

**KLASIFIKASI PENYEBAB KECELAKAAN LALU LINTAS
PADA BERITA ONLINE MENGGUNAKAN METODE
*NEURAL NETWORK***

SKRIPSI

Oleh :

ABDILLAH BAGUS SAPUTRA
NIM. 210605110134



**PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2025**

**KLASIFIKASI PENYEBAB KECELAKAAN LALU LINTAS
PADA BERITA ONLINE MENGGUNAKAN METODE
*NEURAL NETWORK***

SKRIPSI

Diajukan kepada:
Universitas Islam Negeri Maulana Malik Ibrahim Malang
Untuk memenuhi Salah Satu Persyaratan dalam
Memperoleh Gelar Sarjana Komputer (S.Kom)

Oleh :
ABDILLAH BAGUS SAPUTRA
NIM. 210605110134

**PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2025**

HALAMAN PERSETUJUAN

KLASIFIKASI PENYEBAB KECELAKAAN LALU LINTAS PADA BERITA ONLINE MENGGUNAKAN METODE *NEURAL NETWORK*

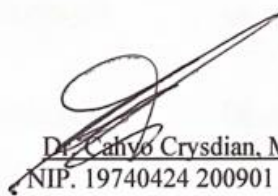
SKRIPSI

Oleh :
ABDILLAH BAGUS SAPUTRA
NIM. 210605110134

Telah Diperiksa dan Disetujui untuk Diuji:
Tanggal: 11 Juni 2025

Pembimbing I,

Pembimbing II,


Dr. Cahyo Crysdian, M.Cs
NIP. 19740424 200901 1 008


Dr. M. Imamudin Lc, MA
NIP. 19740602 200901 1 010

Mengetahui,
Ketua Program Studi Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang



Dr. Fachrul Kurniawan, M.MT., IPU
NIP. 19771020 200912 1 001

HALAMAN PENGESAHAN

KLASIFIKASI PENYEBAB KECELAKAAN LALU LINTAS PADA BERITA ONLINE MENGGUNAKAN METODE *NEURAL NETWORK*

SKRIPSI

Oleh :
ABDILLAH BAGUS SAPUTRA
NIM. 210605110134

Telah Dipertahankan di Depan Dewan Penguji Skripsi
dan Dinyatakan Diterima Sebagai Salah Satu Persyaratan
Untuk Memperoleh Gelar Sarjana Komputer (S.Kom)
Tanggal: 18 Juni 2025

Susunan Dewan Penguji

Ketua Penguji : Dr. Irwan Budi Santoso, M.Kom
NIP. 19770103 201101 1 004

Anggota Penguji I : Dr. Agung Teguh Wibowo Almais, M.T
NIP. 19860301 202321 1 016

Anggota Penguji II : Dr. Cahyo Crys dian, M.CS
NIP. 19740424 200901

Anggota Penguji III : Dr. M. Imamudin Lc, MA
NIP. 19740602 200901 1 010

Mengetahui dan Mengesahkan,
Ketua Program Studi Teknik Informatika
Fakultas Sains dan Teknologi
IAIN
Universitas Islam Negeri Maulana Malik Ibrahim Malang



Dr. H. Fachrul Kurniawan, M.MT., IPU
NIP. 19771020 200912 1 001

PERNYATAAN KEASLIAN TULISAN

Saya yang bertanda tangan di bawah ini:

Nama : Abdillah Bagus Saputra
NIM : 210605110134
Fakultas / Program Studi : Sains dan Teknologi / Teknik Informatika
Judul Skripsi : Klasifikasi Penyebab Kecelakaan Lalu Lintas Pada
Berita Online Menggunakan Metode
Neural Network

Menyatakan dengan sebenarnya bahwa Skripsi yang saya tulis ini benar-benar merupakan hasil karya saya sendiri, bukan merupakan pengambil alihan data, tulisan, atau pikiran orang lain yang saya akui sebagai hasil tulisan atau pikiran saya sendiri, kecuali dengan mencantumkan sumber cuplikan pada daftar pustaka.

Apabila dikemudian hari terbukti atau dapat dibuktikan skripsi ini merupakan hasil jiplakan, maka saya bersedia menerima sanksi atas perbuatan tersebut.

Malang, 19 Juni 2025
Yang membuat pernyataan,



Abdillah Bagus Saputra
NIM.210605110134

MOTTO

إِنَّ اللَّهَ لَا يُغَيِّرُ مَا بِقَوْمٍ حَتَّى يُغَيِّرُوا مَا بِأَنْفُسِهِمْ

“Allah tidak akan mengubah nasib suatu kaum kecuali mereka mengubah diri mereka sendiri”

HALAMAN PERSEMBAHAN

Alhamdulillah rabbil alamin, segala puji bagi Allah Yang Maha Esa atas limpahan karunia dan hidayah-Nya sehingga penulis dapat merampungkan tugas akhir ini. Shalawat dan salam senantiasa tercurah kepada Nabi Muhammad Shallallahu 'alaihi wasallam, sang pembawa cahaya yang menuntun umat manusia dari kegelapan menuju cahaya Islam yang mulia.

Karya ini penulis dedikasikan untuk ayah dan ibu tercinta, serta seluruh keluarga, teman-teman, senior, dan sahabat-sahabat yang telah menjadi bagian dari perjalanan studi penulis. Dukungan dan kebersamaan mereka menjadi kekuatan yang tak ternilai bagi penulis untuk mencapai pencapaian ini.

KATA PENGANTAR

Assalamualaikum wr wb.

Segala puji dan syukur penulis haturkan ke hadirat Allah SWT atas segala berkah dan rahmat-Nya, sehingga penulis dapat menuntaskan penyusunan skripsi dengan judul "Klasifikasi Penyebab Kecelakaan Lalu Lintas Pada Berita Online Menggunakan Metode Neural Network" secara lancar dan sesuai dengan waktu yang ditentukan.

Penulis menyadari bahwa dalam proses penyusunan skripsi ini, banyak pihak yang telah berkontribusi memberikan bimbingan, motivasi, serta dukungan baik secara moral maupun material. Oleh karena itu, penulis bermaksud menyampaikan rasa terima kasih yang sebesar-besarnya kepada:

1. Prof. Dr. H.M. Zainuddin, MA., selaku rektor Universitas Islam Negeri Maulana Malik Ibrahim Malang.
2. Prof. Dr. Sri Hariani, M.Si., selaku dekan Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang.
3. Dr. Fachrul Kurniawan, M.MT., IPU, selaku Ketua Program Studi Teknik Informatika Universitas Islam Negeri Maulana Malik Ibrahim Malang.
4. Dr. Cahyo Crysdian, MCS selaku Dosen Pembimbing I yang telah memberikan banyak dukungan dan bimbingan dalam penulisan skripsi ini.
5. Dr. M. Imamudin Lc, MA selaku Dosen Pembimbing II yang telah memberikan banyak dukungan dan bimbingan dalam penulisan skripsi ini
6. Dr. Irwan Budi Santoso, M.Kom selaku Ketua Penguji yang telah memberikan banyak saran untuk menyelesaikan skripsi ini.

7. Dr. Agung Teguh Wibowo Almais, MT selaku dosen penguji II yang telah menguji serta memberikan masukan dalam penulisan skripsi ini.
8. Seluruh Dosen dan Jajaran Staf Program Studi Teknik Informatika yang telah memberikan banyak bantuan dalam skripsi ini.
9. Kedua orang tua, Bapak Supriyadi dan Ibu Sobatun Nikmah serta adik saya yang telah memberikan dukungan kepada penulis untuk menyelesaikan skripsi ini.
10. Teman-teman Angkatan 2021 Teknik Informatika “ASTER” yang telah memberikan banyak bantuan baik material maupun dukungan intelektual, semangat, serta motivasi dalam menyelesaikan penyusunan skripsi ini.
11. Untuk teman-teman terbaik di kelas E, terima kasih telah menjadi alasan saya tetap waras di tengah tekanan skripsi. Untuk setiap sesi '*healing*' dadakan yang menyegarkan pikiran, setiap pertandingan badminton yang membakar stres, dan untuk segala tawa yang tak ada habisnya hingga membuat lupa sejenak akan revisian.
12. Semua pihak yang tidak dapat disebutkan satu per satu, termasuk responden yang telah memberikan kontribusi, saran, dan dukungan dalam perjalanan penulisan skripsi ini.

Malang, 25 Juni 2025

Penulis

DAFTAR ISI

HALAMAN PENGAJUAN	ii
HALAMAN PERSETUJUAN	iii
HALAMAN PENGESAHAN	iv
PERNYATAAN KEASLIAN TULISAN	v
MOTTO	vi
HALAMAN PERSEMBAHAN	vii
KATA PENGANTAR	viii
DAFTAR ISI	x
DAFTAR GAMBAR	xii
DAFTAR TABEL	xiii
ABSTRAK	xiv
ABSTRACT	xv
المخلص	xvi
BAB I PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Pernyataan Masalah	5
1.3 Batasan Masalah	5
1.4 Tujuan Penelitian	6
1.5 Manfaat Penelitian	6
BAB II STUDI PUSTAKA	7
2.1 Klasifikasi Berita	7
BAB III DESAIN DAN IMPLEMENTASI	13
3.1 Pengumpulan Data	13
3.2 Desain Sistem	16
3.2.1 <i>Preprocessing</i>	17
3.2.2 Ekstraksi Fitur	18
3.2.3 <i>Synthetic Minority Oversampling Technique (SMOTE)</i>	21
3.2.4 <i>Artificial Neural Network</i>	26
3.3 Implementasi Sistem	31
3.3.1 <i>Proses Training</i>	32
BAB IV UJI COBA DAN PEMBAHASAN	40
4.1 Skenario Uji Coba	40
4.2 Hasil Uji Coba	46
4.2.1 Skenario 1	47
4.2.2 Skenario 2	49
4.2.3 Skenario 3	51
4.2.4 Skenario 4	53
4.3 Hasil Uji Coba dengan SMOTE	55
4.3.1 Skenario 1	56
4.3.2 Skenario 2	58
4.3.3 Skenario 3	60
4.3.4 Skenario 4	61
4.4 Pembahasan	64

4.4.1 Pengaruh Konfigurasi Parameter	65
4.4.2 Kemampuan Model dalam Klasifikasi	66
4.4.3 Efisiensi Komputasi dan Konvergensi	68
4.4.4 Integrasi Ayat dan Hadis	71
BAB V KESIMPULAN DAN SARAN	76
5.1 Kesimpulan	76
5.2 Saran	77
DAFTAR PUSTAKA	

DAFTAR GAMBAR

Gambar 3.1 Flowchart Scrapping Data.....	14
Gambar 3.2 Diagram Blok Desain System	16
Gambar 3.3 Diagram Blok Preprocessing.....	17
Gambar 3.4 <i>Flowchart FastText</i>	19
Gambar 3.5 <i>Flowchart SMOTE</i>	23
Gambar 3.6 Arsitektur Neural Network.....	27
Gambar 3.7 <i>Flowchart</i> Proses <i>Feedforward</i>	30
Gambar 3.8 <i>Flowchart</i> Proses <i>Training</i>	32
Gambar 3.9 <i>Flowchart</i> Proses <i>Backpropagation</i>	38
Gambar 4.1 Grafik Loss Skenario 1	47
Gambar 4.2 Confusion Matrix Skenario 1	48
Gambar 4.3 Grafik Loss Skenario 2	49
Gambar 4.4 Confusion Matrix Skenario 2	50
Gambar 4.5 Grafik Loss Skenario 3	51
Gambar 4.6 Confusion Matrix Skenario 3	52
Gambar 4.7 Grafik Loss Skenario 4	53
Gambar 4.8 Confusion Matrix Skenario 4.....	54
Gambar 4.9 Grafik Loss Skenario 1 dengan SMOTE	57
Gambar 4.10 Confusion Matrix Skenario Uji Coba 1 dengan SMOTE	57
Gambar 4.11 Grafik Loss Skenario 2 dengan SMOTE	58
Gambar 4.12 Confusion Matrix Skenario 2 dengan SMOTE.....	59
Gambar 4.13 Grafik Loss Skenario 3 dengan SMOTE	60
Gambar 4.14 Confusion Matrix Skenario 3 dengan SMOTE.....	61
Gambar 4.15 Grafik Loss Skenario 4 dengan SMOTE	62
Gambar 4.16 Confusion Matrix Skenario 4 dengan SMOTE.....	63
Gambar 4.17 Bar Chart Perbandingan Akurasi	65
Gambar 4.18 Perbandingan Waktu Pelatihan	68
Gambar 4.19 Perbandingan Loss Konvergen tanpa SMOTE	69
Gambar 4.20 Perbandingan Loss Konvergen dengan SMOTE	70

DAFTAR TABEL

Tabel 3.1 Contoh Case Folding	17
Tabel 3.2 Contoh Tokenization.....	18
Tabel 3.3 Contoh Stopword Removal.....	18
Tabel 3.4 Contoh Embedding Kata.....	20
Tabel 3.5 Contoh Representasi Berita	21
Tabel 4.1 Skenario Uji Coba.....	41
Tabel 4.2 Confusion Matrix	43
Tabel 4.3 Confusion Matrix Kelas Faktor Jalan	44
Tabel 4.4 Confusion Matrix Kelas Faktor Kendaraan	44
Tabel 4.5 Confusion Matrix Kelas Kelalaian Pengemudi.....	44
Tabel 4.6 Hasil Training	46
Tabel 4.7 Hasil Evaluasi	47
Tabel 4.8 Hasil Training dengan SMOTE	55
Tabel 4.9 hasil Evaluasi dengan SMOTE	56
Tabel 4. 10 Perbandingan Performa Model per Kelas	67

ABSTRAK

Saputra, Abdillah Bagus. 2025. **KLASIFIKASI PENYEBAB KECELAKAAN LALU LINTAS PADA BERITA ONLINE MENGGUNAKAN METODE *NEURAL NETWORK***. Skripsi. Program Studi Teknik Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang. Pembimbing: (I) Dr. Cahyo Crysodian, M.Cs (II) Dr. M. Imamudin Lc, MA.

Kata kunci: *Artificial Neural Network*, *FastText*, SMOTE, klasifikasi berita, kecelakaan lalu lintas, ketidakseimbangan kelas

Keselamatan lalu lintas masih menjadi tantangan utama di Indonesia dengan 79.220 kecelakaan tercatat per Agustus 2024, dimana 76,42% melibatkan sepeda motor. Penelitian ini bertujuan mengukur kinerja model *Artificial Neural Network* (ANN) dalam mengklasifikasikan berita kecelakaan lalu lintas online berdasarkan metrik *accuracy*, *precision*, *recall*, dan *F1-Score*. Data penelitian berupa berita kecelakaan lalu lintas dari Detik, Kompas, dan CNN Indonesia periode 2020-2025 yang dikumpulkan melalui *web scraping* dan diklasifikasikan menjadi tiga kategori: Kelalaian Pengemudi, Faktor Kendaraan, dan Faktor Jalan. *Preprocessing* data meliputi *case folding*, tokenisasi, dan *stopword removal*. Ekstraksi fitur menggunakan *FastText* untuk representasi numerik, serta teknik SMOTE untuk mengatasi ketidakseimbangan kelas. Arsitektur ANN menggunakan *hidden layer* dengan fungsi aktivasi *ReLU* dan *Softmax*. Pengujian dilakukan dalam empat skenario dengan variasi jumlah *node hidden layer* (21 dan 301) dan *Learning rate* (0.001 dan 0.0001). Model tanpa SMOTE mencapai akurasi tertinggi 78% namun mengalami kesulitan dalam mendeteksi kelas minoritas. Penerapan SMOTE meningkatkan kinerja secara signifikan dengan akurasi tertinggi 82%, *precision* 0.72, *recall* 0.73, dan *F1-Score* 0.71 pada konfigurasi 301 *node* dengan *learning rate* 0.001. Penelitian ini menunjukkan bahwa model ANN dengan kombinasi *FastText* efektif untuk klasifikasi berita kecelakaan lalu lintas, dan penerapan teknik SMOTE sangat diperlukan untuk mengatasi masalah ketidakseimbangan data guna mencapai performa optimal.

ABSTRACT

Saputra, Abdillah Bagus. 2025. **CLASSIFICATION OF TRAFFIC ACCIDENT CAUSES IN ONLINE NEWS USING *NEURAL NETWORK* METHOD.**

Undergraduate Thesis. Department of Informatics Engineering, Faculty of Science and Technology, State Islamic University of Maulana Malik Ibrahim Malang. Supervisors: (I) Dr. Cahyo Crysdiyan, M.Cs (II) Dr. M. Imamudin Lc, MA.

Key words: Artificial Neural Network, FastText, SMOTE, news classification, traffic accidents, class imbalance

Traffic safety remains a significant challenge in Indonesia, with 79,220 recorded accidents involving motorcycles as of August 2024, 76.42%. This study aims to measure the performance of Artificial Neural Network (ANN) models in classifying online traffic accident news based on accuracy, precision, recall, and F1-Score metrics. The research data consists of traffic accident news from Detik, Kompas, and CNN Indonesia from 2020-2025 collected through web scraping and classified into three categories: Driver Negligence, Vehicle Factors, and Road Factors. Data preprocessing includes case folding, tokenization, and stopword removal, while feature extraction uses FastText for numerical representation and the SMOTE technique to address class imbalance. The ANN architecture employs hidden layers with ReLU and Softmax activation functions, tested in four scenarios with variations in the number of hidden layer nodes (21 and 301) and *Learning rates* (0.001 and 0.0001). Models without SMOTE achieved a maximum accuracy of 78% but struggled with minority class detection. SMOTE implementation substantially improved performance with the highest accuracy of 82%, precision of 0.72, recall of 0.73, and F1-Score of 0.71 using 301 nodes with a 0.001 *Learning rate*. This research demonstrates that ANN models combined with FastText effectively classify traffic accident news. Implementing the SMOTE technique is essential to address data imbalance issues for optimal performance.

الملخص

سافوترا، عبد الله باغوس. 2025. تصنيف أسباب حوادث المرور في الأخبار عبر الإنترنت باستخدام طريقة الشبكات العصبية الاصطناعية. الرسالة جامعية. قسم هندسة الحاسوب، كلية العلوم والتكنولوجيا، بجامعة مولانا مالك إبراهيم الإسلامية الحكومية مالانج. المشرفون: (1) الدكتور جاهيو كريسديان، دكتوراه في علوم الكمبيوتر (2) الدكتور محمد إمام الدين، ماجستير

الكلمات المفتاحية: الشبكة العصبية الاصطناعية، *FastText*، SMOTE، تصنيف الأخبار، الحوادث المرورية، اختلال التوازن الطبقي

لا تزال السلامة المرورية تمثل تحديًا كبيرًا في إندونيسيا حيث تم تسجيل 79,220 حادثًا حتى أغسطس 2024، منها 76,42% منها تتعلق بالدراجات النارية. تهدف هذه الدراسة إلى قياس أداء نموذج الشبكة العصبية الاصطناعية (ANN) في تصنيف أخبار الحوادث المرورية عبر الإنترنت بناءً على مقاييس الدقة والدقة والاستدعاء و *F1-Score*. بيانات البحث هي في شكل أخبار الحوادث المرورية من ديتيك وكومباس وسي إن إن إندونيسيا للفترة 2020-2025 التي تم جمعها من خلال كشط الويب وتصنيفها إلى ثلاث فئات: إهمال السائق، وعوامل المركبة، وعوامل الطريق. وتتضمن المعالجة المسبقة للبيانات طي الحالات وترميزها وإزالة كلمات التوقف. يستخدم استخلاص الميزات تقنية *FastText* للتمثيل العددي، بالإضافة إلى تقنية SMOTE لمعالجة اختلال التوازن بين الفئات. تستخدم بنية الشبكة العصبية الاصطناعية طبقة مخفية مع دوال تنشيط *ReLU* و *Softmax*. تم إجراء الاختبار في أربعة سيناريوهات مع اختلافات في عدد عقد الطبقة المخفية (21 و 301) ومعدل التعلم (0.001 و 0.0001). حقق النموذج بدون SMOTE أعلى دقة بنسبة 78% ولكنه واجه صعوبة في اكتشاف فئات الأقلية. أدى تطبيق SMOTE إلى تحسين الأداء بشكل كبير مع أعلى دقة بنسبة 82%، ودقة 0.72، واستدعاء 0.73، ودرجة *F1-Score* 0.71 في تكوين 301 عقدة بمعدل تعلم 0.001. تظهر هذه الدراسة أن نموذج ANN المدمج مع *FastText* فعال في تصنيف أخبار الحوادث المرورية، وأن تطبيق تقنية SMOTE ضروري للتغلب على مشكلة عدم توازن البيانات من أجل تحقيق الأداء الأمثل.

BAB I

PENDAHULUAN

1.1 Latar Belakang

Keselamatan lalu lintas masih menjadi salah satu tantangan utama di Indonesia. Meningkatnya jumlah kendaraan dan kepadatan lalu lintas berkontribusi terhadap tingginya angka kecelakaan. Selain itu, infrastruktur jalan yang belum memadai menjadi faktor utama yang memperburuk situasi ini. Lemahnya penegakan hukum serta perilaku pengendara yang kurang disiplin juga turut memperparah kondisi tersebut (Anggraini et al. 2023). Berdasarkan data terbaru dari Korlantas Polri per 5 Agustus 2024, tercatat 79.220 kecelakaan lalu lintas di Indonesia dengan 76.42% melibatkan sepeda motor. Dampaknya sangat signifikan dengan 117.962 orang menjadi korban. Korban tersebut terdiri dari 7.21% meninggal dunia, 8.26% mengalami luka berat, dan 84.51% mengalami luka ringan. Oleh karena itu, perlu ada strategi baru yang tidak hanya bersifat teknis, tetapi juga mampu mengintegrasikan aspek edukatif dan teknologi informasi.

Kesadaran masyarakat dalam berlalu lintas masih menjadi tantangan tersendiri. Banyak pengendara masih sering melanggar aturan seperti menerobos lampu merah, melampaui batas kecepatan, atau menggunakan ponsel saat berkendara (Ahmed and Ahmed 2021). Dalam Islam, prinsip menjaga keselamatan diri dan orang lain sangat ditekankan, sebagaimana disampaikan dalam hadis:

عَنْ أَبِي سَعِيدٍ سَعْدُ بْنُ سِنَانٍ الْخُدْرِيِّ رَضِيَ اللَّهُ عَنْهُ أَنَّ رَسُولَ اللَّهِ صَلَّى اللَّهُ عَلَيْهِ وَسَلَّمَ قَالَ : لَا ضَرَرَ وَلَا ضِرَارَ

"Tidak boleh membahayakan diri sendiri dan tidak boleh pula membahayakan orang lain." (HR. Ibnu Majah).

