3.16 *Flowchart* Admin

3.3.3 Rancangan Struktur Tabel

Tabel 3.1 Tabel Rancangan Struktur Tabel

Nama Kolom	**Tipe Data**	**Atribut**
id	int(7)	PRIMARY KEY
nik	varchar(16)	
nama_lengkap	varchar(32)	
alamat	text	NULL
kontak	varchar(13)	NULL
pekerjaan	enum('DOSEN, 'LAINNYA')	
foto	text	
pelat_nomor	varchar(9)	UNIQUE
foto_kendaraan	text	

Tabel 3.1 merupakan tabel rancangan struktur tabel pada *database*. Tabel ini memiliki 10 kolom. Kolom id bertipe data int(7) untuk menyimpan id maksimal 9.999.999 untuk karyawan UMN. Kolom nik bertipe data varchar(16) dengan tujuan menyimpan nomor induk karyawan UMN dengan maksimum 16 digit. Kolom nama_lengkap bertipe varchar(32) dengan tujuan menyimpan nama lengkap karyawan UMN maksimal tiga kata ditambah spasi dengan masing-masing kata paling banyak sepuluh huruf. Kolom alamat menggunakan tipe data Text karena keberagaman alamat mulai dari nama jalan, kota, provinsi, kelurahan, kecamatan. Kolom kontak bertipe data varchar(13) di mana tiga belas merupakan jumlah digit terbesar pada nomor *Handphone* saat ini. Kolom alamat dan kontak merupakan data *optional* karena hanya sebagai pendukung data pemilik kendaraan. Kolom foto bertipe data text untuk menyimpan URL dari foto karyawan di *server* untuk ditampilkan di aplikasi Android. Kolom pelat nomor bertipe data varchar(9) untuk menyimpan nomor kendaraan bermotor dengan dua kode wilayah depan, empat nomor kendaraan, dan tiga seri akhir wilayah. Pelat nomor memiliki atribut UNIQUE karena tidak mungkin kendaraan yang sama dimiliki lebih dari satu

orang. Kolom foto_kendaraan bertipe data text untuk menyimpan URL dari foto kendaraan pemilik untuk ditampilkan di aplikasi Android.

3.3.4 Rancangan Tampilan Antarmuka

Rancangan tampilan antarmuka ini terbagi sesuai dengan aplikasi yang dibuat yaitu aplikasi Android dan sistem admin. Tampilan antarmuka aplikasi Android terbagi menjadi lima yaitu menu utama, pindai, bantuan, info, dan hasil. Tampilan antarmuka sistem admin terbagi menjadi dua yaitu daftar pengguna dan tambah/ubah pengguna.

A. Rancangan Antarmuka Aplikasi Android

Aplikasi Android ini memiliki lima halaman utama, yaitu halaman menu utama, halaman pindai, halaman bantuan, halaman info, halaman hasil. Pada halaman menu utama akan ada navigasi ke halaman pindai, bantuan, dan halaman info.

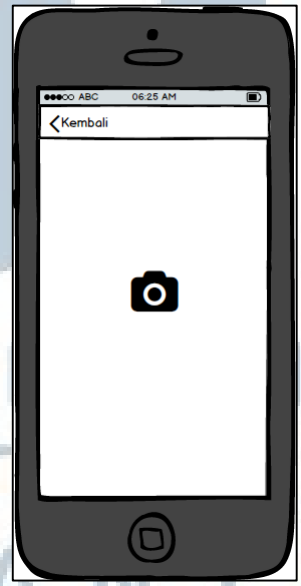
A.1 Menu Utama

Gambar 3.17 merupakan rancangan antarmuka halaman menu utama. Halaman ini berisi logo aplikasi, judul aplikasi, dan tombol untuk berpindah ke halaman pindai, halaman bantuan, atau halaman info.

Gambar 3.17 Halaman Menu Utama

A.2 Pindai

Gambar 3.18 merupakan rancangan antarmuka halaman pindai setelah tombol pindai pada halaman menu utama ditekan. Halaman ini berisi tampilan kamera yang digunakan untuk memindai gambar sekitar untuk melakukan deteksi objek pelat nomor, yang jika terdeteksi ada objek pelat, kamera akan otomatis mengambil gambar dan memproses gambar.



Gambar 3.18 Halaman Pindai

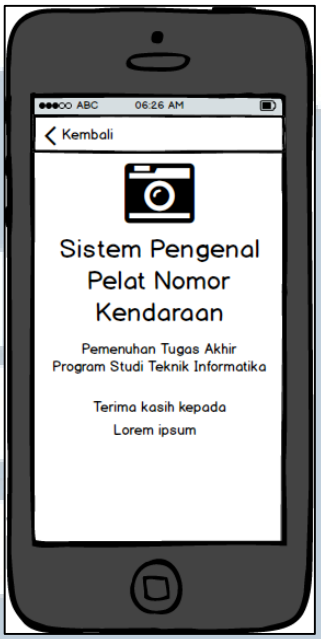
A.3 Bantuan

Gambar 3.19 merupakan rancangan antarmuka halaman bantuan. Halaman ini berisi *full page slider* berisi urutan langkah yang harus dilakukan untuk menggunakan aplikasi ini. Halaman ini dibuat agar pengguna tidak kebingungan dalam menggunakan aplikasi ini.