Imam An-Nawawi menjelaskan bahwa hadis ini merupakan salah satu qaidah kubra (kaidah besar) dalam syariat Islam yang mencakup larangan berbuat madharat terhadap diri sendiri maupun orang lain (*Syarah Al-Arba'in An-Nawawiyyah*). Ibnu Daqiq Al-'Eid menjelaskan perbedaan antara "dharar" dan "dhirar". Dharar adalah kerugian secara umum sedangkan dhirar adalah perbuatan saling merugikan secara sengaja (*Syarah Al-Arba'in An-Nawawiyyah*, hlm. 106). Etika berlalu lintas yang berakar pada nilai-nilai agama dapat menjadi dasar moral dalam menciptakan budaya berkendara yang aman. Dalam Al-Qur'an juga dijelaskan bahwa manusia tidak boleh menjerumuskan dirinya ke dalam kebinasaan, sebagaimana disebutkan dalam QS. Al-Baqarah: 195:

وَأَنْفِقُوا فِي سَبِيلِ اللَّهِ وَلَا تُلْقُوا بِأَيْدِيكُمْ إِلَى التَّهْلُكَةِ وَأَحْسِنُوا إِنَّ اللَّهَ يُحِبُّ الْمُحْسِنِينَ ﴿١٩٥﴾

"Berinfaklah di jalan Allah, janganlah jerumuskan dirimu ke dalam kebinasaan, dan berbuatbaiklah. Sesungguhnya Allah menyukai orang-orang yang berbuat baik." (QS. Al-Baqarah: 195).

Imam As-Suyuthi dalam *Tafsirul Jalalain* menjelaskan bahwa ayat tersebut mengandung perintah untuk berinfak di jalan Allah. Perintah ini mencakup taat kepada-Nya dalam urusan jihad dan aspek lainnya. Selain itu, ayat ini juga mengingatkan umat Islam untuk berbuat ihsan dalam berinfak. Ayat ini juga mengandung peringatan untuk menghindari kehancuran dengan menahan diri dari berinfak untuk jihad yang dapat memperkuat musuh (*Tafsirul Jalalain* pada *Hasyiyatus Shawi*, Juz I, hal. 119). Ayat ini tidak hanya mengingatkan pentingnya menjaga keselamatan pribadi, tetapi juga mendorong manusia untuk melakukan kebaikan demi keselamatan bersama. Dalam konteks modern, kontribusi ini dapat

diwujudkan melalui inovasi teknologi yang mendukung keselamatan lalu lintas seperti pengembangan sistem klasifikasi berita kecelakaan berbasis *machine learning*. *Natural Language Processing* (NLP) dan *machine learning* telah banyak diterapkan dalam klasifikasi berita kecelakaan untuk mengatasi tantangan ini (Malaviya et al. 2024). Salah satu metode *machine learning* yang efektif dalam konteks ini adalah Neural Network karena mampu mengenali pola tersembunyi dalam data teks yang kompleks dan tidak linear (Goodfellow et al. 2016). Keunggulan metode ini juga telah dibuktikan dalam berbagai studi. Penelitian menggunakan ANN untuk klasifikasi berita online berbahasa Indonesia memperoleh akurasi 96.29% (Hasanah, 2022), sedangkan data pelatihan menggunakan ANN untuk klasifikasi berita olahraga mencatat akurasi 99% Ridwan dkk.,(2021). Kombinasi *deep learning* lainnya seperti CNN-LSTM juga menunjukkan kinerja tinggi, seperti yang diperoleh F1-Score 93.27% pada penelitian (Widhiyasana et al. 2021). Temuan ini memperlihatkan bahwa model berbasis jaringan saraf mampu memahami konteks kalimat dan relasi antar kata lebih baik daripada metode klasik.

Namun, penting untuk dicatat bahwa penggunaan Neural Network dalam klasifikasi berita relatif jarang dibandingkan metode lain seperti SVM, KNN, atau *Naïve Bayes*. Sebagai contoh, penelitian menggunakan *Support Vector Machine* (SVM) untuk klasifikasi berita Bahasa Indonesia mencapai akurasi 94.24% (Irham dkk. 2019). Sementara itu, penggunaan *K-Nearest Neighbor* (KNN) untuk klasifikasi berita online mencapai akurasi 87% (Juwita et al., 2022). Meskipun metode-metode tersebut cukup efektif, mereka memiliki keterbatasan dalam

menangani data teks yang kompleks dan non-linear. Salah satu teknik *embedding* yang cukup menjanjikan untuk memperkuat performa klasifikasi teks adalah *FastText*. *FastText* dikembangkan oleh Facebook AI Research dan mampu merepresentasikan kata dalam bentuk vektor berdimensi tetap berdasarkan sub-kata (n-gram). Hal ini membuat *FastText* lebih tangguh dalam menghadapi kata-kata baru. Misalnya, kombinasi *FastText* dan *Bi-LSTM* berhasil mencapai akurasi 97% dalam mendeteksi kecelakaan lalu lintas dari data media sosial (Ali et al. 2021). Hasil ini menunjukkan potensi besar *FastText* dalam mengelola informasi kecelakaan secara sistematis.

Penelitian ini penting untuk dilakukan karena dapat memberikan wawasan baru bagi berbagai pemangku kepentingan. Pemangku kepentingan tersebut antara lain kepolisian, Jasa Raharja, dan Dinas Perhubungan. Misalnya, kepolisian dapat menggunakan hasil klasifikasi ini untuk menganalisis penyebab kecelakaan secara cepat dan efisien. Di sisi lain, Jasa Raharja dapat memanfaatkan informasi ini untuk mengidentifikasi pola kecelakaan yang dapat digunakan dalam pengambilan keputusan terkait santunan atau pencegahan kecelakaan (Prasetya dan Ferdiansyah 2022). Penelitian ini juga relevan dengan perkembangan teknologi NLP, di mana metode seperti Neural Network telah terbukti efektif dalam menangani data teks berbahasa Indonesia dengan akurasi mencapai 94.24% (Irham dkk. 2019). Dengan demikian, penelitian ini diharapkan dapat memberikan kontribusi nyata dalam meningkatkan keselamatan lalu lintas melalui analisis data berita kecelakaan.

1.2 Pernyataan Masalah

Seberapa baik kinerja model *Artificial Neural Network* dalam mengklasifikasikan berita online, berdasarkan metrik *Accuracy*, *precision*, *Recall*, dan *F1-Score*?

1.3 Batasan Masalah

- a. Data yang digunakan berasal dari sumber berita online: **Detik, Kompas, CNN Indonesia**. Sumber data ini dipilih karena reputasi kredibilitasnya dalam melaporkan insiden kecelakaan lalu lintas. Data dari platform ini juga mencakup berbagai kasus kecelakaan, sehingga memberikan gambaran yang komprehensif (Bintari dkk. 2019).
- b. Data yang dianalisis berupa **teks berita**, tidak termasuk audio atau video. Audio dan video tidak termasuk dalam analisis karena memerlukan teknik yang lebih kompleks seperti *speech-to-text* atau pengolahan video yang di luar cakupan penelitian ini.
- c. Data yang dianalisis menggunakan teks **berbahasa Indonesia**. Bahasa Indonesia dipilih untuk memastikan relevansi budaya dan konteks yang lebih tepat dalam klasifikasi penyebab kecelakaan yang terjadi di negara tersebut. Campuran bahasa dapat mengurangi akurasi model dalam memahami makna kata, terutama jika model tidak dirancang untuk mengelola *multilingual text* (Takawane et al. 2023).
- d. Untuk menghindari **duplikasi informasi**, dilakukan proses penghapusan data/berita yang sama dengan cara membandingkan judul berita yang memiliki kemiripan tinggi dan memperhatikan rentang waktu unggah

maksimal 2 minggu. Jika terdapat dua atau lebih berita dengan judul yang hampir identik dan dipublikasikan dalam rentang waktu tersebut, maka hanya satu berita yang dipertahankan. Langkah ini bertujuan untuk menghindari bias dalam analisis akibat pengulangan kasus yang sama dari sumber berita berbeda.

1.4 Tujuan Penelitian

Mengukur kinerja model *Artificial Neural Network* dalam melakukan klasifikasi berita online, dengan mengevaluasi metrik *Accuracy*, *precision*, *Recall*, dan *F1-Score*.

1.5 Manfaat Penelitian

Hasil dari penelitian ini diharapkan dapat memberikan manfaat kepada berbagai pihak, di antaranya:

1. Pihak kepolisian: untuk membantu analisis penyebab kecelakaan secara cepat dan efisien.
2. Jasa Raharja: untuk mengidentifikasi pola kecelakaan yang dapat digunakan dalam pengambilan keputusan terkait santunan atau pencegahan kecelakaan.
3. Bagi Dinas Perhubungan: Menyediakan data analitik yang mendalam mengenai faktor penyebab kecelakaan untuk perencanaan kebijakan lalu lintas yang lebih aman.

BAB II

STUDI PUSTAKA

2.1 Klasifikasi Berita

Penelitian oleh Irham dkk., mengklasifikasikan berita dalam Bahasa Indonesia menggunakan teknik *preprocessing* (*stopword removal* dan *stemming*), seleksi fitur MI, dan metode SVM yang menghasilkan akurasi optimal 94,24% pada skenario tertentu (Irham dkk. 2019). Penelitian dalam lima tahun terakhir yang terkait dengan klasifikasi berita khususnya menggunakan SVM, relevan untuk memahami perkembangan metode klasifikasi otomatis dalam teks Bahasa Indonesia.

Penelitian oleh Hanifa et al., klasifikasi berita palsu berbahasa Indonesia dilakukan dengan membandingkan dua model RNN, yaitu LSTM dan *Gated Recurrent Unit* (GRU) (Hanifa et al. 2021). Pencarian parameter terbaik dilakukan guna menghasilkan kinerja klasifikasi yang optimal. Data yang digunakan adalah berita-berita terkini yang berbahasa Indonesia terkait berbagai kejadian di Indonesia. Parameter optimal yang diperoleh untuk model LSTM mencakup jumlah *epochs* sebanyak 15, fungsi optimizer dengan metode *Gradient descent* yaitu *rmsprop*, dan ukuran batch sebesar 64. Berdasarkan konfigurasi tersebut, model LSTM mencapai akurasi sebesar 73% sedangkan model GRU menghasilkan akurasi sebesar 64%.

Penelitian oleh Ali et al., mengembangkan sistem deteksi kecelakaan lalu lintas menggunakan data dari media sosial dengan mengimplementasikan teknik

pemrosesan bahasa alami dan model *machine learning* seperti OLDA dan Bi-LSTM (Ali et al. 2021). Sistem ini memanfaatkan teknik *crawling* untuk mengumpulkan data secara *real-Time* dari platform Twitter dan Facebook yang kemudian diproses dengan langkah-langkah *preprocessing* seperti tokenisasi, penghapusan *stopword*, dan *lemmatization*. Ekstraksi fitur dalam penelitian ini menggunakan *FastText* untuk representasi kata yang memungkinkan pemrosesan kata-kata formal dan informal dari teks media sosial dalam vektor berdimensi rendah. Selain itu, OLDA digunakan untuk pemodelan topik otomatis guna memberi label pada data yang berkaitan dengan kecelakaan lalu lintas sementara Bi-LSTM diterapkan untuk menganalisis kondisi kecelakaan berdasarkan sentimen yang terdeteksi dalam data tersebut. Hasil evaluasi menunjukkan bahwa sistem ini mencapai akurasi sebesar 97%, lebih unggul dibandingkan dengan pendekatan sistem deteksi kecelakaan sebelumnya. Dengan menggunakan pendekatan ini, penelitian ini berhasil mengatasi tantangan yang dihadapi oleh sistem berbasis sensor dan memberikan solusi untuk mendeteksi kecelakaan lalu lintas secara efisien dan akurat menggunakan data tidak terstruktur dari media sosial.

Penelitian oleh Rizal, berfokus pada perkembangan pemberitaan olahraga di portal berita online berbasis web yang semakin diminati masyarakat (Rizal 2021). Untuk mengklasifikasikan berita olahraga digunakan metode *K-Nearest Neighbour* (KNN) yang mengategorikan objek berdasarkan jarak terdekatnya dengan data pembelajaran. Di sisi lain, *Levenshtein Distance* diterapkan untuk mengukur perbedaan antara *string* sumber dan target dalam proses pengujian. Penelitian ini melibatkan 1000 data latih dan 60 data uji dengan tiap berita uji diuji berdasarkan

kemiripan kalimat dengan data latih. Hasil pengujian menunjukkan bahwa akurasi sistem bervariasi: akurasi 100% diperoleh jika semua kalimat berita uji sesuai dengan kategori olahraga, sedangkan akurasi menurun menjadi 48,3% dan 13,3% saat tingkat kemiripan kalimat mencapai 50% dan 0%.

Penelitian oleh Ridwan dkk., menyoroti tingginya minat masyarakat terhadap berita olahraga yang kini telah berkembang menjadi salah satu bentuk hiburan (Ridwan dkk. 2021). Berita olahraga mencakup berbagai cabang olahraga mulai dari olahraga bola seperti sepak bola hingga olahraga otomotif seperti Formula 1 serta aspek-aspek lain seperti manajerial klub, hasil pertandingan, cedera pemain, dan lainnya. Namun, keragaman konten ini dapat menyebabkan tumpang tindih data di dalam dokumen berita di mana kalimat-kalimat dari topik berbeda bercampur menjadi satu. Hal ini menyulitkan pembaca yang ingin fokus pada topik tertentu dalam berita. Penelitian ini menggunakan *Artificial Neural Network* (ANN) dengan metode pembelajaran *backpropagation* untuk mengklasifikasikan kalimat berdasarkan makna. Fitur yang digunakan adalah frekuensi kemunculan istilah dan hasil perhitungan distribusi istilah dalam kalimat terkait. Hasil pengujian menunjukkan akurasi 99% pada data pelatihan dan 57% pada data uji, menandakan potensi ANN dalam meningkatkan klasifikasi kalimat dalam berita olahraga.

Penelitian oleh Widhiyasana et al., mengeksplorasi penerapan kombinasi metode *deep learning*, yaitu CNN dan LSTM (*Convolutional Long Short-Term Memory* atau C-LSTM) untuk klasifikasi teks berita berbahasa Indonesia (Widhiyasana et al. 2021). Data berita dikumpulkan dari portal berita dan dikelompokkan dalam tiga kategori: nasional, internasional, dan regional.

Penelitian ini menguji tiga variabel, yaitu jumlah dokumen, ukuran *batch*, dan nilai *Learning rate*. Hasil eksperimen menunjukkan bahwa metode C-LSTM mencapai nilai *F1-Score* sebesar 93,27%, lebih tinggi dibandingkan CNN (89,85%) dan LSTM (90,87%), menunjukkan bahwa kombinasi C-LSTM lebih efektif dalam klasifikasi teks berita dibandingkan penggunaan CNN atau LSTM secara terpisah.

Penelitian oleh Afandi et al., membahas klasifikasi judul berita *clickbait* dan *non-clickbait* menggunakan arsitektur RNN-LSTM (Afandi et al. 2022). Seiring berkembangnya teknologi, banyak media daring yang memanfaatkan platform digital untuk meningkatkan lalu lintas dan keuntungan dengan menyebarkan berita yang menarik perhatian sering kali melalui judul yang berlebihan atau tidak sesuai dengan isi berita sebenarnya fenomena yang dikenal sebagai "*clickbait*." Penelitian ini bertujuan membedakan antara judul berita yang sensasional dan yang autentik dengan pendekatan RNN-LSTM. Hasil penelitian menunjukkan bahwa model RNN-LSTM berhasil mencapai akurasi sebesar 79% pada data pelatihan dan 77% pada data uji, memperlihatkan efektivitas model dalam mendeteksi judul *clickbait*. Penelitian ini diharapkan dapat berkontribusi dalam pengembangan sistem deteksi *clickbait* otomatis untuk membantu pembaca mengakses berita yang lebih kredibel.

Penelitian oleh Hasanah, mengembangkan sistem klasifikasi berita online menggunakan metode *Artificial Neural Network* (ANN) dengan pendekatan *Natural Language Processing* (NLP) (Hasanah 2022). Penelitian ini menggunakan 360 data artikel berita yang diklasifikasikan dalam sembilan kategori, seperti politik, teknologi, dan lainnya. Hasil pengujian dengan tiga skenario pembagian data latih dan uji menunjukkan akurasi tertinggi sebesar 96,29% pada rasio 90:10,

sementara skenario lain menunjukkan penurunan akurasi hingga 88,27%. Temuan ini menegaskan bahwa semakin banyak data latih yang digunakan, semakin baik performa klasifikasi yang dihasilkan.

Penelitian oleh Dewi, mengaplikasikan metode *Naïve Bayes Classifier* untuk mengklasifikasikan berita online di Provinsi Riau yang terbagi dalam tiga kategori: demokrasi, kemiskinan, dan ketenagakerjaan (Dewi 2021). Dengan data sebanyak 10.500 berita penelitian ini bertujuan membantu Badan Pusat Statistik Provinsi Riau dalam mengklasifikasikan berita terkait fenomena lokal berdasarkan indeks demokrasi, ketenagakerjaan, dan kemiskinan. Tahapan yang dilakukan meliputi pengumpulan data, *preprocessing* teks, pembobotan kata, dan klasifikasi menggunakan *Naïve Bayes*. Penelitian ini mencapai akurasi tertinggi sebesar 96% pada pembagian data uji 10% dan data latih 90%, menunjukkan efektivitas metode ini dalam klasifikasi berita yang lebih spesifik.

Penelitian oleh Prasetya dan Ferdiansyah, berfokus pada deteksi berita hoaks Covid-19 menggunakan algoritma *Naïve Bayes* (Prasetya and Ferdiansyah 2022). Berdasarkan data Kominfo akhir tahun 2021, terdapat 1773 berita hoaks yang berhasil diklarifikasi, dengan 723 hoaks terkait Covid-19. Peneliti mencoba menguji algoritma *Naïve Bayes* untuk mengklasifikasikan berita hoaks Covid-19. Hasil pengujian menunjukkan bahwa model *Naïve Bayes* dengan *cross-validation* mampu mengklasifikasikan berita hoaks dengan akurasi 86,3%, yang termasuk dalam kategori baik (80-90%). Dari total 300 dataset, hanya 41 data yang salah prediksi atau kurang dari 2%. Ini menunjukkan potensi model ini untuk menjadi

referensi dalam pengembangan model prediksi yang lebih kompleks, seperti model berbasis web.

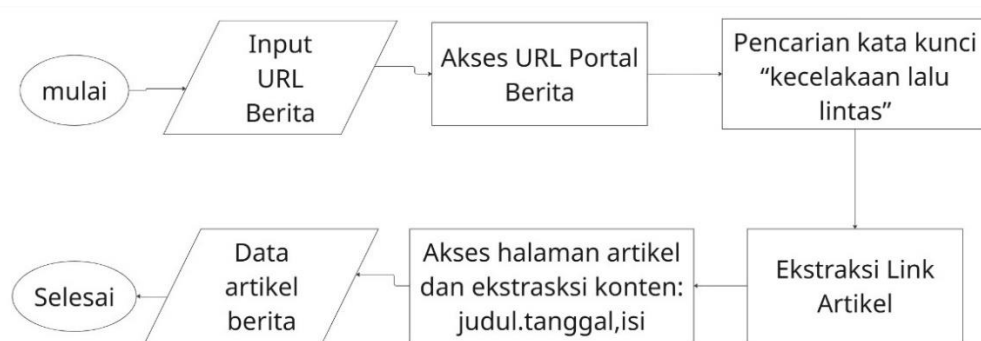
Penelitian oleh Juwita et al., bertujuan mengklasifikasikan berita online menggunakan metode *K-Nearest Neighbor* (KNN) dengan fokus pada tiga kategori utama: demokrassi, kemiskinan, dan ketenagakerjaan (Juwita et al. 2022). Masalah utama yang dihadapi adalah klasifikasi berita yang masih umum, sehingga menyulitkan pembaca yang ingin mencari berita spesifik. Untuk mengatasi hal ini, peneliti mengumpulkan 510 data berita, melakukan pelabelan manual, *preprocessing* teks, pembobotan kata, dan akhirnya mengaplikasikan metode KNN dengan penambahan *cosinus similarity* untuk meningkatkan akurasi. Hasil pengujian menunjukkan bahwa akurasi tertinggi sebesar 87% dicapai dengan nilai $k = 3$ dan distribusi data uji 20% serta data latih 80%. Penelitian ini menyimpulkan bahwa metode KNN efektif dalam klasifikasi berita online yang lebih spesifik dan relevan bagi pengguna.

BAB III

DESAIN DAN IMPLEMENTASI

3.1 Pengumpulan Data

Dalam penelitian ini, data yang digunakan untuk klasifikasi penyebab kecelakaan lalu lintas dikumpulkan dari berbagai sumber berita online, yaitu Detik, CNN Indonesia, dan Kompas dengan periode pengambilan data antara tahun 2020 hingga 2025. Sumber ini dipilih karena jangkauan informasinya yang luas, sehingga dapat memberikan gambaran mengenai insiden kecelakaan lalu lintas yang terjadi di Indonesia. Data yang berhasil dikumpulkan berjumlah 500, yang kemudian diklasifikasikan ke dalam tiga faktor penyebab yakni 63 data untuk faktor jalan, 58 data untuk faktor kendaraan, dan 379 data untuk kelalaian pengemudi. Pengumpulan data dilakukan secara otomatis menggunakan teknik *web scraping* berbasis *Python* dengan memanfaatkan *library Selenium* dan *BeautifulSoup*. *Selenium* digunakan untuk mengotomatisasi interaksi dengan halaman *web* secara dinamis, sementara *BeautifulSoup* digunakan untuk mengekstrak elemen-elemen *HTML* tertentu dari struktur dokumen *web*. Proses *scraping* dilakukan secara *headless*, yaitu tanpa menampilkan tampilan *browser*, untuk meningkatkan efisiensi pemrosesan. Tahapan *Scraping Data*:



Gambar 3.1 *Flowchart Scrapping Data*

1. Akses Halaman Pencarian

Proses dimulai dengan mengakses halaman pencarian pada masing-masing situs berita, menggunakan kata kunci "kecelakaan lalu lintas". Pada situs Kompas, misalnya, halaman pencarian diakses melalui parameter *query*:

<https://search.kompas.com/search/?q=kecelakaan+lalu+lintas>.

Halaman pencarian dari situs lain seperti Detik, CNN Indonesia, dan Tribunnews menggunakan struktur URL yang berbeda, namun prinsip akses dan pencarian tetap sama.

2. Ekstraksi Tautan Artikel

Setelah halaman pencarian termuat, program mengekstrak semua tautan yang mengarah ke halaman artikel berita yang relevan. Setiap situs memiliki struktur *HTML* yang berbeda, sehingga pencarian elemen harus disesuaikan.

Misalnya, pada situs Kompas artikel berada dalam elemen `<div class="gsc-expansionArea">`, sementara pada Tribunnews artikel ditampilkan dalam elemen `<div class=" gsc-webResult gsc-result">` atau variasi lain.

3. Navigasi dan Ekstraksi Detail Artikel.

Setiap tautan yang diperoleh kemudian diakses untuk mengekstrak informasi berikut:

Judul artikel: Biasanya dalam tag `<h1>`.

Tanggal publikasi: Biasanya dalam elemen dengan class seperti *read__Time, date*, atau *article__date*.

Isi artikel: Diambil dari paragraf-paragraf dalam elemen dengan *class* seperti *read__content*, *article__content*, atau *content-detail*, tergantung situs.

Karena struktur *DOM* setiap situs berbeda, maka proses parsing harus diadaptasi untuk masing-masing situs. Sebagai contoh, struktur elemen pada Kompas tidak dapat digunakan langsung untuk Detik atau CNN Indonesia.

4. Penyimpanan Data

Data yang berhasil diambil disimpan ke dalam file **XLSX** menggunakan modul *xlsx Python*. Data disimpan dengan atribut: *Title*, *Date*, *Link*, dan *Content*.

Setelah data terkumpul, dilakukan proses pelabelan terhadap setiap berita berdasarkan kategori penyebab kecelakaan. Pelabelan ini bertujuan untuk mengklasifikasikan artikel ke dalam kelas yang relevan untuk digunakan sebagai *ground truth* pada proses pelatihan model klasifikasi. Kategori pelabelan yang digunakan dalam penelitian ini dibagi menjadi tiga kelas utama, yaitu:

- a. **Kelalaian Pengemudi:** Mencakup perilaku pengemudi yang berisiko menyebabkan kecelakaan, seperti mengemudi dalam kondisi mabuk,

menggunakan ponsel saat berkendara, melanggar rambu lalu lintas, serta kurangnya perhatian atau konsentrasi saat mengemudi.

- b. **Faktor Kendaraan:** Mencakup kondisi teknis kendaraan yang tidak berfungsi dengan baik, seperti rem yang tidak berfungsi, ban yang bocor, atau masalah lainnya yang dapat memengaruhi kinerja kendaraan.
- c. **Faktor Jalan:** Karena keterbatasan data yang tersedia dalam berita, kategori ini menggabungkan dua faktor: kondisi jalan (seperti jalan berlubang, licin, atau minim penerangan) dan kondisi lingkungan (seperti cuaca buruk, keberadaan hewan, atau gangguan lain yang memengaruhi konsentrasi pengemudi).

Pelabelan ini mengacu pada kategori penyebab kecelakaan yang telah dikemukakan dalam literatur dan berdasarkan referensi dari Badan Penelitian dan Pengembangan Perhubungan (Balitbanghub). Proses pelabelan ini juga diperkuat dengan survei kepada responden untuk memperoleh validasi persepsi masyarakat mengenai penyebab kecelakaan yang sering terjadi. Dengan demikian, pelabelan yang dilakukan menjadi lebih objektif.

3.2 Desain Sistem

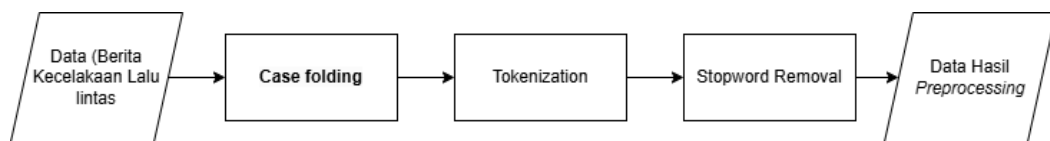
Pada penelitian ini, dapat dilihat pada gambar 3.2 terdapat beberapa proses yang terlibat dalam sistem Klasifikasi penyebab kecelakaan lalu lintas. Proses-proses tersebut meliputi:



Gambar 3.2 Diagram Blok Desain System

3.2.1 Preprocessing

Bagian *preprocessing* bertujuan untuk mengolah data teks mentah menjadi bentuk yang lebih bersih dan terstruktur agar dapat digunakan dalam analisis atau proses lebih lanjut. Dalam implementasi ini, *preprocessing* mencakup langkah-langkah pembersihan teks, tokenisasi, penghapusan kata-kata yang tidak penting (*stopwords*). Berikut gambar 3.3 proses dari *preprocessing*:



Gambar 3.3 Diagram Blok *Preprocessing*

1. Case Folding

Case folding merupakan proses mengubah seluruh huruf dalam teks menjadi huruf kecil (*lowercase*). Proses ini bertujuan untuk mengurangi variasi data yang tidak diperlukan, seperti perbedaan antara huruf besar dan huruf kecil. Dalam analisis teks, huruf besar dan kecil biasanya tidak memengaruhi makna sehingga langkah ini penting untuk meningkatkan konsistensi data.

Tabel 3.1 Contoh *Case Folding*

Teks Awal	"Kecelakaan Terjadi di Jalan Tol."
Hasil Case Folding	"kecelakaan terjadi di jalan tol."

2. Tokenization

Tokenization adalah proses memecah teks menjadi elemen-elemen yang lebih kecil yang disebut token. Dalam konteks ini, token biasanya berupa kata-kata individual. Langkah ini penting karena setiap token akan dianalisis secara terpisah dalam tahapan selanjutnya. Tokenisasi dilakukan menggunakan metode *word tokenization*.

Tabel 3.2 Contoh *Tokenization*

Teks Setelah Case Folding	"kecelakaan terjadi di jalan tol."
Hasil Tokenisasi	['kecelakaan', 'terjadi', 'di', 'jalan', 'tol']

3. *Stopword Removal*

Stopword removal adalah proses menghilangkan kata-kata umum yang sering muncul tetapi tidak memiliki makna signifikan dalam analisis teks. Kata-kata ini, seperti "di", "dan", "yang", umumnya tidak memberikan kontribusi terhadap konteks atau informasi penting dalam teks. *Stopword* yang digunakan dalam penelitian ini merujuk pada daftar *stopword* bahasa Indonesia yang disediakan oleh pustaka dari Sastrawi.

Tabel 3.3 Contoh *Stopword Removal*

Hasil Tokenisasi	['kecelakaan', 'terjadi', 'di', 'jalan', 'tol']
Hasil Setelah <i>Stopword Removal</i>	['kecelakaan', 'terjadi', 'jalan', 'tol']

Setelah data teks diproses, langkah selanjutnya adalah mengubah label kategori menjadi representasi numerik yang dapat dipahami oleh model *machine learning*. Dalam penelitian ini, digunakan teknik *One-Hot Encoding* untuk mengubah label yang bersifat kategorikal menjadi vektor biner. Misalnya, untuk kategori "Faktor Jalan", "Faktor Kendaraan", dan "Kelalaian Pengemudi", *One-Hot Encoding* menghasilkan vektor biner seperti berikut:

Faktor Jalan : [1, 0, 0]

Faktor Kendaraan : [0, 1, 0]

Kelalaian Pengemudi : [0, 0, 1]

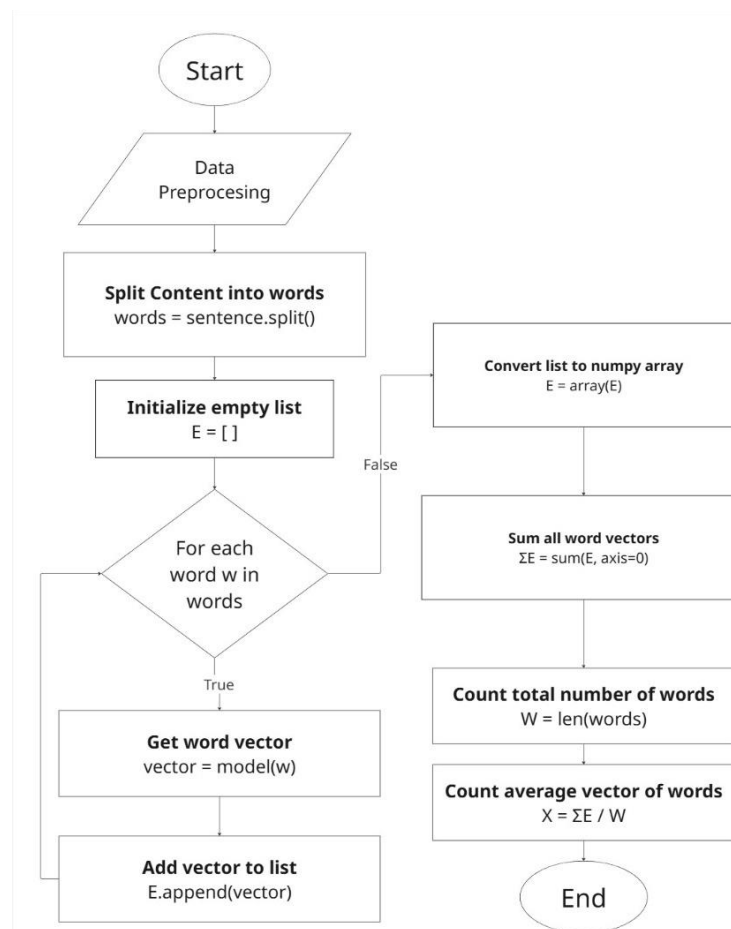
3.2.2 Ekstraksi Fitur

Pada tahap ini, dilakukan proses ekstraksi fitur dari teks berita online yang telah melalui tahap *preprocessing* data. Ekstraksi fitur bertujuan untuk mengubah

teks yang tidak terstruktur menjadi bentuk representasi numerik yang dapat diolah oleh model *machine learning*. Teknik yang digunakan dalam ekstraksi fitur ini adalah *FastText*.

Model ini merepresentasikan kata-kata dalam bentuk vektor numerik berdimensi tetap. Berbeda dengan metode seperti Word2Vec, *FastText* menggunakan subkata (n-gram) untuk menangkap informasi dari setiap kata (Bojanowski et al. 2017). Sehingga mampu menangani kata-kata baru (*out-of-vocabulary*) dengan lebih baik.

Langkah-Langkah Proses ekstraksi fitur menggunakan *FastText* :



Gambar 3.4 Flowchart *FastText*

1. Embedding Kata:

Sebelum dilakukan proses *embedding*, teks dipecah menjadi token-token (kata) dan disimpan dalam variabel W .

Contoh:

Words= $\{w_1, w_2, w_3 \dots w_i\}$, dimana w_1, w_2, w_3 adalah kata-kata seperti "kecelakaan", "terjadi", "jalan".

Lalu setiap kata w_1 memiliki vektor *embedding* yang diperoleh dari model *FastText*

Contoh: misalkan *embedding* untuk masing-masing kata berdimensi 150:

Tabel 3.4 Contoh *Embedding* Kata

Token	Embedding
"kecelakaan"	[0.21, 0.15, 0.30, ..., -0.03]
"terjadi"	[0.25, -0.10, 0.35, ..., 0.02]
"jalan"	[0.23, -0.30, 0.45, ..., 0.05]
"tol"	[-0.27, 0.20, 0.40, ..., -0.04]

2. Representasi Berita:

Setelah semua kata dalam *content* berita memiliki vektor *embedding*, representasi keseluruhan *content* berita dihitung dengan cara merata-ratakan semua vektor kata.

$$X = \frac{\sum E}{W} \quad (3.1)$$

Keterangan:

X : Vektor embedding untuk content berita..

E : Vektor embedding kata w .

W : Jumlah kata dalam content berita.

Contoh representasi berita: misalkan kita memiliki berita dengan token:

['kecelakaan', 'terjadi', 'jalan', 'tol']

Tabel 3.5 Contoh Representasi Berita

<i>Content</i>	Representasi
“kecelakaan terjadi jalan tol”	[0.24, -0.19, 0.37, ..., 0.00]

Dengan demikian, melalui langkah-langkah *FastText* yang terdiri dari subkata, embedding, dan representasi rata-rata kata untuk berita. Sehingga dapat mengubah teks berita yang tidak terstruktur menjadi representasi numerik yang siap diolah oleh model Neural Network untuk klasifikasi atau analisis lebih lanjut.

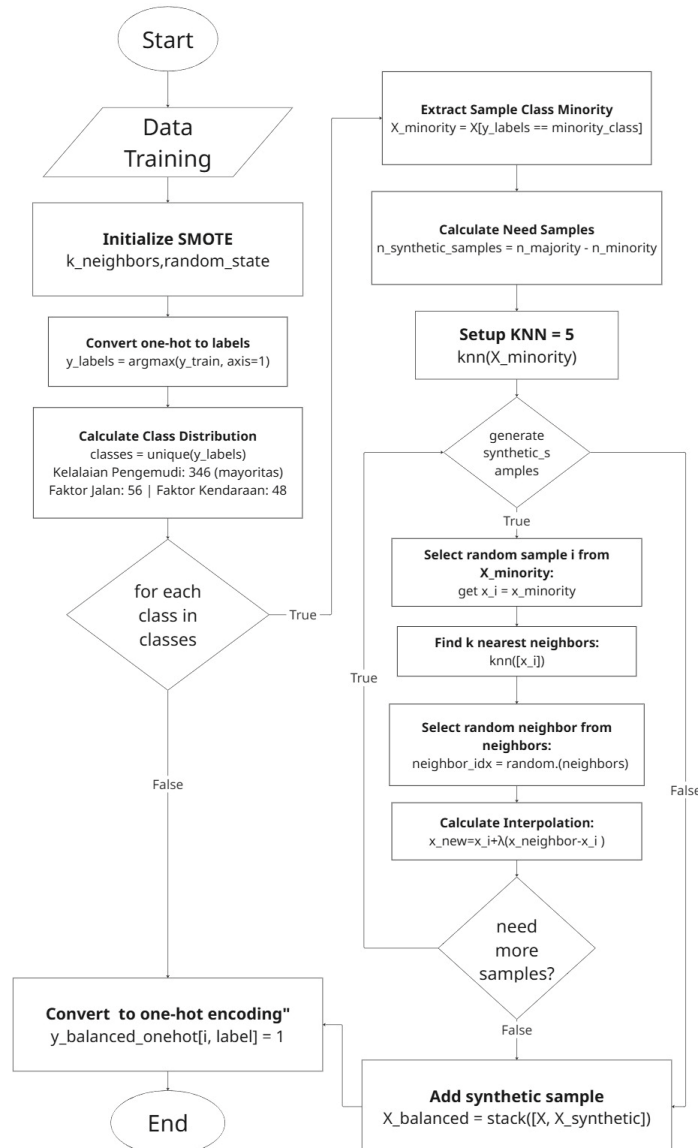
3.2.3 *Synthetic Minority Oversampling Technique (SMOTE)*

Setelah melakukan ekstraksi fitur menggunakan teknik *FastText*, langkah selanjutnya adalah mengatasi masalah ketidakseimbangan kelas dalam dataset yang dapat memengaruhi kinerja model. SMOTE (*Synthetic Minority Over-sampling Technique*) digunakan untuk menangani masalah ini dengan cara menghasilkan sampel sintetis dari kelas minoritas. SMOTE bekerja dengan cara menciptakan sampel sintetis baru berdasarkan interpolasi linear antara sampel kelas minoritas yang ada dengan tetangga terdekatnya dalam ruang fitur. Keunggulan SMOTE dibandingkan teknik oversampling tradisional seperti random oversampling adalah kemampuannya untuk menciptakan sampel yang bervariasi daripada hanya menduplikasi data yang sudah ada. Hal ini membantu mencegah overfitting dan meningkatkan generalisasi model (Chawla et al. 2002).

Dalam konteks klasifikasi teks, penerapan SMOTE memerlukan perhatian khusus karena data teks harus terlebih dahulu dikonversi ke dalam representasi numerik melalui teknik seperti *FastText*. Penelitian menunjukkan bahwa SMOTE tetap efektif dalam meningkatkan performa klasifikasi teks, terutama ketika dikombinasikan dengan representasi vektor *FastText* atau *Word2vec* (Cahyana et

al. 2025) . Setelah melakukan *split* data, langkah selanjutnya adalah mengatasi masalah ketidakseimbangan kelas dalam data *training* yang dapat memengaruhi kinerja model. Data *training* menunjukkan ketidakseimbangan yang signifikan dimana kelas "Kelalaian Pengemudi" mendominasi dengan 346 sampel, sementara "Faktor Jalan" hanya memiliki 56 sampel dan "Faktor Kendaraan" memiliki 48 sampel. Untuk mengatasi masalah ini, diterapkan teknik *Synthetic Minority Over-sampling Technique* (SMOTE) yang diimplementasikan secara manual menggunakan *Python*.

SMOTE bekerja dengan cara menghasilkan sampel sintetis dari kelas minoritas melalui interpolasi linear antara sampel kelas minoritas yang ada dengan tetangga terdekatnya dalam ruang fitur. Implementasi manual SMOTE dalam penelitian ini menggunakan algoritma *K-Nearest Neighbors* (KNN) dari *library scikit-learn* untuk menemukan tetangga terdekat, serta *library NumPy* untuk operasi matematika. Proses SMOTE dilakukan dengan parameter *k_neighbors=5* yang menentukan jumlah tetangga terdekat untuk interpolasi.



Gambar 3.5 Flowchart SMOTE

1. Konversi Format Label

Proses dimulai dengan mengonversi format *one-hot encoded* menjadi label tunggal menggunakan fungsi *argmax()*. Hal ini dilakukan karena algoritma SMOTE membutuhkan label dalam format kategorikal tunggal (0, 1, atau 2) untuk memproses setiap kelas secara terpisah. Konversi ini mengubah *array* berukuran (450, 3) menjadi *array* satu dimensi.

2. Analisis Distribusi Kelas

Setelah konversi label, dilakukan penghitungan distribusi kelas untuk mengidentifikasi kelas mayoritas dan minoritas. Hasil analisis menunjukkan:

Kelas 0 (Faktor Jalan): 56 sampel - kelas minoritas

Kelas 1 (Faktor Kendaraan): 48 sampel - kelas minoritas

Kelas 2 (Kelalaian Pengemudi): 346 sampel - kelas mayoritas

Kelas mayoritas (346 sampel) dijadikan target untuk menyeimbangkan semua kelas.

3. Inisialisasi SMOTE

Implementasi SMOTE menggunakan kelas Manual SMOTE yang diinisialisasi dengan parameter:

k_neighbors=5: Menentukan 5 tetangga terdekat untuk proses interpolasi

random_state=42: Memastikan hasil dapat direproduksi

4. Proses *Oversampling* Kelas Minoritas

Untuk setiap kelas minoritas, dilakukan tahapan berikut:

Ekstraksi Sampel Minoritas program mengidentifikasi sampel yang termasuk kelas minoritas menggunakan *boolean indexing* dengan kondisi *y_labels == minority_class*, kemudian mengekstrak vektor fitur *FastText* dari sampel-sampel tersebut dengan *X_minority = X[y_labels == minority_class]*.

Pembentukan Model KNN dibentuk model *K-Nearest Neighbors* menggunakan *NearestNeighbors* dari *scikit-learn*. Model ini dilatih pada

sampel-sampel kelas minoritas untuk menemukan tetangga terdekat dalam ruang vektor 150 dimensi.

Generate Sampel Sintetis proses pembuatan sampel sintetis dilakukan melalui:

- a) Pemilihan sampel acak dari kelas minoritas
- b) Pencarian k tetangga terdekat menggunakan KNN
- c) Pemilihan satu tetangga secara acak dari k tetangga (tidak termasuk sampel itu sendiri)
- d) Interpolasi linear:

$$x_{new} = x_i + \lambda(x_{neighbor} - x_i) \quad (3.2)$$

Keterangan:

x_{new} : sampel sintetis yang dihasilkan..
 x_i : titik data yang dipilih dari kelas minoritas
 $x_{neighbor}$: salah satu tetangga terdekat dari x_i
 λ : nilai acak antara 0 dan 1, yang menentukan seberapa jauh titik sintetis berada di antara x_i dan x_k .

5. Penggabungan Data

Sampel sintetis yang dihasilkan digabungkan dengan data asli menggunakan `np.vstack()` untuk fitur dan `np.concatenate()` untuk label. Proses ini menghasilkan:

Faktor Jalan: 56 → 346 sampel (290 sampel sintetis)

Faktor Kendaraan: 48 → 346 sampel (298 sampel sintetis)

Kelalaian Pengemudi: tetap 346 sampel

6. Konversi Kembali ke One-hot

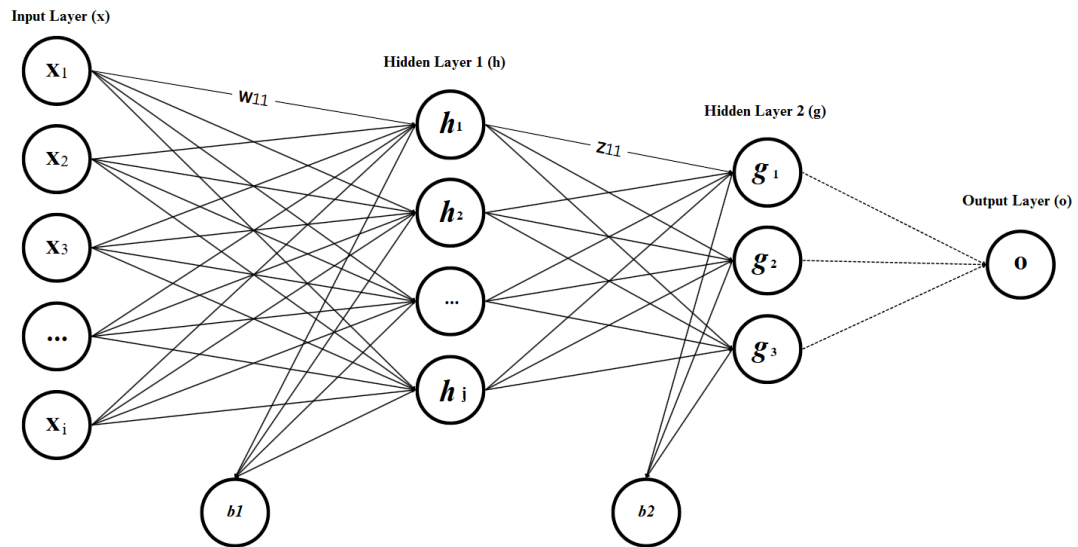
Tahap akhir adalah mengkonversi label tunggal kembali ke format *one-hot encoded* untuk kompatibilitas dengan model klasifikasi. *Array* label

berukuran (1038,) dikonversi menjadi *array* (1038, 3) dimana setiap baris hanya memiliki satu nilai 1 yang menunjukkan kelas sampel tersebut.

Hasil akhir dari proses SMOTE adalah data *training* yang seimbang dengan total 1038 sampel (346×3 kelas), meningkat dari data *training* asli yang hanya memiliki 450 sampel. Keseimbangan ini diharapkan dapat meningkatkan performa model klasifikasi dalam mengidentifikasi semua kategori penyebab kecelakaan lalu lintas secara adil, tidak hanya terfokus pada kelas mayoritas.

3.2.4 Artificial Neural Network

Setelah melalui tahap ekstraksi fitur menggunakan *FastText*, data yang telah diproses dimasukkan ke dalam *Artificial Neural Network* (ANN) untuk klasifikasi. *Artificial Neural Network* adalah model *machine learning* yang terinspirasi oleh cara kerja otak manusia, di mana data melalui tahapan pemrosesan internal menyerupai aktivasi neuron untuk menghasilkan *output*. ANN dipilih karena menangani hubungan non-linear antara fitur teks dan kelas penyebab kecelakaan serta fleksibilitas dalam menyesuaikan kompleksitas model melalui penambahan lapisan tersembunyi (*Hidden Layers*) (Goodfellow et al. 2016). Keunggulan ANN juga terletak pada kemampuannya untuk mempelajari representasi data dalam bentuk yang lebih abstrak, yang sangat penting untuk memahami konteks yang lebih dalam data teks berita tentang kecelakaan lalu lintas. Arsitektur dari *Neural Network* yang digunakan dalam penelitian ini dapat dilihat pada gambar 3.6.