Gambar 3.19 Halaman Bantuan

A.4 Info

Gambar 3.20 merupakan rancangan antarmuka halaman info. Halaman info ini akan berisi logo, judul aplikasi, dan teks berisi ucapan terima kasih terhadap dukungan-dukungan dalam pembuatan aplikasi.



Gambar 3.20 Halaman Info

A.5 Hasil

Gambar 3.21 merupakan rancangan antarmuka halaman hasil setelah dilakukan pemindaian. Halaman ini berisi data pengguna beserta kendaraan yang berhasil dipindai oleh aplikasi ini.

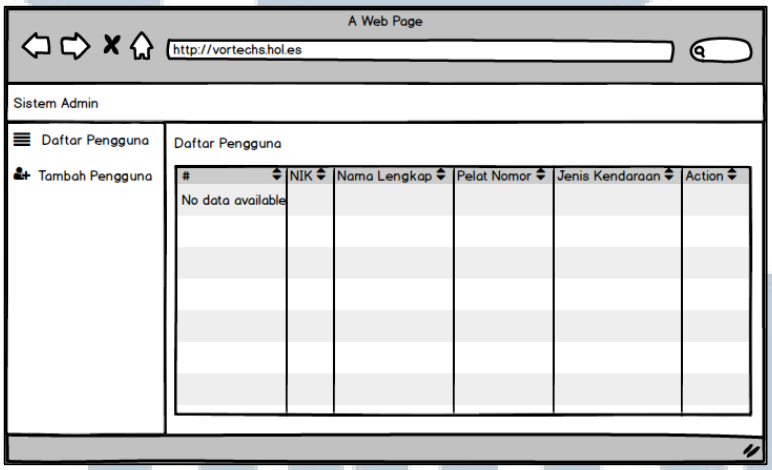
Gambar 3.21 Halaman Hasil

B. Rancangan Antarmuka Aplikasi Admin

Pada rancangan tampilan antarmuka sistem admin ini akan memiliki dua halaman secara keseluruhan yaitu halaman daftar pengguna dan halaman tambah atau ubah pengguna.

B.1 Halaman Daftar Pengguna

Gambar 3.22 merupakan rancangan antarmuka halaman daftar pengguna berisi tabel dengan data yang perlu ditampilkan untuk membedakan data satu dengan lainnya. Data tersebut terdiri dari kolom-kolom nomor, NIK, nama lengkap, pelat nomor, jenis kendaraan dan *action*. *Action* berisi *popover* untuk ke halaman tambah/edit dan hapus.

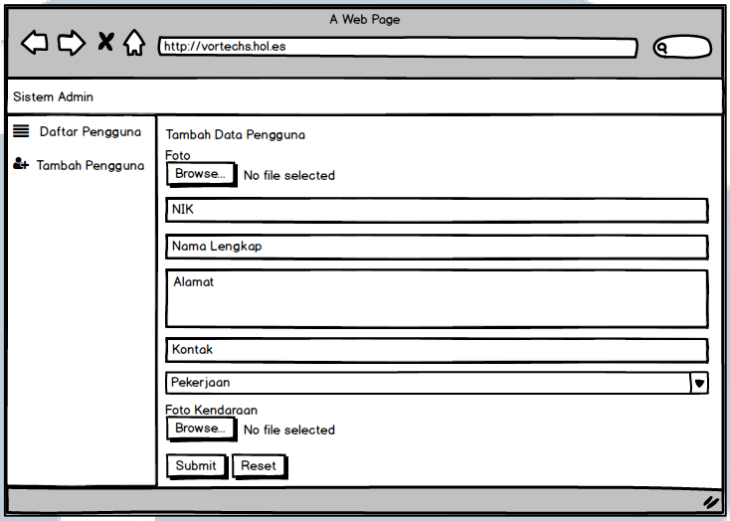


Gambar 3.22 Antarmuka Halaman Daftar Pengguna

B.2 Halaman Tambah / Ubah Pengguna

Gambar 3.23 merupakan rancangan antarmuka halaman tambah/ubah pengguna. Halaman tersebut berisi *field-field* data pengguna. Halaman tambah dan ubah berbeda pada *field* di halaman tersebut. Ketika ingin ubah akan ada data

pengguna lama sebelum data diubah, sedangkan tambah data akan berisi *field* kosong semua.



Gambar 3.23 Antarmuka Halaman Tambah / Ubah Pengguna