Gambar 3.6 Arsitektur *Neural Network*

Arsitektur *Artificial Neural Network* yang digunakan dalam penelitian ini mencakup empat lapisan utama. Pembagian lapisan ini dirancang untuk memastikan efisiensi pemrosesan data dari *input* hingga *output*, sehingga setiap lapisan memiliki peran khusus dalam ekstraksi dan pengolahan fitur. Pemilihan jumlah layer dan *Node* pada penelitian ini didasarkan pada beberapa kriteria:

1. *Input Layer*: Memiliki sejumlah *node* yang disesuaikan dengan dimensi *output* dari ekstraksi fitur (*FastText*).
2. *Hidden Layer 1*: Jumlah *node* ditentukan berdasarkan pada skenario pengujian.
3. *Hidden Layer 2*: Terdiri dari 3 *node*, sesuai jumlah kelas *output* yang diprediksi.
4. *Output Layer*: Menggunakan 1 *node* untuk menghasilkan salah satu prediksi kelas.

Proses alur data dari *input* hingga *output* dikenal sebagai *feedforward*, di mana setiap neuron pada layer *input* menerima sinyal mengalikannya dengan bobot, menambahkan bias, dan meneruskan hasilnya ke layer berikutnya hingga mencapai layer *output*. *Feedforward* sangat penting dalam ANN karena memungkinkan jaringan untuk mempelajari pola data secara berurutan. Setiap lapisan bertanggung jawab untuk mengekstraksi fitur yang lebih kompleks dan abstrak, meningkatkan kemampuan jaringan dalam mengenali pola-pola kompleks seperti yang terdapat pada berita kecelakaan lalu lintas. Beberapa studi menunjukkan bahwa *feedforward* memiliki keunggulan dalam kecepatan konvergensi dan efisiensi perhitungan, khususnya pada model yang melibatkan banyak data dan parameter, serta dapat mempercepat pelatihan model dalam skala besar (Gelvez-Almeida et al. 2024). Selama proses *feedforward*, dilakukan langkah-langkah berikut:

1. Perhitungan pada *Hidden Layer* 1

Fungsi aktivasi yang digunakan pada *Hidden Layer* 1 adalah ReLU

(*Rectified Linear Unit*), yang dirumuskan sebagai:

$$h_j = \max\left(0, \left(b_1 + \sum_i w_{ij} \cdot x_i\right)\right) \quad (3.3)$$

Keterangan:

b_1 : Bias pada *Hidden Layer* 1.

w_{ij} : Bobot antara neuron input ke- i dan neuron ke- j di *Hidden Layer* 1.

x_i : Nilai input pada neuron ke- i .

jika > 0 , maka $h_j = h$

Jika < 0 , maka $h_j = 0$

Nilai ini memastikan bahwa hanya nilai positif yang diteruskan ke lapisan berikutnya membantu jaringan untuk belajar representasi non-linear. Penggunaan *ReLU* dalam *Hidden Layer* memiliki keunggulan dibandingkan *Sigmoid* dalam hal

kecepatan konvergensi dan akurasi. Penelitian menunjukkan bahwa *ReLU* lebih stabil dalam proses pelatihan dan dapat meningkatkan *generalization* pada model *Artificial Neural Network* (ANN), terutama pada jaringan dengan jumlah unit tersembunyi yang besar. *Sigmoid* sering kali mengalami transisi yang tidak kontinu dan kesulitan dengan set pelatihan kecil, yang menyebabkan keterlambatan dalam proses pembelajaran (Oostwal, Straat, and Biehl 2021).

2. Perhitungan pada *Hidden Layer 2*

Aktivasi pada *Hidden Layer 2* dilakukan menggunakan fungsi *Softmax*, yang dirumuskan sebagai:

$$g_k = \frac{e^{(b_2 + \sum_j z_{jk} \cdot h_j)}}{\sum_{k=1}^K e^{(b_2 + \sum_j z_{jk} \cdot h_j)}} \quad (3.4)$$

$$o = \operatorname{argmax}(g_k) \quad (3.5)$$

Keterangan :

g_k : Probabilitas hasil kelas ke- k .

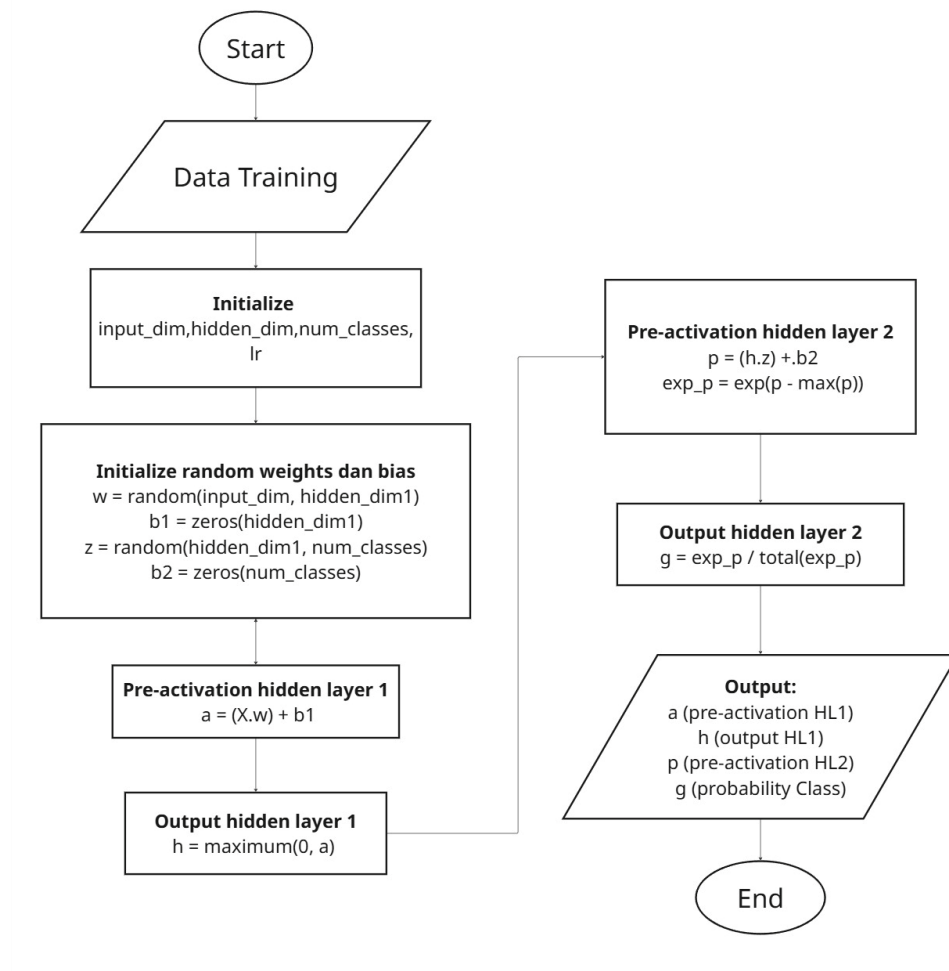
b_2 : Bias untuk neuron ke- k di *Hidden Layer 2*.

z_{jk} : Bobot antara neuron ke- j di *Hidden Layer 1* dan neuron ke- k di *Hidden Layer 2*.

$\sum_{k=1}^K e^{p_k}$: Normalisasi nilai eksponensial untuk semua kelas.

Fungsi *Softmax* digunakan pada *Output Layer* untuk mengonversi hasil dari *Hidden Layer 2* menjadi probabilitas kelas. Fungsi ini mengonversi nilai *output* menjadi distribusi probabilitas, yang menjamin bahwa total probabilitas untuk semua kelas adalah 1. Penelitian menunjukkan bahwa *Softmax* efektif digunakan pada jaringan yang lebih besar dan kompleks, membantu jaringan untuk meningkatkan akurasi dalam klasifikasi dan mengurangi kesalahan dibandingkan dengan fungsi aktivasi lainnya seperti *Sigmoid*. *Softmax* memberikan keuntungan dalam hal komputasi dan efisiensi pada *Artificial Neural Networks* (ANN) yang

lebih besar, yang digunakan untuk mengklasifikasikan data dengan banyak kelas, serta meningkatkan *generalization* jaringan (Almurieb and Bhaya 2020).



Gambar 3.7 Flowchart Proses Feedforward

Pada Gambar 3.7, ditampilkan *flowchart* proses *feedforward* yang menunjukkan implementasi dari rumus-rumus *feedforward* yang telah dijelaskan sebelumnya untuk di implementasikan ke dalam *code*. dimulai dengan inisialisasi boot yang meliputi *input_dim* (dimensi fitur *FastText*), *hidden_dim* (jumlah *node* pada *Hidden Layer 1*), *num_classes* (jumlah kelas *output*), dan *learning rate* (*lr*). Dalam implementasi *input_dim* setara dengan dimensi *xi*, *hidden_dim* setara dengan jumlah dari *node* pada *hidden layer 1*, *num_classes* bernilai 3 sesuai dengan

jumlah kelas klasifikasi. Selanjutnya dilakukan inisialisasi *random* bobot dan bias dimana w dalam implementasi merepresentasikan bobot w_{ij} antara *Input Layer* dan *Hidden Layer 1*, z merepresentasikan bobot z_{jk} antara *Hidden Layer 1* dan *Output Layer*. Sementara b_1 dan b_2 masing-masing merepresentasikan bias b_1 dan b_2 pada *Hidden Layer 1* dan *Output Layer*.

Proses *feedforward* yang digambarkan dalam *flowchart* mengimplementasikan rumus (3.3) dan (3.4) secara praktis. *Pre-activation* pada *Hidden Layer 1* dihitung menggunakan operasi $a = (X \cdot w) + b_1$, dimana $X \cdot w$ merupakan implementasi *matrix multiplication* yang setara dengan operasi penjumlahan $\sum w_{ij} \cdot x_i$ dalam rumus (3.3). Aktivasi ReLU kemudian diterapkan dengan $h = \text{maximum}(0, a)$ yang mengimplementasikan fungsi $\max(0, \dots)$ dari rumus (3.3). Pada *Hidden Layer 2*, *pre-activation* dihitung dengan $p = h \cdot z + b_2$, dimana operasi *matrix multiplication* ini setara dengan $\sum_j z_{jk} \cdot h_j$ dalam rumus (3.4). Aktivasi *Softmax* diimplementasikan melalui perhitungan $\exp_p = \exp(p - \max(p, \text{axis}=1))$ dan $g = \exp_p / \text{sum}(\exp_p, \text{axis}=1,)$, dengan penambahan *numerical stability* untuk mencegah *overflow* yang dapat terjadi dalam komputasi praktis.

3.3 Implementasi Sistem

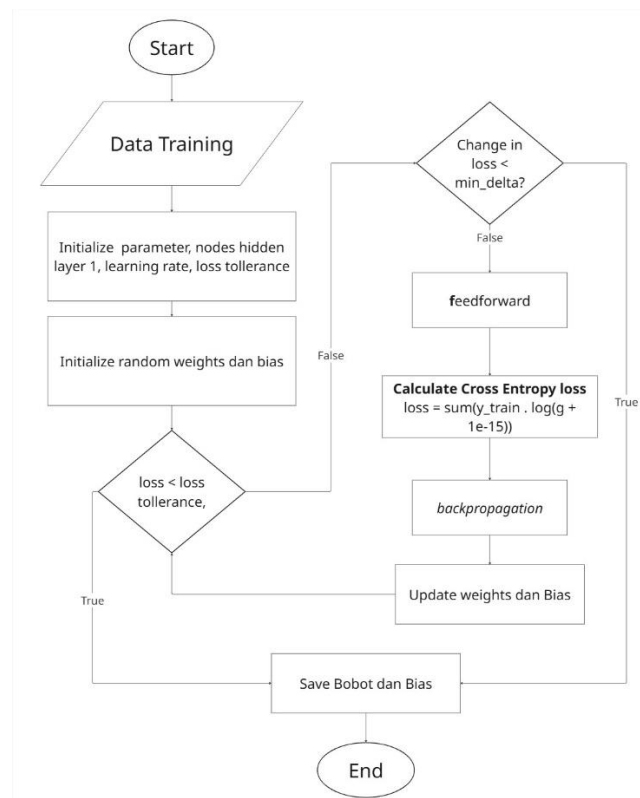
Agar model dapat mengklasifikasi faktor penyebab kecelakaan lalu lintas, langkah pertama yang harus dilakukan adalah melaksanakan proses pelatihan (*training*). Proses pelatihan ini akan menghasilkan nilai kesalahan (*error*) yang selanjutnya digunakan untuk memperbarui dan mengoptimalkan nilai bobot (*weight*) dan bias yang ada dalam model jaringan saraf tiruan (*neural network*).

Setelah proses pelatihan selesai, model akan menjalani tahap pengujian (*testing*) untuk mengevaluasi dan mengukur kinerja model tersebut.

3.3.1 Proses *Training*

Proses *training* merupakan fase pembelajaran dimana model *neural network* melakukan optimasi parameter untuk meminimalkan kesalahan klasifikasi. Selama proses ini, model akan melakukan evaluasi secara berkala menggunakan fungsi *loss* untuk mengukur seberapa baik performa model dalam memprediksi label target.

Dapat dilihat pada gambar 3.8



Gambar 3.8 Flowchart Proses *Training*

Pada gambar 3.8 ditampilkan *flowchart* proses *training* yang menggambarkan alur pembelajaran model *neural network* dari awal *input* hingga mendapatkan bobot

yang optimal. Proses *training* dimulai dengan inisialisasi parameter meliputi *nodes hidden layer*, *learning rate*, *loss tolerance*. Selanjutnya, model memasuki *loop training* yang akan terus berulang sampai salah satu kondisi berhenti terpenuhi. Dalam setiap iterasi, dilakukan proses *feedforward* untuk menghasilkan probabilitas, kemudian dihitung *Cross-Entropy Loss* menggunakan rumus (3.6) yang dalam *code* diimplementasikan sebagai variabel *loss*. Kondisi berhenti pertama akan dicek jika nilai *loss* sudah lebih kecil dari *tolerance*, maka *training* dihentikan karena model sudah mencapai tingkat akurasi yang diinginkan. Namun jika *loss* masih lebih besar dari *tolerance*, proses dilanjutkan ke tahap *backpropagation* untuk menghitung gradien dan *update* bobot serta bias. Selama proses *training* akan memeriksa apakah terjadi perbaikan signifikan dengan membandingkan perubahan *loss* terhadap *min_delta*. Jika perubahan *loss* lebih kecil dari *min_delta*, ini mengindikasikan bahwa model tidak mengalami perbaikan. Sebaliknya, jika perubahan *loss* masih cukup besar (melebihi *min_delta*). Kondisi berhenti kedua terjadi ketika mencapai nilai *patience*, yang menandakan model sudah tidak dapat diperbaiki lagi meskipun *loss* belum mencapai *tolerance*. Dalam penelitian ini, digunakan fungsi *Cross-Entropy Loss* untuk mengukur kesalahan antara prediksi model (g_k) dengan label target (y_k). Fungsi *loss* dirumuskan sebagai:

$$l = - \sum^K y_k \log(g_k) \quad (3.6)$$

Keterangan:

l : Nilai *loss* (kesalahan).

y_k : Label target untuk kelas ke- k .

g_k : Probabilitas prediksi untuk kelas ke- k .

K : Jumlah kelas.

Proses pelatihan dalam *Artificial Neural Network* (ANN) dilakukan menggunakan algoritma *backpropagation*. Algoritma ini menghitung Gradien dari fungsi *loss* terhadap setiap bobot dalam jaringan lalu memperbarui bobot secara iteratif untuk meminimalkan nilai *loss*. Setelah menghitung nilai *loss*, Gradien *loss* digunakan untuk memperbarui bobot (w, z) dan bias (b) jaringan saraf agar model dapat belajar secara optimal. *Backpropagation* memungkinkan model untuk belajar secara efisien dengan mengoptimalkan bobot dan bias meningkatkan kemampuan model untuk mengenali pola dalam data yang diberikan.

Selain itu, *backpropagation* sangat berguna dalam mengatasi masalah konvergensi pada jaringan saraf dengan banyak lapisan. Algoritma ini memungkinkan jaringan untuk mengoptimalkan bobot dan bias secara lebih efisien mempercepat proses pelatihan, serta meningkatkan kemampuan jaringan untuk menangani data besar dan kompleks. Tanpa *backpropagation*, jaringan saraf tidak akan dapat belajar dengan cara yang efisien dan kesulitan untuk mencapai solusi optimal. *Backpropagation* juga sangat efektif dalam jaringan yang memiliki banyak parameter, mempercepat proses pelatihan dan meningkatkan kinerja model dalam tugas klasifikasi (Siddique and Tokhi 2001).

1. Menghitung Gradien *Loss* pada *Output Layer*

Gradien *loss* pada *Output Layer* dihitung dengan menurunkan fungsi Cross-Entropy *Loss* terhadap output *Softmax* (g_k) :

$$\frac{\partial L}{\partial g_k} = g_k - y_k \quad (3.7)$$

Keterangan:

$\frac{\partial L}{\partial g_k}$: Gradien output (g_k).

g_k : Probabilitas prediksi untuk kelas ke- k .

y_k : Label target untuk kelas ke- k .

2. Menghitung Gradien *Loss*, Pembaruan Bobot, dan Bias pada *Hidden Layer 2*

a. Gradien pada *Hidden Layer 2*

Gradien *loss* terhadap nilai pada hidden layer 2 dihitung menggunakan chain rule sebagai berikut:

$$\frac{\partial L}{\partial p_k} = \frac{\partial L}{\partial g_k} \quad (3.8)$$

Keterangan :

$\frac{\partial L}{\partial p_k}$: Gradien *loss* pada nilai logit.

b. Gradien *loss* dan pembaruan bobot (z_{jk})

Gradien *loss* pada bobot antara neuron ke- j di *Hidden Layer 1* dan neuron k di *Hidden Layer 2* dihitung sebagai :

$$\frac{\partial L}{\partial z_{jk}} = \frac{\partial L}{\partial p_k} \cdot h_j \quad (3.9)$$

Pembaruan bobot dilakukan dengan rumus :

$$z_{jk}^{(t+1)} = z_{jk}^{(t)} - \eta \cdot \frac{\partial L}{\partial z_{jk}} \quad (3.10)$$

Keterangan :

$\frac{\partial L}{\partial z_{jk}}$: Gradien *loss* bobot (z_{jk}).

h_j : *Output* dari fungsi aktivasi ReLU pada *neuron* ke- j di *Hidden Layer 1*.

z_{jk} : Bobot antara *neuron* ke- j di *Hidden Layer 1* dan *neuron* ke- k di *Hidden Layer 2*.

η : *Learning rate*.

t : iterasi saat ini.

c. Gradien *Loss* dan Pembaruan Bias (b_2)

Gradien *loss* pada bias di *Hidden Layer 2* dihitung sebagai:

$$\frac{\partial L}{\partial b_2} = \frac{\partial L}{\partial p_k} \quad (3.11)$$

Pembaruan bias dilakukan dengan rumus :

$$b_2^{(t+1)} = b_2^{(t)} - \eta \cdot \frac{\partial L}{\partial b_2} \quad (3.12)$$

Keterangan :

$\frac{\partial L}{\partial b_2}$: Gradien *loss* bias (b_2).

b_2 : Bias pada *neuron* ke-k di *Hidden Layer 2*.

3. Menghitung Gradien *Loss*, Pembaruan Bobot, dan Bias pada *Hidden Layer 1*

a. Gradien pada *Hidden Layer 1* (h_j)

Jika $h'_j > 0$

Gradien *loss* terhadap neuron ke-j di *Hidden Layer 1* dihitung

menggunakan:

$$\frac{\partial L}{\partial h_j} = \sum^K \left(\frac{\partial L}{\partial p_k} \cdot z_{jk} \right) \quad (3.13)$$

Keterangan :

$\frac{\partial L}{\partial h_j}$: Gradien *loss* pada *neuron* ke-j *Hidden Layer 1*.

z_{jk} : Bobot lama antara *neuron* ke-j di *Hidden Layer 1* dan *neuron* ke-k di *Hidden Layer 2*.

Jika $h'_j < 0$

turunan fungsi aktivasi ReLU h'_j akan bernilai 0, sehingga gradien *loss*

tidak akan berpengaruh dan tidak akan diperbarui pada neuron tersebut

$$\frac{\partial L}{\partial h_j} = 0 \quad (3.14)$$

b. Gradien dan pembaruan bobot (w_{ij})

Gradien *loss* pada bobot antara neuron ke- i di *Input Layer* dan neuron ke- j di *Hidden Layer* 1 dihitung sebagai :

$$\frac{\partial L}{\partial w_{ij}} = \frac{\partial L}{\partial h_j} \cdot x_i \quad (3.15)$$

Pembaruan bobot dilakukan dengan rumus :

$$w_{ij}^{(t+1)} = w_{ij}^{(t)} - \eta \cdot \frac{\partial L}{\partial w_{ij}} \quad (3.16)$$

Keterangan :

$\frac{\partial L}{\partial w_{ij}}$: Gradien bobot (w_{ij}).

w_{ij} : bobot antara neuron ke- i di *Input Layer* dan neuron ke- j di *Hidden Layer* .

x_i : Nilai input dari data.

c. Gradien dan pembaruan bias (b_1)

Gradien *loss* pada bias di *Hidden Layer* 1 dihitung sebagai :

$$\frac{\partial L}{\partial b_1} = \frac{\partial L}{\partial h_j} \quad (3.17)$$

Pembaruan bias dilakukan dengan rumus :

$$b_1^{(t+1)} = b_1^{(t)} - \eta \cdot \frac{\partial L}{\partial b_1} \quad (3.18)$$

Keterangan :

$\frac{\partial L}{\partial b_1}$: Gradien bias (b_1).

b_1 : Bias pada neuron ke- j di *Hidden Layer* 1.

Berdasarkan rumus dari algoritma *Backpropagation* yang telah dijelaskan sebelumnya, dan mengacu pada Gambar 3.8, berikut disajikan *flowchart* yang lebih

rinci mengenai proses *Backpropagation* yang akan diimplementasikan ke dalam kode program, seperti ditunjukkan pada Gambar 3.9.



Gambar 3.9 Flowchart Proses *Backpropagation*

Gambar 3.9 menampilkan *flowchart* detail proses *backpropagation* yang merupakan implementasi dari rumus-rumus gradien (3.7-3.18) ke dalam *code*. Proses dimulai dengan menerima *output* dari *feedforward* yaitu nilai a , h , p , dan g ,

kemudian melakukan propagasi mundur untuk menghitung gradien setiap parameter. Tahap pertama menghitung gradien *output layer* merupakan implementasi dari rumus (3.7). Di sini terdapat perbedaan notasi dimana y_k dalam rumus diimplementasikan sebagai y_true dalam *code* untuk menunjukkan label target yang sebenarnya. Karena turunan *softmax* memiliki sifat khusus, gradien *pre-activation Hidden Layer 2* sama dengan gradien *output*, sehingga sesuai rumus (3.8). Selanjutnya gradien *weights Hidden Layer 2* dihitung menggunakan dimana dalam rumus (3.9). Perbedaan terlihat pada z_{jk} dalam rumus yang diimplementasikan sebagai z dalam *code* dan penggunaan *transpose* (h.T) untuk mencocokkan dimensi matriks dalam *batch processing*. *Update weights* z dilakukan dimana η (eta) dalam rumus (3.10) diimplementasikan sebagai *self.lr* (*learning rate*) dalam *code*. Untuk gradien bias *Hidden Layer 2*, *code* mengimplementasikan rumus (3.11). Notasi b_2 dalam rumus konsisten dengan b_2 dalam *code*, namun operasi *sum* dengan *axis=0* diperlukan untuk agregasi *batch* yang tidak eksplisit ditunjukkan dalam rumus matematika. Proses berlanjut ke *Hidden Layer 1* dengan menghitung turunan ReLU menggunakan $relu_derivative = (a > 0)$. Ini merupakan implementasi praktis dari kondisi dalam rumus (3.13) dan (3.14) dimana gradien bergantung pada nilai aktivasi. Gradien h dihitung dengan mengimplementasikan rumus (3.13). Untuk gradien *weights input layer* mengimplementasikan rumus (3.15). Perbedaan notasi terlihat pada w_{ij} dalam rumus yang diimplementasikan sebagai w dalam *code*, dan x_i yang dalam *code* adalah X (huruf kapital menunjukkan matriks *batch*). Begitu pula untuk bias, b_1 dalam rumus konsisten dengan *self.b1* dalam *code*.

BAB IV

UJI COBA DAN PEMBAHASAN

4.1 Skenario Uji Coba

Tahap ini melakukan skenario pengujian untuk mengevaluasi performa model dalam berbagai kombinasi parameter utama. Tujuannya adalah menemukan konfigurasi optimal untuk klasifikasi berita kecelakaan lalu lintas, dengan fokus pada konfigurasi yang menghasilkan akurasi tertinggi dan kesalahan minimal. Evaluasi dilakukan dengan memantau metrik seperti akurasi dan *loss* konvergensi pada berbagai kombinasi parameter. Parameter utama yang diuji meliputi:

1. *Learning rate*: Kecepatan pembaruan bobot pada setiap iterasi pelatihan. *Learning rate* yang terlalu besar dapat menyebabkan model mengalami *overshooting*, sedangkan *Learning rate* yang terlalu kecil dapat memperlambat proses konvergensi (Goodfellow et al. 2016).
2. Toleransi Kesalahan (*Error Tolerance*): Ambang batas kesalahan yang menentukan kapan pelatihan dihentikan.
3. Jumlah *Node* di *Hidden Layer* 1: Memengaruhi kompleksitas model. Jumlah *Node* yang lebih banyak dapat meningkatkan kapasitas representasi model, tetapi juga meningkatkan risiko *overfitting* jika jumlah data terbatas (Bengio, Courville, and Vincent 2012).

Variasi parameter yang digunakan dalam pengujian ini dirangkum dalam Tabel 4.1 berikut :

Tabel 4.1 Skenario Uji Coba

No.	Jumlah <i>Node</i> (<i>Hidden Layer</i> 1)	<i>Learning rate</i>	Toleransi Kesalahan
1	21	0.001	0.01
2	21	0.0001	0.01
3	301	0.001	0.01
4	301	0.0001	0.01

Tabel 4.1 menyajikan empat skenario uji coba dengan jumlah data yang relatif kecil, yaitu 500. Variasi parameter pada tabel dirancang untuk mengevaluasi performa model dengan kombinasi dimensi *FastText*, *Learning rate*, dan toleransi kesalahan yang sesuai untuk dataset ini. Berdasarkan hasil penelitian (Držik and Kapusta 2024), dimensi *FastText* yang digunakan dibatasi pada nilai 150 karena dimensi yang lebih kecil dari 50 dianggap tidak memberikan hasil optimal sedangkan dimensi yang lebih besar dari 150 tidak menunjukkan peningkatan signifikan dalam kualitas hasil. Pemilihan dimensi 150 pada skenario pertama dan kedua dilakukan untuk menghasilkan representasi yang lebih sederhana dan menghindari kompleksitas yang berlebihan sekaligus meminimalkan risiko *overfitting* pada jumlah data yang terbatas.

Jumlah *Node* pada *Hidden Layer* pertama dipilih dengan mempertimbangkan kapasitas model untuk menangkap pola kompleks, tanpa mengorbankan kecepatan komputasi dan risiko *overfitting*. Dua variasi jumlah *Node* yang digunakan adalah 21 dan 301, dengan tujuan untuk mengevaluasi dampaknya terhadap performa model. Jumlah *Node* sebesar 21 diperoleh menggunakan rumus yang disarankan dalam artikel di LinkedIn, Jumlah neuron = $\sqrt{(\text{jumlah neuron input} \cdot \text{jumlah neuron output})}$ (Sachdev 2020). Dengan memasukkan nilai *input* sebanyak 150 dan *output* sebanyak 3, diperoleh 21 neuron. Dan berdasarkan penelitian oleh Syaharuddin

dkk., jumlah *Node* optimal pada *Hidden Layer* dapat dihitung menggunakan rumus Hecht-Nelson yang cenderung menghasilkan jumlah neuron yang lebih besar untuk meningkatkan kapasitas model dalam menangkap pola data yang lebih kompleks (Syaharuddin dkk. 2022). Untuk kasus ini, dengan 150 neuron *input* dan 3 neuron *output* rumus Hecht-Nelson menghasilkan jumlah neuron sekitar 301.

Dalam penelitian ini, nilai *Learning rate* divariasikan menjadi dua tingkatan, yaitu 0.001 dan 0.0001. Nilai *Learning rate* 0.001 dipilih karena berdasarkan penelitian Widhiyasana et al., nilai 0.001 lebih baik daripada 0.0001 dengan akurasi mencapai 80.98% (Widhiyasana et al. 2021). Sedangkan nilai *Learning rate* 0.0001 dipilih karena penelitian Jepkoech et al., menunjukkan bahwa *Learning rate* sebesar 0.0001 memberikan hasil optimal dengan akurasi pengujian mencapai 99.4% dan akurasi pelatihan 100% (Jepkoech et al. 2021). Hal ini mengindikasikan bahwa toleransi kesalahan yang lebih ketat meskipun membutuhkan waktu pelatihan tambahan dapat memastikan bahwa model mencapai tingkat akurasi yang lebih tinggi dan *loss* yang lebih rendah sesuai dengan tujuan penelitian ini.

Toleransi kesalahan digunakan sebagai ambang batas untuk menghentikan pelatihan model sebelum terjadinya *overfitting*. Nilai toleransi seperti 0,001 diterapkan untuk pelatihan yang moderat, di mana pelatihan dihentikan saat nilai *loss* sudah kecil. Pendekatan ini sejalan dengan konsep *early stopping* sebagaimana diuraikan dalam penelitian oleh Vilares Ferro et al., di mana teknik ini menggunakan indikator online untuk mendeteksi *overfitting* dengan mengandalkan analisis korelasi pada indikator tersebut (Vilares Ferro et al. 2023). Hal ini

membantu menghindari *overfitting* secara lebih andal dan meningkatkan efisiensi pelatihan.

Setelah model dilatih, evaluasi kinerja model menjadi langkah penting untuk menilai seberapa baik model dapat mengklasifikasikan data yang belum pernah dilihat sebelumnya. Salah satu metode yang sering digunakan untuk mengevaluasi model klasifikasi adalah *Confusion Matrix* dengan memberikan gambaran rinci tentang bagaimana model melakukan klasifikasi berdasarkan kelas yang sebenarnya (*true classes*) dan kelas yang diprediksi (*predicted classes*) (Sathyanarayanan 2024). *Confusion Matrix* menghasilkan empat komponen utama dalam evaluasinya. *True Positive* (TP) terjadi ketika model berhasil mengklasifikasikan data ulasan dengan tepat ke dalam kategori positif, sedangkan *False Positive* (FP) muncul ketika model keliru mengategorikan data sebagai kelas positif. *True Negative* (TN) menunjukkan keberhasilan model dalam mengidentifikasi kelas negatif secara akurat, sementara *False Negative* (FN) terjadi ketika model gagal mengenali kelas negatif dengan benar.

Tabel 4.2 merepresentasikan confusion matrix digunakan untuk mengukur performa model klasifikasi aspek penyebab kecelakaan lalu lintas dimana A, B, C merepresentasikan kelas aspek penyebab kecelakaan lalu lintas. A untuk aspek Faktor Jalan, B untuk aspek Faktor Kendaraan, dan C untuk aspek Kelalaian Pengemudi.

Tabel 4.2 *Confusion Matrix*

Kelas Aktual	Kelas Prediksi		
	Faktor Jalan (A)	Faktor Kendaraan (B)	Kelalaian Pengemudi (C)
Faktor Jalan (A)	AA	AB	AC
Faktor Kendaraan (B)	BA	BB	BC

Kelalaian Pengemudi (B)	CA	CB	CC
-------------------------	----	----	----

Studi ini menghasilkan tiga kategori klasifikasi sebagai keluaran. Setiap kelas dalam confusion matrix terbagi menjadi empat komponen evaluasi: TP, TN, FP, dan FN. Komponen-komponen tersebut berfungsi untuk mengukur jumlah prediksi *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN) pada masing-masing kategori. Representasi confusion matrix untuk tiap kelas dapat dilihat secara berturut-turut dalam Tabel 4.3, Tabel 4.4, dan Tabel 4.5.

Tabel 4.3 *Confusion Matrix* Kelas Faktor Jalan

Kelas Aktual	Kelas Prediksi	
	Faktor Jalan (A)	Bukan Faktor Jalan
Faktor Jalan (A)	<i>True Positive</i> (AA)	<i>False Negative</i> (AB + AC)
Bukan Faktor Jalan	<i>False Positive</i> (BA + CA)	<i>True Negative</i> (BB + BC + CB + CC)

Tabel 4.4 *Confusion Matrix* Kelas Faktor Kendaraan

Kelas Aktual	Kelas Prediksi	
	Faktor Kendaraan (B)	Bukan Faktor Kendaraan
Faktor Kendaraan (B)	<i>True Positive</i> (BB)	<i>False Negative</i> (BA + BC)
Bukan Faktor kendaraan	<i>False Positive</i> (AB + CB)	<i>True Negative</i> (AA + AC + CA + CC)

Tabel 4.5 *Confusion Matrix* Kelas Kelalaian Pengemudi

Kelas Aktual	Kelas Prediksi	
	Kelalaian Pengemudi (C)	Bukan Kelalaian Pengemudi
Kelalaian Pengemudi (C)	<i>True Positive</i> (CC)	<i>False Negative</i> (CA + CB)
Bukan Kelalaian Pengemudi	<i>False Positive</i> (AC + BC)	<i>True Negative</i> (AA + AB + BA + BB)

Berikut adalah penjelasan mengenai empat komponen utama dari *confusion matrix*:

1. **True Positive (TP)** model *machine learning* berhasil memprediksi data dengan benar sebagai kelas positif. Artinya, data yang sebenarnya memang termasuk dalam kategori positif juga diprediksi sebagai positif oleh model.
2. **True Negative (TN)** model secara tepat mengklasifikasikan data sebagai kelas negatif, dimana data tersebut memang benar-benar bukan termasuk dalam kelas positif yang sedang dianalisis. Komponen ini mengukur kemampuan model untuk menolak atau tidak mengidentifikasi data yang memang bukan bagian dari kelas target.
3. **False Positive (FP)** model salah mengategorikan data sebagai kelas positif, padahal sebenarnya data tersebut termasuk dalam kelas negatif. Kondisi ini sering disebut sebagai "kesalahan tipe I" yang menunjukkan bahwa model terlalu sensitif dalam mendeteksi kelas positif.
4. **False Negative (FN)** merupakan kebalikan dari *False Positive*, yaitu ketika model salah memprediksi data yang sebenarnya positif sebagai kelas negatif. Kesalahan ini disebut juga "kesalahan tipe II" yang mengindikasikan bahwa model kurang sensitif dalam mengenali kelas positif yang seharusnya terdeteksi.

Dari *confusion matrix* yang diperoleh, kita dapat menghitung beberapa metrik evaluasi penting yang memberikan gambaran lebih jelas tentang kinerja model:

1. **Akurasi (Accuracy):** Mengukur seberapa banyak prediksi yang benar dibandingkan dengan total data.

$$Akurasi = \frac{TP + TN}{Total Sampel} \quad (4.1)$$

2. **Precision:** Mengukur seberapa banyak prediksi positif yang benar untuk setiap kelas. *Precision* dihitung untuk masing-masing kelas:

$$Precision = \frac{TP}{TP + FP} \quad (4.2)$$

3. **Recall (Sensitivitas):** Mengukur seberapa banyak data yang benar-benar termasuk dalam kelas tersebut dapat terdeteksi oleh model.

$$Recall = \frac{TP}{TP + FN} \quad (4.3)$$

4. **F1-Score:** Merupakan *harmonic mean* dari *Precision* dan *Recall*, memberikan gambaran tentang keseimbangan antara keduanya.

$$Akurasi = 2 \frac{Precision \cdot Recall}{Precision + Recall} \quad (4.4)$$

4.2 Hasil Uji Coba

Pada Tabel 4.6, ditampilkan berbagai skenario uji coba yang dilakukan untuk mencari model terbaik. Di sana terdapat variasi parameter *Learning rate* dan jumlah *Node* pada *Hidden Layer 1* yang bertujuan untuk memperoleh model yang optimal. Berikut adalah hasilnya:

Tabel 4.6 Hasil *Training*

No.	Jumlah Node (Hidden Layer 1)	Learning rate	Toleransi Kesalahan	Epoch	Time
1	21	0.001	0.01	445.534	6m 57s
2	21	0.0001	0.01	1.489.684	21m 9s
3	301	0.001	0.01	438.015	45m 6s
4	301	0.0001	0.01	1.431.840	128m 27s

Dari Tabel 4.6, dapat dilihat bahwa semakin kecil nilai *Learning rate* dan jumlah *Node* pada *Hidden Layer 1*, semakin banyak *epoch* dan waktu yang

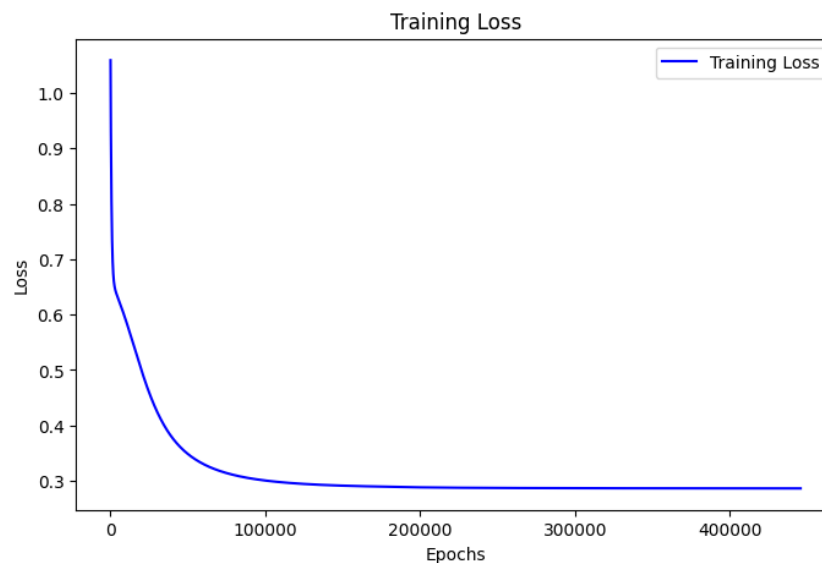
diperlukan untuk mencapai toleransi *error* yang di tetapkan. Berikut adalah hasil evaluasi dengan data testing pada Tabel 4.7.

Tabel 4.7 Hasil Evaluasi

No.	Skenario Uji Coba	Support Data	<i>Accuracy</i>	<i>Precision</i>	<i>Recall</i>	<i>F1-Score</i>
1	1	50	0.80	0.69	0.61	0.63
2	2	50	0.80	0.69	0.61	0.63
3	3	50	0.80	0.69	0.61	0.63
4	4	50	0.80	0.69	0.61	0.63

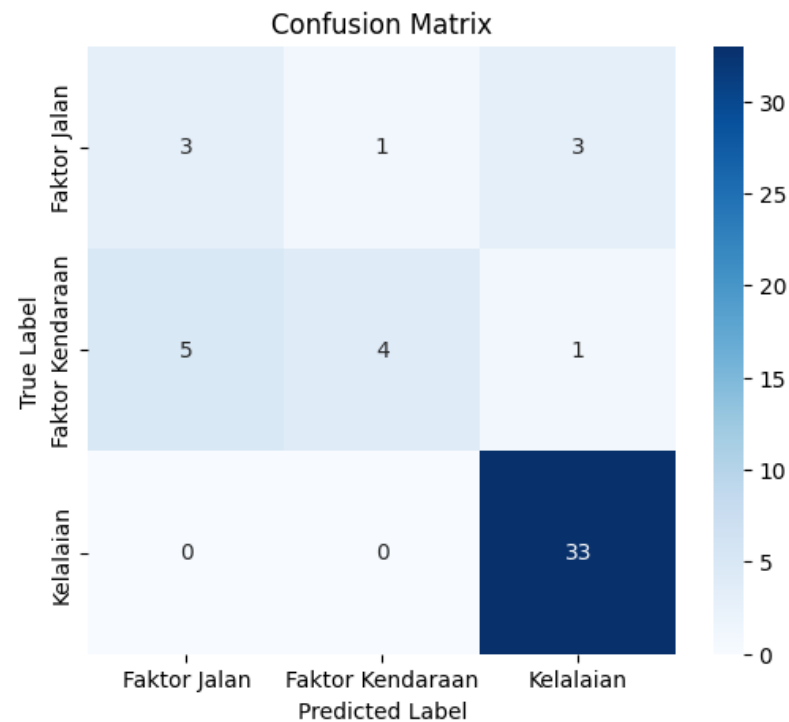
4.2.1 Skenario 1

Dari skenario 1, model mencapai *loss* sebesar 0.28 dengan *epoch* sebanyak 445.534 dan waktu pelatihan 6 menit 25 detik. Untuk memberikan gambaran lebih lanjut, dapat dilihat grafik *loss* yang menunjukkan konvergensi pada gambar 4.1.



Gambar 4.1 Grafik *Loss* Skenario 1

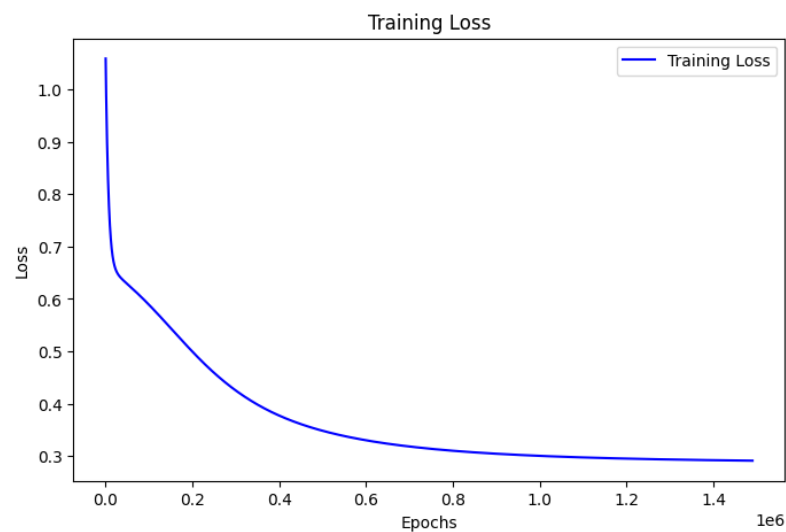
Gambar 4.1 menunjukkan bahwa *loss* menurun dengan stabil hingga mencapai nilai sesuai dengan yang ditetapkan pada skenario . Selanjutnya, hasil pengujian dapat dilihat pada gambar 4.2.

Gambar 4.2 *Confusion Matrix* Skenario 1

Dari gambar 4.2, terlihat bahwa model memiliki performa yang bervariasi antar kelas. Model menunjukkan performa terbaik pada kelas "Kelalaian Pengemudi" dengan *precision* 0.89, *recall* 1.00, dan *f1-score* 0.94 diikuti oleh "Faktor Kendaraan" dengan *precision* 0.80, *recall* 0.40, dan *f1-score* 0.53. Namun, kelas "Faktor Jalan" menunjukkan performa yang lemah dengan *precision* 0.38, *recall* 0.43, dan *f1-score* 0.40 mengindikasikan model kesulitan mendeteksi kelas minoritas ini. Ketidakseimbangan ini disebabkan oleh distribusi data yang tidak merata dimana "Kelalaian Pengemudi" memiliki 33 sampel, "Faktor Kendaraan" 10 sampel, dan "Faktor Jalan" hanya 7 sampel. Secara keseluruhan, model mencapai akurasi 80% dengan *average f1-score* 0.63% yang menunjukkan adanya ketidakseimbangan performa antar kelas.

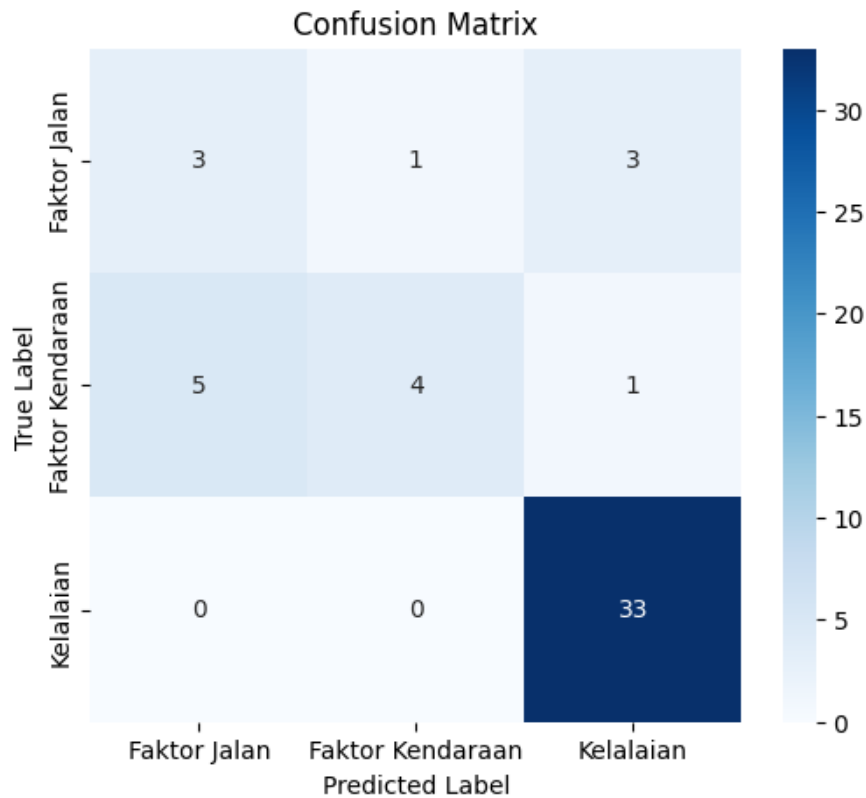
4.2.2 Skenario 2

Pada skenario 2, model mencapai *loss* sebesar 0.29 dengan *epoch* sebanyak 1.489.684 dan waktu pelatihan 21 menit 9 detik. Waktu pelatihan ini lebih panjang dibandingkan skenario sebelumnya. Untuk memberikan gambaran lebih lanjut dapat dilihat grafik *loss* yang menunjukkan konvergensi pada gambar 4.3.



Gambar 4.3 Grafik *Loss* Skenario 2

Gambar 4.3 menunjukkan bahwa *loss* menurun dengan stabil hingga mencapai nilai yang sesuai dengan skenario yang ditetapkan, meskipun pelatihan berlangsung lebih lama. Selanjutnya, hasil pengujian dapat dilihat pada gambar 4.4.

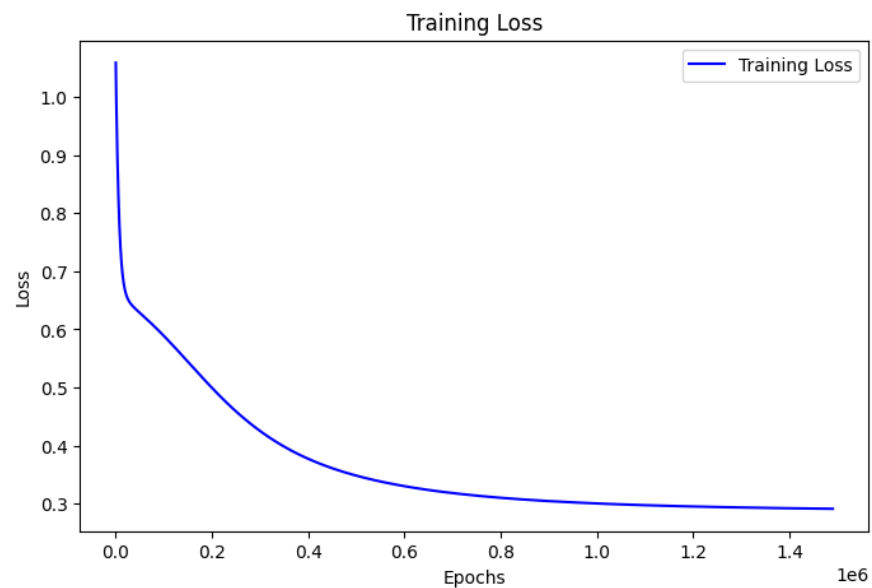


Gambar 4.4 *Confusion Matrix* Skenario 2

Dari gambar 4.4, hasil evaluasi menunjukkan performa yang konsisten dengan skenario 1. Model mencapai akurasi 80% dengan distribusi performa yang sama antar kelas. Kelas "Kelalaian Pengemudi" menunjukkan performa terbaik dengan *precision* 0.89, *recall* 1.00 dan *f1-score* 0.94. Kelas "Faktor Kendaraan" memiliki *precision* 0.80, *recall* 0.40 dan *f1-score* 0.53. Sementara itu kelas "Faktor Jalan" menunjukkan performa dengan *precision* 0.38, *recall* 0.43 dan *f1-score* 0.40. Meskipun waktu pelatihan lebih lama akibat *learning rate* yang lebih kecil, model tetap menghasilkan performa yang sama dengan *average f1-score* 0.63 hal ini menunjukkan bahwa penurunan *learning rate* tidak memberikan peningkatan signifikan pada performa model.

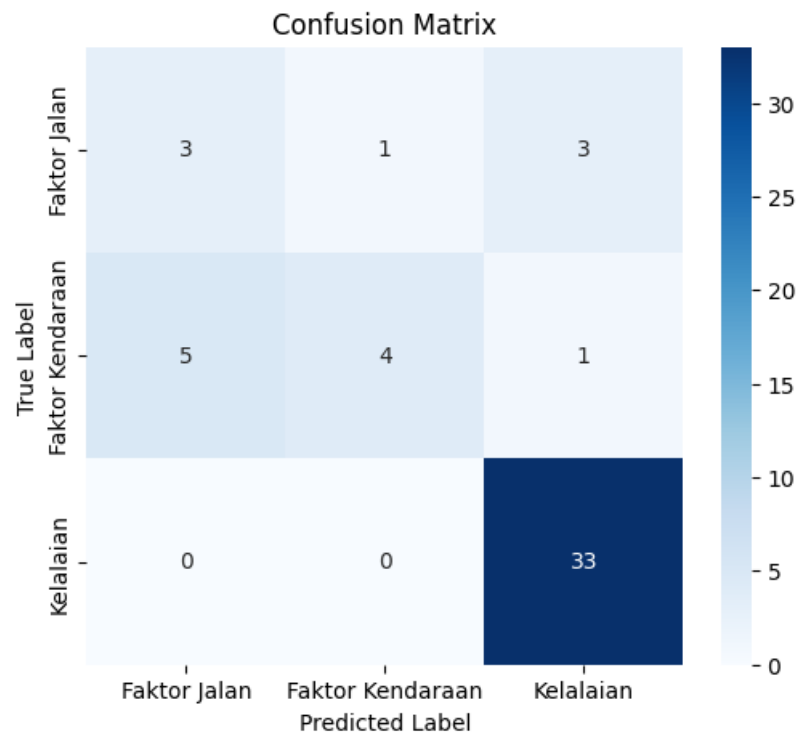
4.2.3 Skenario 3

Pada skenario 3, model mencapai *loss* sebesar 0.28 dengan 438.015 *epoch* dan waktu pelatihan 45 menit 6 detik. Untuk memberikan gambaran lebih lanjut, dapat dilihat grafik *loss* yang menunjukkan konvergensi pada gambar 4.5.



Gambar 4.5 Grafik *Loss* Skenario 3

Gambar 4.5 menunjukkan bahwa *loss* menurun dengan stabil hingga mencapai nilai yang sesuai dengan skenario yang ditetapkan. Selanjutnya, hasil pengujian dapat dilihat pada gambar 4.6.

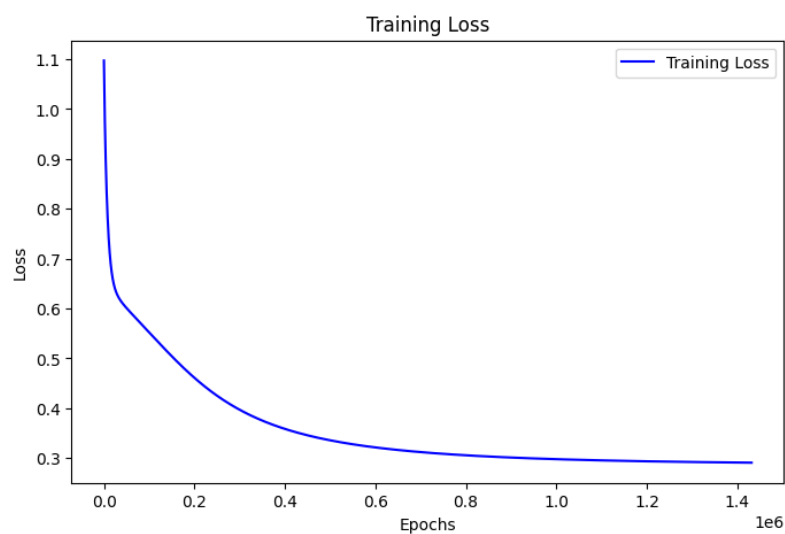


Gambar 4.6 *Confusion Matrix* Skenario 3

Dari gambar 4.6, hasil evaluasi menunjukkan performa yang identik dengan skenario 1 dan 2. Model mencapai akurasi 80% dengan distribusi performa yang sama antar kelas. Model menunjukkan performa terbaik pada kelas "Kelalaian Pengemudi" dengan *precision* 0.89, *recall* 1.00 dan *f1-score* 0.94. Kelas "Faktor Kendaraan" memiliki *precision* 0.80, *recall* 0.40 dan *f1-score* 0.53. Sementara itu kelas "Faktor Jalan" menunjukkan performa dengan *precision* 0.38, *recall* 0.43 dan *f1-score* 0.40. Meskipun menggunakan jumlah *node* yang lebih banyak (301), skenario ini tidak memberikan peningkatan performa dibandingkan skenario 1 dan 2. *Average f1-score* tetap di 0.63 hal ini menunjukkan bahwa peningkatan kompleksitas model tidak selalu menghasilkan peningkatan performa pada dataset ini.

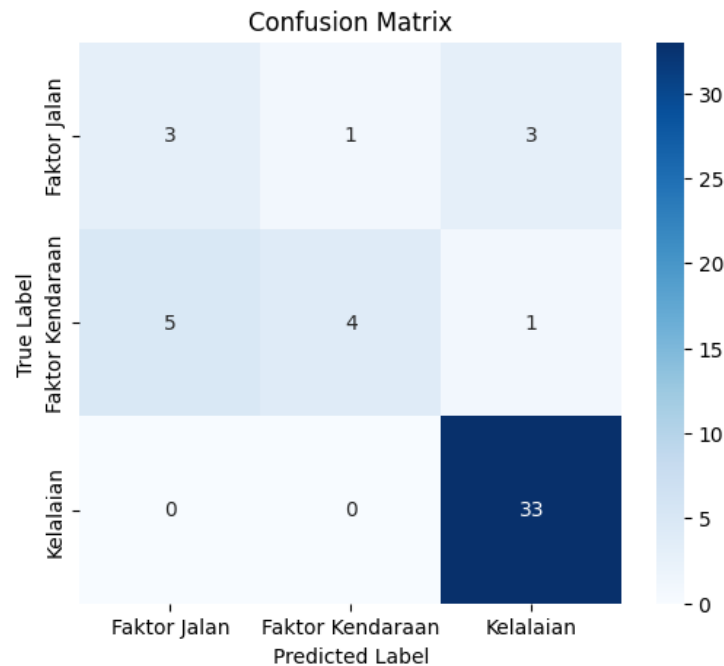
4.2.4 Skenario 4

Pada Skenario 4, model mencapai *loss* sebesar 0.29 dengan 1.431.840 *epoch* dan waktu pelatihan yang paling lama yaitu 128 menit 27 detik. Grafik *loss* yang menunjukkan konvergensi dapat dilihat pada gambar 4.7, yang menunjukkan penurunan *loss* secara stabil hingga mencapai nilai yang sesuai dengan skenario yang ditetapkan.



Gambar 4.7 Grafik *Loss* Skenario 4

Gambar 4.7, menunjukkan penurunan *loss* yang konsisten selama pelatihan meskipun waktu yang dibutuhkan lebih lama. Selanjutnya, hasil pengujian pada skenario ini dapat dilihat pada gambar 4.8 yang menunjukkan *confusion matrix*.



Gambar 4.8 *Confusion Matrix* Skenario 4

Dari gambar 4.8, hasil evaluasi menunjukkan performa yang konsisten dengan ketiga skenario sebelumnya. Model mencapai akurasi 80% dengan distribusi performa yang sama antar kelas. Kelas "Kelalaian Pengemudi" menunjukkan performa yang baik dengan *precision* 0.89, *recall* 1.00 dan *f1-score* 0.94. Kelas "Faktor Kendaraan" memiliki *precision* 0.80, *recall* 0.40 dan *f1-score* 0.53. Kelas "Faktor Jalan" menunjukkan performa dengan *precision* 0.38, *recall* 0.43 dan *f1-score* 0.40. Meskipun menggunakan learning rate yang lebih kecil (0.0001) dan jumlah node yang lebih banyak (301) serta waktu pelatihan yang sangat lama, skenario ini tidak memberikan peningkatan performa dibandingkan skenario lainnya. *Average f1-score* tetap di 0.63. Hal ini mengindikasikan bahwa kombinasi parameter yang lebih kompleks tidak memberikan manfaat tambahan dan justru membutuhkan sumber daya komputasi yang jauh lebih besar.

4.3 Hasil Uji Coba dengan SMOTE

Pada Tabel 4.8, terlihat berbagai skenario uji coba dengan teknik SMOTE yang digunakan untuk mengatasi ketidakseimbangan kelas. Dalam uji coba ini, parameter seperti *Learning rate* dan jumlah *Node* pada *Hidden Layer* 1 divariasikan untuk mendapatkan model yang optimal. Berikut adalah hasilnya:

Tabel 4.8 Hasil Training dengan SMOTE

No.	Jumlah <i>Node</i> (Hidden Layer 1)	<i>Learning rate</i>	Toleransi Kesalahan	<i>Epoch</i>	Time
1	21	0.001	0.01	141.652	3m 51s
2	21	0.0001	0.01	1.490.714	69m 39s
3	301	0.001	0.01	143.455	23m 44s
4	301	0.0001	0.01	1.242.529	226m 30s

Dari Tabel 4.8, dapat dilihat bahwa semakin kecil nilai *Learning rate* dan jumlah *Node* pada *Hidden Layer* 1 semakin banyak *epoch* dan waktu pelatihan yang diperlukan untuk mencapai toleransi kesalahan yang ditetapkan. Hal ini serupa dengan skenario uji coba tanpa SMOTE namun waktu yang dibutuhkan untuk pelatihan lebih lama dibandingkan dengan skenario tanpa SMOTE. Selanjutnya, Tabel 4.9 menunjukkan hasil evaluasi model menggunakan data testing setelah penerapan SMOTE pada masing-masing skenario.

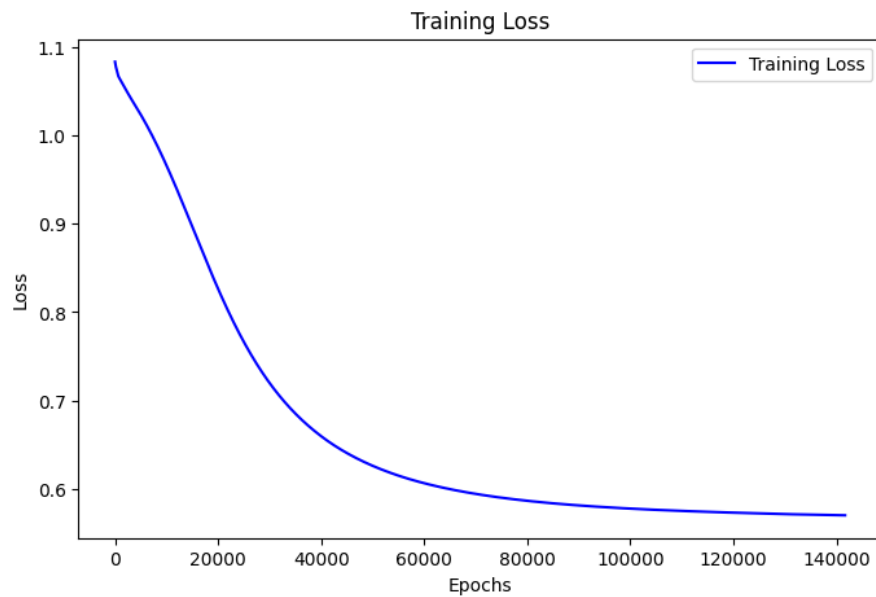
Tabel 4.9 hasil Evaluasi dengan SMOTE

No.	Skenario Uji Coba	Support Data	<i>Accuracy</i>	<i>Precision</i>	<i>Recall</i>	<i>F1-Score</i>
1	1	50	0.82	0.71	0.70	0.70
2	2	50	0.82	0.71	0.70	0.70
3	3	50	0.84	0.73	0.74	0.73
4	4	50	0.84	0.73	0.74	0.73

Dengan penerapan SMOTE, terlihat adanya peningkatan signifikan dalam *Accuracy*, *precision*, *Recall*, dan *F1-Score* dibandingkan dengan hasil uji coba sebelumnya. Pada semua skenario, akurasi meningkat menjadi sekitar 82% pada skenario 1 dan 84% pada skenario 2, 3, dan 4. *Precision* dan *Recall* juga menunjukkan hasil yang lebih baik, mencerminkan bahwa teknik SMOTE berhasil meningkatkan kinerja model, khususnya dalam mendeteksi kelas minoritas dengan lebih baik.

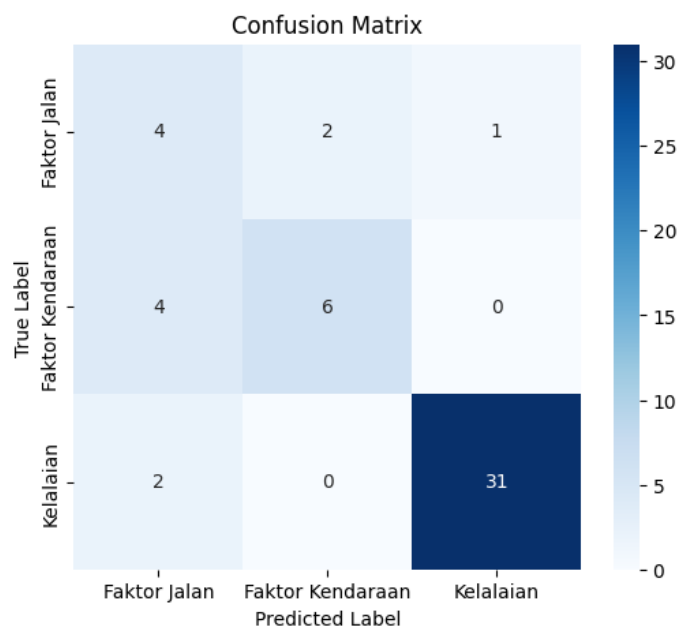
4.3.1 Skenario 1

Pada Skenario 1 dengan penerapan SMOTE, model mencapai *loss* sebesar 0.57 dengan *epoch* sebanyak 141.652 dan waktu pelatihan 3 menit 51 detik. Grafik *loss* yang menunjukkan konvergensi dapat dilihat pada gambar 4.9, yang memperlihatkan penurunan *loss* secara stabil hingga mendekati nilai yang diinginkan.



Gambar 4.9 Grafik *Loss* Skenario 1 dengan SMOTE

Gambar 4.9 menggambarkan penurunan *loss* yang stabil sepanjang *epoch*, yang menunjukkan efektivitas SMOTE dalam mempercepat proses pelatihan model. Selanjutnya, hasil uji coba pada gambar 4.10 menunjukkan *confusion matrix* untuk skenario 1 dengan SMOTE.

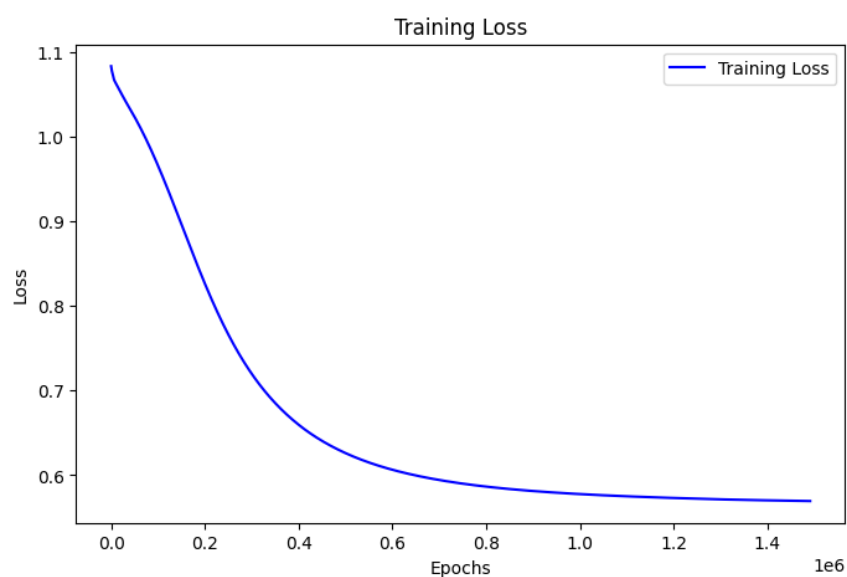


Gambar 4.10 *Confusion Matrix* Skenario Uji Coba 1 dengan SMOTE

Dari hasil evaluasi, terlihat bahwa penerapan SMOTE memberikan peningkatan performa dibandingkan dengan skenario tanpa SMOTE. Model menunjukkan performa yang meningkat pada kelas "Faktor Jalan" dengan *precision* 0.40, *recall* 0.57, dan *f1-score* 0.44. Kelas "Faktor Kendaraan" menunjukkan *precision* 0.75, *recall* 0.60, dan *f1-score* 0.67. Kelas "Kelalaian Pengemudi" menunjukkan *precision* yang tinggi (0.97) dengan *recall* 0.94 dan *f1-score* 0.95. Secara keseluruhan, model mencapai akurasi 82% dengan *average f1-score* 0.70, yang menunjukkan peningkatan dari 0.63 pada skenario tanpa SMOTE.

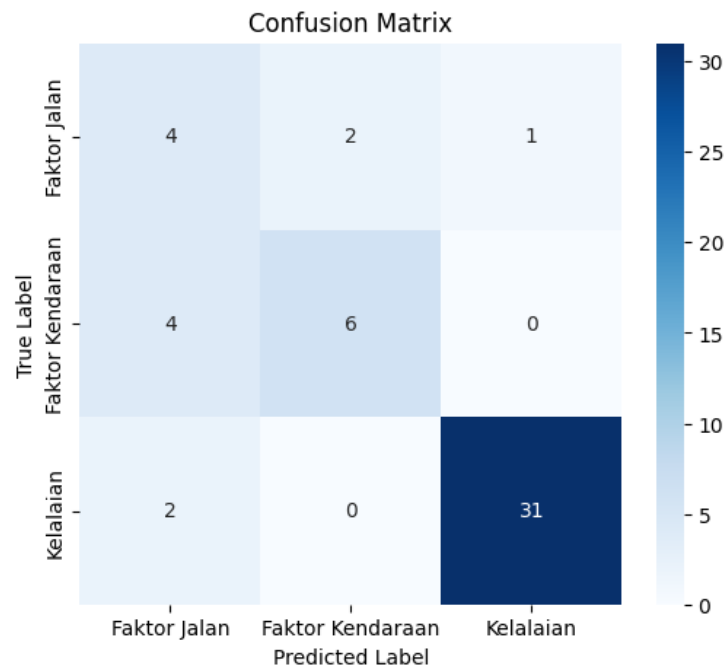
4.3.2 Skenario 2

Pada Skenario 2 dengan penerapan SMOTE, model mencapai *loss* sebesar 0,56 dengan *epoch* sebanyak 1.490.714 dan waktu pelatihan 69 menit 39 detik. Grafik *loss* yang menunjukkan konvergensi dapat dilihat pada gambar 4.11, yang menggambarkan penurunan *loss* secara stabil hingga mencapai nilai yang diinginkan.



Gambar 4.11 Grafik *Loss* Skenario 2 dengan SMOTE

Penurunan *loss* yang stabil menandakan bahwa teknik SMOTE efektif dalam membantu proses pelatihan model. Selanjutnya, gambar 4.12 menunjukkan *confusion matrix* untuk Skenario 2.

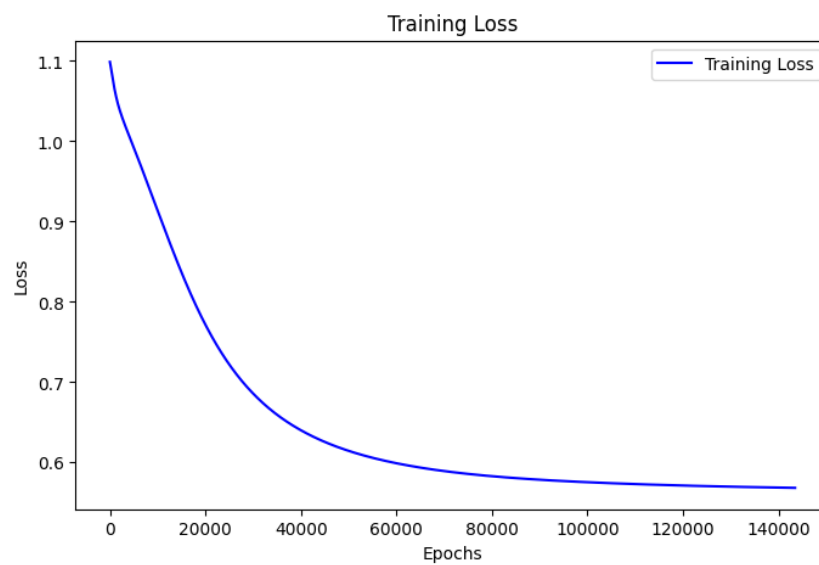


Gambar 4.12 *Confusion Matrix* Skenario 2 dengan SMOTE

Hasil gambar 4.12, menunjukkan performa yang konsisten dengan skenario 1 dengan SMOTE. Model mencapai akurasi 82% dengan distribusi performa yang sama antar kelas. Kelas "Faktor Jalan" menunjukkan *precision* 0.40, *recall* 0.57 dan *f1-score* 0.44. Kelas "Faktor Kendaraan" memiliki *precision* 0.75, *recall* 0.60 dan *f1-score* 0.67. Kelas "Kelalaian Pengemudi" tetap menunjukkan performa terbaik dengan *precision* 0.97, *recall* 0.94 dan *f1-score* 0.95. *Average f1-score* tetap di 0.70. Meskipun waktu pelatihan lebih lama akibat *learning rate* yang lebih kecil, model menghasilkan performa yang sama dengan skenario 1.

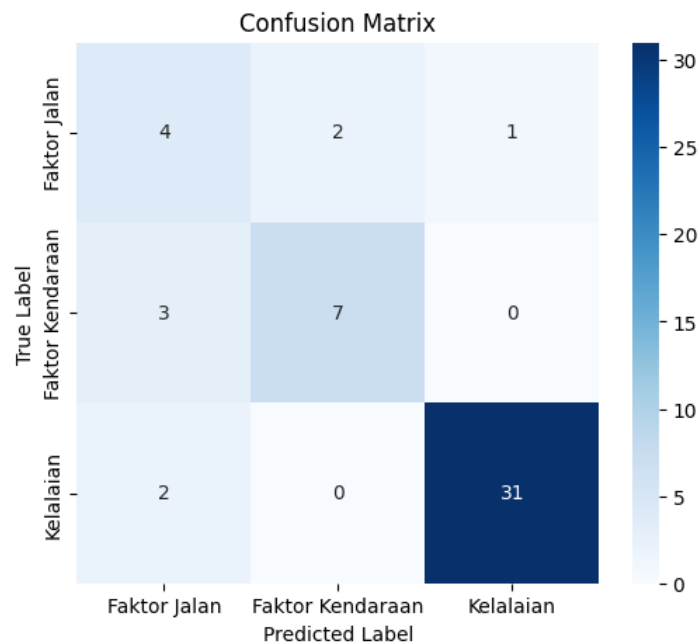
4.3.3 Skenario 3

Pada Skenario 3 dengan penerapan SMOTE, model mencapai *loss* sebesar 0.56 dengan *epoch* sebanyak 143.455 dan waktu pelatihan 23 menit 44 detik. Grafik *loss* konvergen dapat dilihat pada gambar 4.13, yang menunjukkan penurunan *loss* secara stabil hingga mendekati nilai yang diinginkan.



Gambar 4.13 Grafik *Loss* Skenario 3 dengan SMOTE

Penurunan *loss* yang stabil ini menggambarkan efektivitas SMOTE dalam meningkatkan kinerja model dan mempercepat proses konvergensi. Selanjutnya, gambar 4.14 menunjukkan *confusion matrix* untuk Skenario 3.



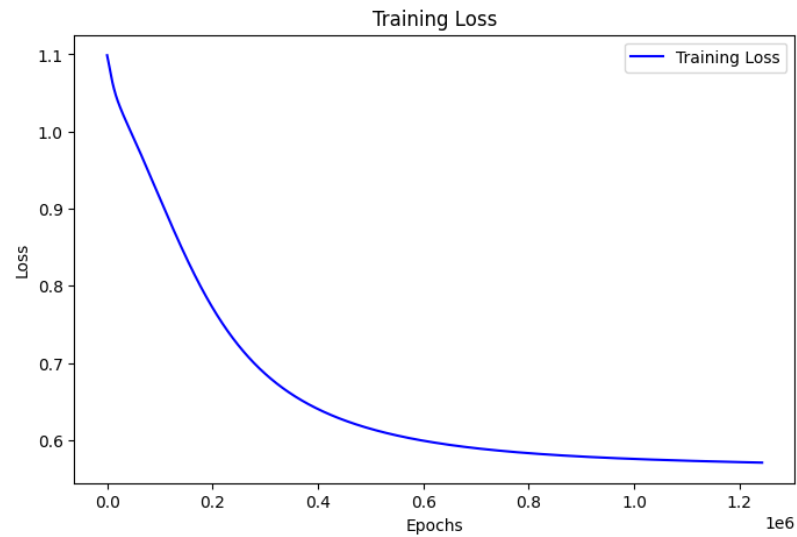
Gambar 4.14 *Confusion Matrix* Skenario 3 dengan SMOTE

Hasil gambar 4.14, menunjukkan peningkatan performa dibandingkan skenario 1 dan 2 dengan SMOTE. Model mencapai akurasi 84% dengan performa yang lebih baik pada semua kelas. Kelas "Faktor Jalan" menunjukkan *precision* 0.50, *recall* 0.71 dan *f1-score* 0.59. Kelas "Faktor Kendaraan" memiliki *precision* 0.82, *recall* 0.70 dan *f1-score* 0.76. Kelas "Kelalaian Pengemudi" tetap menunjukkan performa terbaik dengan *precision* 0.97, *recall* 0.91 dan *f1-score* 0.94. *Average f1-score* meningkat menjadi 0.73. Peningkatan jumlah *node* pada *hidden layer* memberikan kontribusi positif terhadap performa model.

4.3.4 Skenario 4

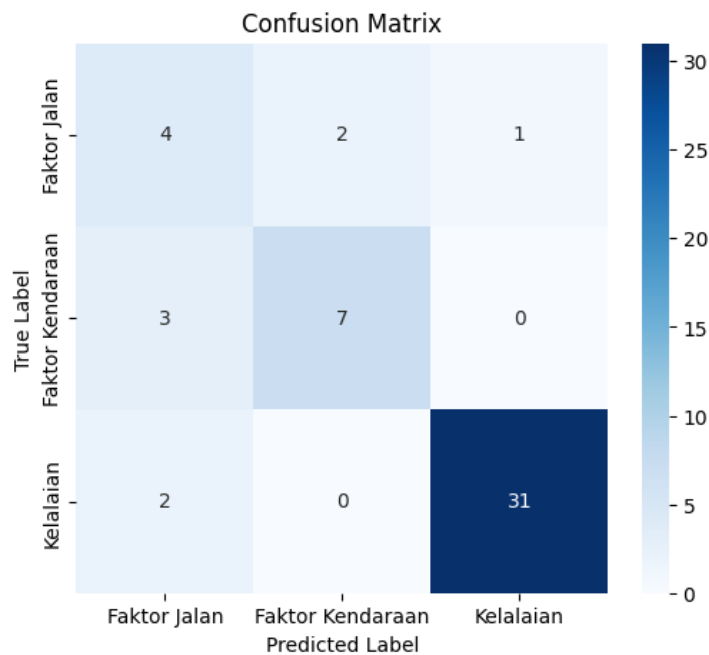
Pada Skenario 4 dengan penerapan SMOTE, model mencapai *loss* sebesar 0.57 dengan *epoch* sebanyak 1.242.529 dan waktu pelatihan yang paling lama yaitu

226 menit 30 detik. Grafik *loss* dapat dilihat pada gambar 4.15, yang menunjukkan penurunan *loss* secara stabil hingga mencapai nilai yang diinginkan.



Gambar 4.15 Grafik *Loss* Skenario 4 dengan SMOTE

Penurunan *loss* yang stabil ini menandakan efektivitas SMOTE dalam meningkatkan kinerja model. Selanjutnya, gambar 4.16 menunjukkan *confusion matrix* untuk Skenario 4.



Gambar 4.16 *Confusion Matrix* Skenario 4 dengan SMOTE

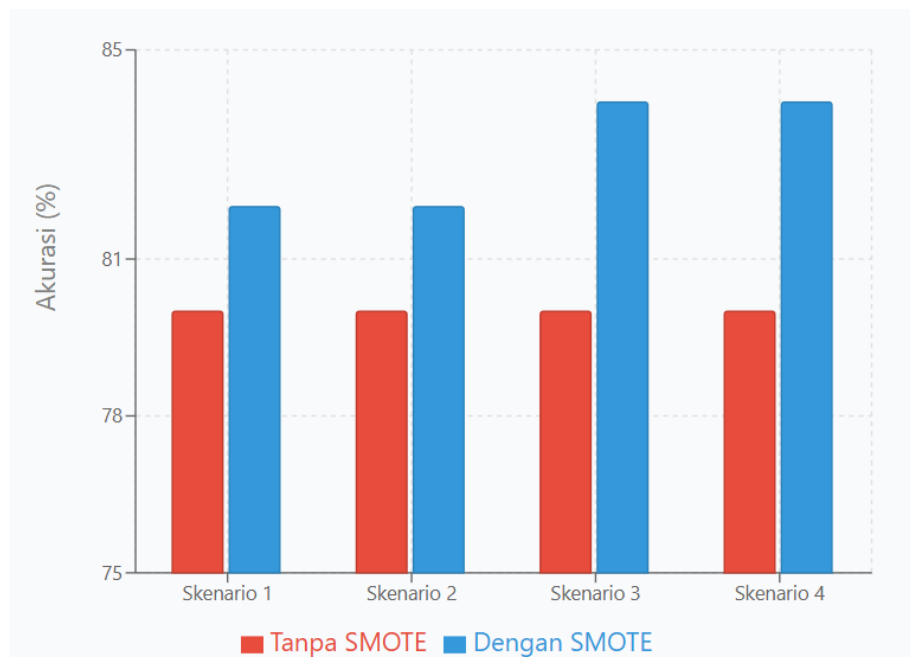
Gambar 4.16, menunjukkan performa yang identik dengan skenario 3 dengan SMOTE. Model mencapai akurasi 84% dengan distribusi performa yang sama antar kelas. Kelas "Faktor Jalan" menunjukkan *precision* 0.50, *recall* 0.71 dan *f1-score* 0.59. Kelas "Faktor Kendaraan" memiliki *precision* 0.82, *recall* 0.70 dan *f1-score* 0.76. Kelas "Kelalaian Pengemudi" tetap menunjukkan performa terbaik dengan *precision* 0.97, *recall* 0.91 dan *f1-score* 0.94. Meskipun menggunakan *learning rate* yang lebih kecil (0.0001) dengan jumlah *node* yang banyak (301) serta waktu pelatihan yang sangat lama, skenario ini tidak memberikan peningkatan performa tambahan. *Average f1-score* tetap di 0.73. Hal ini menunjukkan bahwa kombinasi parameter kompleks tidak selalu memberikan hasil yang lebih baik.

4.4 Pembahasan

Penelitian ini bertujuan untuk mengukur kinerja model *Artificial Neural Network* (ANN) dalam mengklasifikasikan berita online berdasarkan metrik *Accuracy*, *Precision*, *Recall*, dan *F1-Score*. Berdasarkan hasil pengujian yang telah dilakukan dapat dianalisis beberapa aspek penting terkait performa model dalam tugas klasifikasi berita kecelakaan lalu lintas.

Hasil pengujian menunjukkan bahwa model ANN mampu melakukan klasifikasi berita dengan tingkat akurasi yang bervariasi tergantung pada konfigurasi parameter yang digunakan. Pada pengujian tanpa penyeimbangan kelas, model mencapai akurasi tertinggi sebesar 80% pada semua skenario dengan *precision* berkisar 0.69, *recall* 0.61, dan *F1-Score* 0.63. Hasil analisis menunjukkan bahwa model cenderung memiliki performa yang baik pada kelas mayoritas "Kelalaian Pengemudi" dengan *precision* 0.89 dan *recall* 1.00. Namun, performa pada kelas minoritas sangat bervariasi: kelas "Faktor Kendaraan" menunjukkan performa yang relatif baik dengan *precision* 0.80 dan *recall* 0.40, sedangkan kelas "Faktor Jalan" menunjukkan performa yang sangat lemah dengan *precision* 0.38 dan *recall* 0.43. Hal ini mengindikasikan bahwa ketidakseimbangan distribusi data menjadi tantangan utama dalam klasifikasi berita online.

Ketika diterapkan teknik penyeimbangan data, model menunjukkan peningkatan yang konsisten dalam semua metrik evaluasi. Berdasarkan perbandingan Tabel 4.7 dan Tabel 4.9 terlihat peningkatan yang cukup baik:



Gambar 4.17 Bar Chart Perbandingan Akurasi

Akurasi meningkat dari rentang 80% menjadi 82-84%. Peningkatan paling signifikan terlihat pada kelas minoritas "Faktor Jalan" yang mengalami peningkatan *recall* dari 0.43 menjadi 0.57-0.71. Kelas "Faktor Kendaraan" juga mengalami peningkatan *recall* dari 0.40 menjadi 0.60-0.70. Peningkatan ini menunjukkan bahwa model ANN memiliki kapasitas yang baik untuk mempelajari pola klasifikasi berita, namun performanya sangat dipengaruhi oleh distribusi data *training*. Model mampu mencapai performa yang sangat baik ketika setiap kelas memiliki representasi yang memadai dalam dataset.

4.4.1 Pengaruh Konfigurasi Parameter

Hasil pengujian menunjukkan bahwa *Learning rate* memiliki dampak signifikan terhadap efisiensi pelatihan model. Skenario dengan *Learning rate* 0.001 mencapai konvergensi dalam waktu yang lebih singkat (438.015-445.534 *epoch*)

dibandingkan dengan *Learning rate* 0.0001 (1.431.840-1.489.684 *epoch*). Meskipun *learning rate* yang lebih kecil (0.0001) memerlukan waktu pelatihan yang jauh lebih lama, semua skenario menghasilkan akurasi yang sama 80% tanpa SMOTE. Hal ini menunjukkan bahwa *learning rate* tidak memberikan perbedaan signifikan pada performa akhir model.

Jumlah *node hidden layer* perbandingan antara 21 *node* dan 301 *node* pada *hidden layer* menunjukkan hasil yang menarik. Tanpa menggunakan SMOTE, peningkatan jumlah *node* dari 21 ke 301 tidak memberikan peningkatan akurasi tetap di 80%. Dengan penerapan SMOTE terdapat perbedaan hasil. Skenario dengan 21 *node* (skenario 1 dan 2) mencapai akurasi 82%, sedangkan skenario dengan 301 *node* (skenario 3 dan 4) mencapai akurasi 84%. Hal ini menunjukkan bahwa peningkatan kompleksitas model memberikan manfaat ketika data sudah seimbang.

4.4.2 Kemampuan Model dalam Klasifikasi

Model ANN menunjukkan kemampuan yang baik dalam menangani tugas klasifikasi multi-kelas pada domain berita kecelakaan lalu lintas. Dapat dilihat analisis per-kelas pada tabel 4.10:

Tabel 4. 10 Perbandingan Performa Model per Kelas

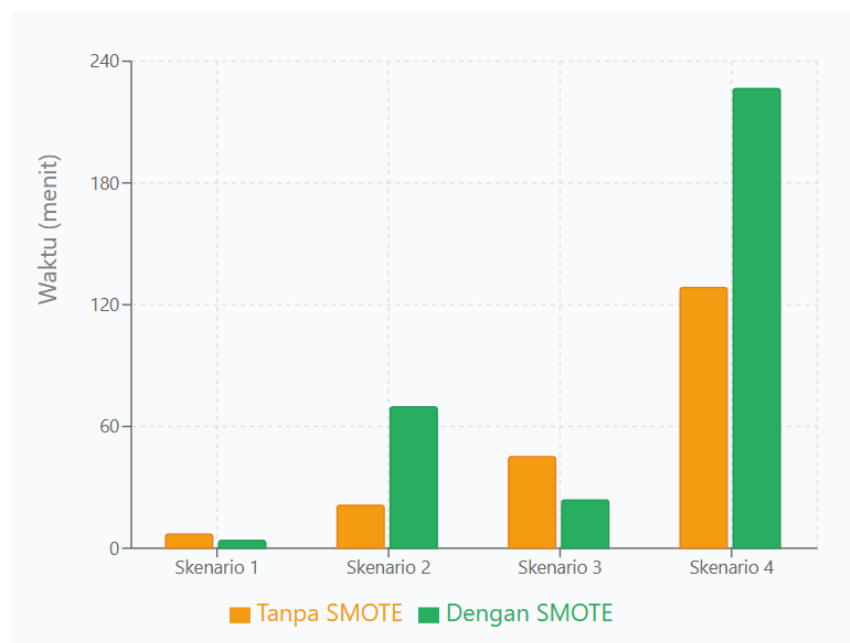
Kelas	Metrik	Tanpa SMOTE	SMOTE	Perubahan
Kelalaian Pengemudi	<i>Precision</i>	0.89	0.97	+8%
	<i>Recall</i>	1.00	0.91-0.94	-6%
	<i>F1-Score</i>	0.94	0.94-0.95	+3%
Faktor Kendaraan	<i>Precision</i>	0.80	0.75-0.82	-
	<i>Recall</i>	0.40	0.60-0.70	+75%
	<i>F1-Score</i>	0.53	0.67-0.76	+48%
Faktor Jalan	<i>Precision</i>	0.38	0.40-0.50	+76%
	<i>Recall</i>	0.43	0.57-0.71	+133%
	<i>F1-Score</i>	0.40	0.44-0.59	+93%

1. **Kelas "Kelalaian Pengemudi"**: Model ANN menunjukkan performa konsisten yang baik pada kelas ini. Model mencapai *recall* sempurna (1.00) dengan *f1-score* 0.94. Setelah penyeimbangan data terjadi sedikit penurunan *recall* menjadi 0.91-0.94 namun *precision* meningkat menjadi 0.97. *F1-score* tetap stabil di 0.94-0.95. Ini mencerminkan kemampuan model ANN yang sangat baik dalam mengenali pola teks yang berkaitan dengan faktor human *error*.
2. **Kelas "Faktor Kendaraan"**: Terjadi peningkatan signifikan setelah penyeimbangan data. Sebelumnya *recall* hanya 0.40 dengan *f1-score* 0.53. Setelah SMOTE, *recall* meningkat menjadi 0.60-0.70 dan *f1-score* meningkat menjadi 0.67-0.76. *Precision* relatif stabil di kisaran 0.75-0.82.

3. **Kelas "Faktor Jalan"**: Mengalami peningkatan paling signifikan setelah penyeimbangan data. Dari performa rendah dengan *precision* 0.38, *recall* 0.43 dan *f1-score* 0.40, meningkat menjadi *precision* 0.40-0.50, *recall* 0.57-0.71 dan *f1-score* 0.44-0.59. Ini menunjukkan bahwa penyeimbangan data berhasil membantu model ANN dalam menangani kelas minoritas yang sebelumnya sulit dikenali.

4.4.3 Efisiensi Komputasi dan Konvergensi

Analisis waktu pelatihan menunjukkan bahwa model ANN memerlukan *resources* komputasi yang bervariasi tergantung konfigurasi parameter. Berdasarkan perbandingan Tabel 4.6 dan Tabel 4.8 waktu pelatihan meningkat secara signifikan pada model dengan penyeimbangan data:



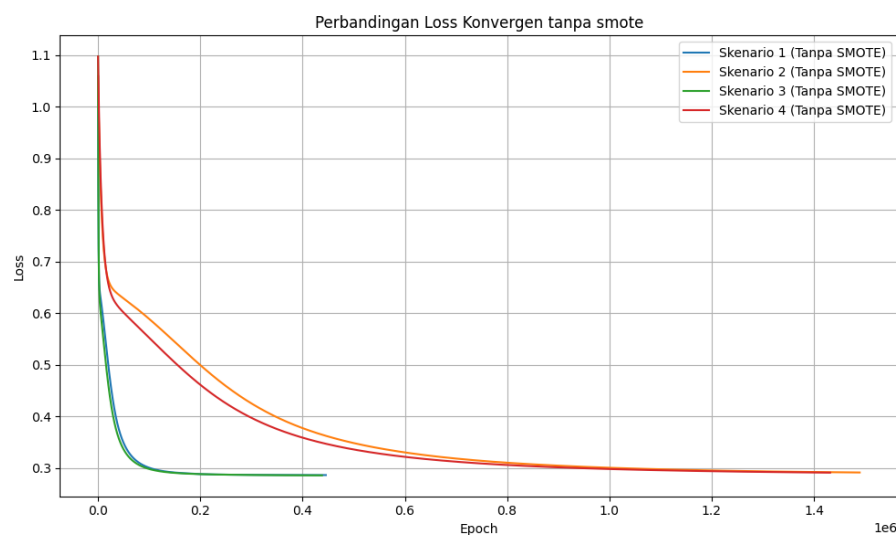
Gambar 4.18 Perbandingan Waktu Pelatihan

- Skenario 1: dari 6m 57s menjadi 3m 51s (penurunan 3 kali)
- Skenario 2: dari 21m 9s menjadi 69m 39s (peningkatan 2.3 kali)

- c) Skenario 3: dari 45m 6s menjadi 23m 44s (penurunan 2 kali)
- d) Skenario 4: dari 128m 27s menjadi 226m 30s (peningkatan 2 kali)

Peningkatan waktu pelatihan ini disebabkan oleh bertambahnya jumlah data yang harus diproses setelah augmentasi SMOTE dari 450 data *training* menjadi 1.038 data *training*, namun sebanding dengan peningkatan performa model yang signifikan.

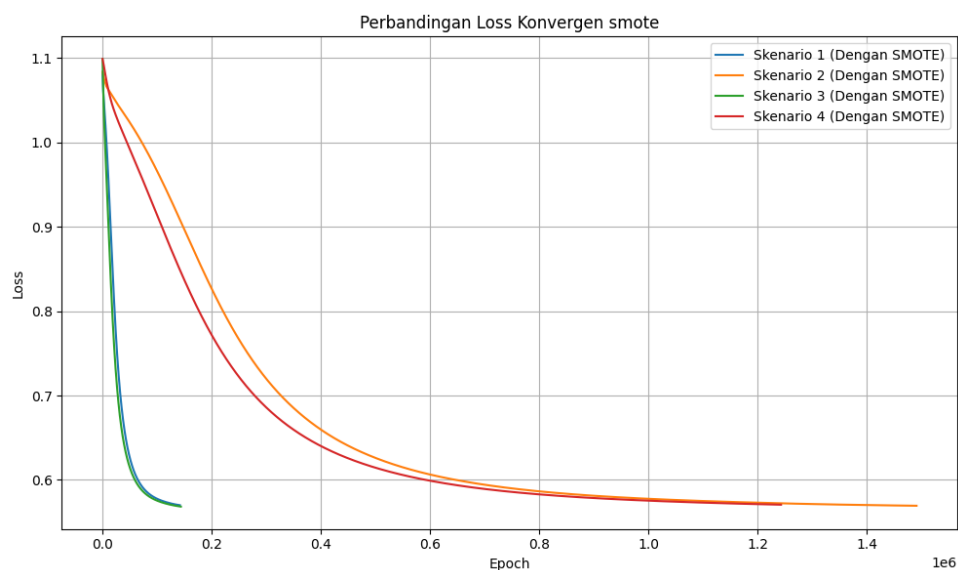
Grafik konvergensi *loss* pada semua skenario menunjukkan pola penurunan yang stabil dan konsisten. Berdasarkan gambar 4.19 dan 4.20 menunjukkan pengaruh signifikan *learning rate* terhadap efisiensi pelatihan. Model dengan *learning rate* 0.001 mencapai konvergensi dalam waktu singkat (<450.000 *epoch*). Sementara *learning rate* 0.0001 memerlukan 1.200.000-1.500.000 *epoch*. Meskipun demikian semua model berhasil mencapai konvergensi yang stabil



Gambar 4.19 Perbandingan *Loss* Konvergen tanpa SMOTE

Berdasarkan gambar 4.19 terlihat jelas bahwa nilai *learning rate* memiliki pengaruh yang signifikan terhadap kecepatan konvergensi model selama pelatihan.

Skenario 1 dan 3 yang menggunakan *learning rate* lebih tinggi sebesar 0.001 menunjukkan konvergensi yang sangat cepat. Mereka berhasil mencapai titik optimal dalam kurang dari 450.000 *epoch*. Sebaliknya Skenario 2 dan 4 dengan *learning rate* yang lebih rendah (0.0001) memerlukan jumlah *epoch* yang jauh lebih banyak. Mereka berkisar antara 1.400.000 hingga 1.500.000 untuk mencapai tingkat konvergensi yang sama. Meskipun terdapat perbedaan drastis dalam kecepatan pelatihan, perlu dicatat bahwa semua skenario dimulai dari tingkat *loss* awal yang serupa yaitu antara 0.6 hingga 1.1. Pada akhirnya semua berhasil mencapai tingkat *loss* akhir yang konvergen mendekati nilai 0.28-0.29.



Gambar 4.20 Perbandingan *Loss* Konvergen dengan SMOTE

Analisis pada gambar 4.20 menunjukkan bahwa pola konvergensi yang terjadi secara umum mirip dengan skenario tanpa SMOTE. Pengaruh *learning rate* terhadap kecepatan pelatihan tetap dominan. Meskipun demikian penerapan

SMOTE memperkenalkan beberapa perbedaan yang signifikan. Tercatat bahwa nilai *loss* awal sedikit lebih tinggi berada di kisaran 1.0 hingga 1.1. Hal ini kemungkinan disebabkan oleh peningkatan ukuran dan variasi data *training* setelah proses augmentasi. Skenario 1 dan 3 dengan *learning rate* lebih tinggi tetap menunjukkan konvergensi yang lebih cepat dibandingkan Skenario 2 dan 4. Namun perbedaan yang paling mencolok adalah pada hasil akhir. Pelatihan dengan SMOTE mencapai tingkat *loss* konvergen yang lebih tinggi yakni sekitar 0.56-0.59 jika dibandingkan dengan hasil pada dataset yang tidak seimbang.

Perbedaan *loss* akhir antara model tanpa SMOTE (sekitar 0.28-0.29) dan dengan SMOTE (sekitar 0.56-0.59) tidak mengindikasikan performa yang lebih buruk. Sebaliknya ini menunjukkan bahwa model dengan SMOTE belajar dari data *training* yang lebih kompleks dan seimbang. Meskipun *loss* lebih tinggi, metrik evaluasi seperti akurasi dan F1-Score justru menunjukkan peningkatan. Hal ini menunjukkan bahwa teknik SMOTE tidak hanya tidak memperlambat konvergensi model. Bahkan sedikit mempercepat pencapaian *loss* target kemungkinan karena distribusi data yang lebih seimbang memudahkan proses pembelajaran.

4.4.4 Integrasi Ayat dan Hadis

Hasil penelitian menunjukkan bahwa model ANN dapat menjadi solusi yang efektif untuk klasifikasi berita online, khususnya dalam kecelakaan lalu lintas. Kemampuan model untuk mencapai akurasi hingga 84% menunjukkan potensi untuk menganalisis dan mengategorikan berita berdasarkan faktor penyebab kecelakaan. Dalam perspektif Islam, proses klasifikasi dan pengelompokan merupakan bagian dari *sunnatullah* yang Allah SWT tetapkan dalam kehidupan

manusia untuk memahami realitas dengan lebih baik. Allah SWT berfirman dalam Al-Qur'an:

يَا أَيُّهَا النَّاسُ إِنَّا خَلَقْنَاكُمْ مِنْ ذَكَرٍ وَأُنْثَىٰ وَجَعَلْنَاكُمْ شُعُوبًا وَقَبَائِلَ لِتَعَارَفُوا إِنَّ أَكْرَمَكُمْ عِنْدَ اللَّهِ أَتْقَاهُمْ إِنَّ اللَّهَ عَلِيمٌ
خَبِيرٌ ﴿١٣﴾

"Wahai manusia, sesungguhnya Kami telah menciptakan kamu dari laki-laki dan perempuan, dan Kami jadikan kamu bersuku-suku dan berbangsa-bangsa, agar kamu saling kenal-mengenal. Sesungguhnya orang yang paling mulia di antara kamu di sisi Allah adalah orang yang paling bertakwa. Sesungguhnya Allah Maha Mengetahui lagi Maha Teliti." (QS. Al-Hujurat: 13)

Menurut tafsir *Fi Zhilalil Qur'an* karya Sayyid Quthb, ayat ini menegaskan bahwa keragaman suku dan bangsa bukan untuk saling membanggakan diri atau merasa lebih tinggi dari yang lain, melainkan untuk saling mengenal dan memahami. Sedangkan dalam tafsir *Al-Munir* karya Wahbah Az-Zuhaili, dijelaskan bahwa kemuliaan seseorang tidak ditentukan oleh keturunan, harta, atau jabatan melainkan oleh ketakwaannya kepada Allah SWT (Tafsir Al-Munir, jilid 27, hal. 260). Ayat ini mengingatkan kita akan pentingnya pengelompokan dan keberagaman dalam kehidupan sosial manusia. Konsep klasifikasi dalam penelitian ini sejalan dengan prinsip tersebut, di mana pengelompokan berita kecelakaan ke dalam kategori-kategori membantu memahami pola dan karakteristik setiap jenis kecelakaan dengan lebih baik. Implementasi klasifikasi berita kecelakaan lalu lintas dalam penelitian ini mencerminkan dua dimensi hubungan dalam Islam:

1. ***Muamalah Ma'a Allah***

Konsep *hablum minAllah* (hubungan dengan Allah SWT) juga relevan dalam konteks penelitian ini khususnya dalam memahami faktor "Kelalaian Pengemudi"

yang mendominasi penyebab kecelakaan. Dalam Islam, ketaatan merupakan cerminan dari iman yang sempurna. Allah SWT berfirman:

يَا أَيُّهَا الَّذِينَ آمَنُوا أَطِيعُوا اللَّهَ وَأَطِيعُوا الرَّسُولَ وَأُولَى الْأَمْرِ مِنْكُمْ فَإِنْ تَنَازَعْتُمْ فِي شَيْءٍ فَرُدُّوهُ إِلَى اللَّهِ وَالرَّسُولِ إِنْ كُنْتُمْ تُؤْمِنُونَ بِاللَّهِ وَالْيَوْمِ الْآخِرِ ذَلِكَ خَيْرٌ وَأَحْسَنُ تَأْوِيلًا ﴿٥٩﴾

"Wahai orang-orang yang beriman, taatilah Allah dan taatilah Rasul-Nya dan ulil amri di antara kamu. Kemudian jika kamu berlainan pendapat tentang sesuatu, maka kembalikanlah kepada Allah dan Rasul-Nya, jika kamu benar-benar beriman kepada Allah dan hari akhir. Yang demikian itu lebih baik dan lebih baik akibatnya" (QS. An-Nisa: 59)

Menurut Dr. KH. Muhammad Afifuddin Dimiyathi, dijelaskan bahwa ayat ini memerintahkan orang-orang beriman untuk mentaati Allah SWT dan Rasulullah SAW dengan melaksanakan perintah-perintah mereka dan menjauhi larangan-larangan mereka. Ketaatan kepada ulil amri (pemimpin) adalah kepada para pemimpin dari kalangan penguasa dan ulama yang sejalan dengan Allah SWT dan Rasulullah SAW dalam mengutamakan keadilan dan kebenaran (*Hidayatul Quran fi Tafsir al-Qur'an bi al-Qur'an*, Jilid 1, hal 311-312). Dalam konteks keselamatan lalu lintas, ayat ini memberikan gambaran yang mendalam. Ketaatan terhadap peraturan lalu lintas dapat dipahami dari ketaatan kepada ulil amri yang telah menetapkan aturan-aturan tersebut demi kemaslahatan bersama. Ketika model ANN berhasil mengklasifikasikan "Kelalaian Pengemudi" dengan akurasi tinggi, hal ini sebenarnya mengidentifikasi pelanggaran terhadap prinsip ketaatan yang diajarkan Islam. Setiap kelalaian di jalan raya seperti melanggar rambu lalu lintas, menerobos lampu merah, atau mengemudi ugal-ugalan bukan sekadar pelanggaran hukum negara tetapi juga bentuk pembangkangan terhadap prinsip ketaatan yang Allah perintahkan. Maka dengan mengidentifikasi dan mengklasifikasikan pola-

pola kelalaian, model ini dapat membantu meningkatkan kesadaran bagi pengemudi sebagai bentuk ketaatan terhadap aturan yang bertujuan melindungi nyawa

2. *Muamalah Ma'a An-Nas*

Di sisi lain, *hablum minannas* (hubungan dengan sesama manusia) mengingatkan kita akan tanggung jawab sosial dalam setiap upaya kita. Dalam hal ini, klasifikasi data terkait kecelakaan lalu lintas tidak hanya bertujuan untuk analisis teknis, tetapi juga untuk meningkatkan keselamatan dan kesejahteraan masyarakat. Rasulullah SAW bersabda:

مَنْ دَلَّ عَلَى خَيْرٍ فَلَهُ مِثْلُ أَجْرِ فَاعِلِهِ

“Barangsiapa yang menunjuki kepada kebaikan maka dia akan mendapatkan pahala seperti pahala orang yang mengerjakannya” (HR. Muslim no. 1893).

Dalam *Syarah Shahih Muslim* karya Imam Nawawi, dijelaskan bahwa hadis ini mencakup segala bentuk kebaikan termasuk ilmu yang bermanfaat dan petunjuk yang benar (*Syarh al-Nawawi ‘alā Muslim*, jilid 13, hal. 39). Penelitian ini berkontribusi dalam menunjukkan jalan kebaikan melalui identifikasi pola kecelakaan yang dapat digunakan untuk pencegahan. Kemampuan model dalam mengklasifikasikan faktor-faktor kecelakaan memberikan informasi berharga bagi pihak berwenang untuk merancang kebijakan yang lebih efektif dalam melindungi nyawa manusia di jalan raya.

Dengan demikian, pengembangan teknologi klasifikasi berita kecelakaan lalu lintas ini bukan hanya pencapaian teknis semata, tetapi juga merupakan ikhtiar untuk mewujudkan kemaslahatan umat sesuai dengan prinsip-prinsip Islam. Model yang mampu mengidentifikasi dan mengklasifikasikan penyebab kecelakaan

dengan akurat dapat menjadi sarana untuk meningkatkan kesadaran, mendorong ketaatan terhadap aturan, dan pada akhirnya menyelamatkan nyawa manusia yang merupakan tujuan utama syariat Islam (maqashid syariah).

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Penelitian ini berhasil mendemonstrasikan efektivitas Neural Network dalam klasifikasi berita kecelakaan lalu lintas dengan akurasi optimal mencapai 84%. Model terbukti mampu mengklasifikasikan berita berdasarkan tiga kategori faktor penyebab kecelakaan. Kategori tersebut adalah Kelalaian Pengemudi, Faktor Kendaraan dan Faktor Jalan. Konfigurasi paling optimal menggunakan kombinasi teknik SMOTE dengan parameter 301 *nodes* dan *learning rate* 0.001. *Learning rate* 0.001 terbukti lebih efisien dengan waktu konvergensi yang jauh lebih cepat dibandingkan 0.0001. Integrasi *FastText embedding* juga terbukti efektif dalam menangkap makna semantik kata-kata domain-spesifik kecelakaan lalu lintas.

Tantangan utama yang dihadapi adalah ketidakseimbangan distribusi data antar kelas. Masalah ini berhasil diatasi melalui penerapan teknik SMOTE. Teknik ini meningkatkan akurasi dari 80% menjadi 82-84% dengan peningkatan paling signifikan pada kelas minoritas. Analisis performa per kelas menunjukkan bahwa kelas "Kelalaian Pengemudi" mencapai performa tertinggi. Kelas ini memiliki nilai *precision* 0.97, *recall* 0.91-0.94 dan F1-Score 0.94-0.95. Hasil ini mengindikasikan bahwa model sangat akurat dalam mengidentifikasi faktor human *error* sebagai penyebab dominan kecelakaan lalu lintas. Kelas "Faktor Kendaraan" menunjukkan performa menengah dengan F1-Score 0.67-0.76. Kelas "Faktor Jalan" memiliki

performa terendah dengan F1-Score 0.44-0.59. Meskipun demikian kelas ini tetap menunjukkan peningkatan signifikan setelah penerapan SMOTE.

Temuan bahwa "Kelalaian Pengemudi" memiliki performa klasifikasi tertinggi memberikan *insight* bagi pihak kepolisian. Dominasi faktor human *error* ini menunjukkan bahwa strategi pencegahan kecelakaan harus difokuskan pada peningkatan disiplin dan kesadaran pengemudi. Strategi tersebut dapat berupa kampanye edukasi, pengetatan pengawasan dan penegakan hukum yang lebih konsisten. Model ini dapat diimplementasikan sebagai sistem pendukung keputusan untuk menganalisis berita kecelakaan secara otomatis dan *real-time*. Sistem ini memungkinkan identifikasi tren dan pola yang dapat menginformasikan kebijakan keselamatan lalu lintas yang lebih efektif.

5.2 Saran

Berdasarkan hasil penelitian dan kesimpulan yang telah diuraikan, berikut adalah beberapa saran untuk pengembangan penelitian selanjutnya:

1. **Eksplorasi Parameter *FastText* yang Lebih Mendalam** Parameter pada model *FastText* perlu dieksplorasi lebih lanjut untuk mendapatkan performa yang optimal. Pengaturan parameter yang tepat seperti ukuran n-gram, dimensi vektor, dan *window size* akan memberikan *embedding* kata yang lebih baik. Selain itu, pengaturan ini harus diarahkan untuk memperoleh *average loss* yang rendah, sehingga model dapat lebih mudah mempelajari data dan memberikan hasil yang lebih akurat. Penelitian mendatang dapat melakukan *grid search* atau hyperparameter tuning yang lebih sistematis untuk menemukan kombinasi parameter *FastText* yang optimal.

2. **Peningkatan Kualitas dan Keseimbangan Dataset** Dataset perlu ditingkatkan dengan mencari dan mengumpulkan lebih banyak data untuk mencapai keseimbangan yang lebih baik antar kelas. Meskipun kelas "Kelalaian Pengemudi" merupakan mayoritas dalam dataset, kelas "Faktor Jalan" dan "Faktor Kendaraan" masih memerlukan penambahan data untuk mencapai distribusi yang lebih seimbang. Upaya pencarian data yang lebih beragam dari berbagai sumber berita dapat membantu mengatasi ketidakseimbangan ini secara natural, sehingga mengurangi ketergantungan pada teknik sintesis data.
3. **Optimasi Arsitektur *Neural Network*** Penelitian selanjutnya dapat mengeksplorasi arsitektur *Neural Network* yang lebih kompleks, seperti *deep Neural Network* dengan *multiple hidden layers* atau implementasi teknik regularisasi seperti *dropout* dan *batch normalization* untuk meningkatkan generalisasi model dan mencegah *overfitting*.

DAFTAR PUSTAKA

- Afandi, Widi Et Al. 2022. "Klasifikasi Judul Berita Clickbait Menggunakan Rnn-Lstm." *Jurnal Informatika: Jurnal Pengembangan It* 7(2): 85–89. <https://doi.org/10.30591/jpit.v7i2.3401>.
- Ahmed, Jeelani, And Muqem Ahmed. 2021. "Online News Classification Using Machine Learning Techniques." *Iium Engineering Journal* 22(2): 210–25. <https://doi.org/10.31436/iiumej.v22i2.1662>.
- Ali, Farman Et Al. 2021. "Traffic Accident Detection And Condition Analysis Based On Social Networking Data." *Accident Analysis And Prevention* 151(September 2020): 105973. <https://doi.org/10.1016/j.aap.2021.105973>.
- Almurieb, Hawraa A, And Eman S Bhaya. 2020. "Softmax Neural Best Approximation." *Iop Conference Series: Materials Science And Engineering* 871(1): 012040. <https://iopscience.iop.org/article/10.1088/1757-899x/871/1/012040>.
- Bengio, Yoshua, Aaron Courville, And Pascal Vincent. 2012. "Representation Learning: A Review And New Perspectives." (1993): 1–30. <http://arxiv.org/abs/1206.5538>.
- Bintari, E D, Gunawan, And A Indriani. 2019. "Classification Of News On 'Radar' Tarakan Online Using K-Nearest Neighbor Method With N-Gram Features." *Iop Conference Series: Materials Science And Engineering* 676(1): 012008. <https://doi.org/10.1088/1757-899x/676/1/012008>.
- Bojanowski, Piotr, Edouard Grave, Armand Joulin, And Tomas Mikolov. 2017. "Enriching Word Vectors With Subword Information." *Transactions Of The Association For Computational Linguistics* 5: 135–46. https://doi.org/10.1162/tacl_a_00051.
- Cahyana, Nur Heri Et Al. 2025. "The Evaluation Of Effects Of Oversampling And Word Embedding On Sentiment Analysis." *Jurnal Infotel* 17(1): 54–67. <https://doi.org/10.20895/infotel.v17i1.1077>.
- Chawla, Nitesh V, Kevin W Bowyer, Lawrence O Hall, And W. P. Kegelmeyer. 2002. "Smote: Synthetic Minority Over-Sampling Technique." *Journal Of Artificial Intelligence Research* 16: 321–57. <https://doi.org/10.1613/jair.953>.
- Dewi, Findra Kartika Sari. 2021. "Klasifikasi Berita Menggunakan Metode Multinomial Naïve Bayes." *Scan - Jurnal Teknologi Informasi Dan Komunikasi* 16(3): 11. <https://doi.org/10.33005/scan.v16i3.2870>.

- Držík, Dávid, And Jozef Kapusta. 2024. "Effect Of Dimension Size And Window Size On Word Embedding In Classification Tasks." : 1–27. <https://doi.org/10.21203/rs.3.rs-4532901/v1>.
- Esti Anggraini, Ratih Nur, Hana Machmudah, And Riyanarto Sarno. 2023. "Hierarchical Topic Mining And Multi-Label Classification On Online News In Bahasa." In *2023 International Conference On Advanced Mechatronics, Intelligent Manufacture And Industrial Automation (Icamimia)*, Ieee, 1–6. <https://doi.org/10.1109/Icamimia60881.2023.10427844>.
- Gelvez-Almeida, Elkin Et Al. 2024. "A Review On Large-Scale Data Processing With Parallel And Distributed Randomized Extreme Learning Machine Neural Networks." *Mathematical And Computational Applications* 29(3). <https://doi.org/10.3390/Mca29030040>.
- Goodfellow, Ian, Yoshua Bengio, And Aaron Courville. 2016. "Deep Learning." 22(4): 351–54. <https://doi.org/10.4258/Hir.2016.22.4.351>.
- Hanifa, Aini, Sugih Ahmad Fauzan, Muhammad Hikah, And Muhammad Bahrul Ashfiya. 2021a. "Perbandingan Metode Lstm Dan Gru (Rnn) Untuk Klasifikasi Berita Palsu Berbahasa Indonesia." *Dinamika Rekayasa* 17(1): 33. <https://doi.org/10.20884/1.Dr.2021.17.1.436>.
- Hasanah, Ukhty Uswatun. 2022. "Klasifikasi Berita Online Menggunakan Metode Artificial Neural Network (Ann)." Universitas Islam Negeri Maulana Malik Ibrahim. <http://etheses.uin-malang.ac.id/id/eprint/42836>.
- Irham, Lalu Gias, Adiwijaya Adiwijaya, And Untari Novia Wisesty. 2019. "Klasifikasi Berita Bahasa Indonesia Menggunakan Mutual Information Dan Support Vector Machine." *Jurnal Media Informatika Budidarma* 3(4): 284. <https://doi.org/10.30865/Mib.V3i4.1410>.
- Jepkoech, Jennifer, David Muchangi Mugo, Benson K Kenduiywo, And Edna Chebet Too. 2021. "The Effect Of Adaptive Learning Rate On The Accuracy Of Neural Networks." *International Journal Of Advanced Computer Science And Applications* 12(8): 736–51. <https://doi.org/10.14569/Ijacs.2021.0120885>.
- Juwita, Rahmah Miya, Elin Haerani, Siska Kurnia Gusti, And Siti Ramadhani. 2022. "Klasifikasi Berita Menggunakan Metode K-Nearest Neighbor." *Jurnal Nasional Komputasi Dan Teknologi Informasi (Jnkti)* 5(2): 259–68. <https://doi.org/10.32672/Jnkti.V5i2.4192>.
- Malaviya, Priyanshu, Hardeep Patel, Priyansh Sanghavi, And Yogesh Kumar. 2024. "Classification Of Online News Articles Using Optimized Deep Learning Models." In *2024 International Conference On Intelligent And Innovative Technologies In Computing, Electrical And Electronics (Iitcee)*,

Ieee, 1–7. <https://doi.org/10.1109/Iitcee59897.2024.10467504>.

Mit, Rizal. 2021. “Klasifikasi Berita Olahraga Pada Portal Berita Online Dengan Metode K-Nearest Neighbour (Knn) Dan Levenshtein Distance.” *Jurnal Teknologi Terapan And Sains* 4.0 2(1): 365. <https://doi.org/10.29103/Tts.V2i1.3760>.

Oostwal, Elisa, Michiel Straat, And Michael Biehl. 2021. “Hidden Unit Specialization In Layered Neural Networks: Relu Vs. Sigmoidal Activation.” *Physica A: Statistical Mechanics And Its Applications* 564: 125517. <https://doi.org/10.1016/J.Physa.2020.125517>.

Prasetya, Fani, And Ferdiansyah Ferdiansyah. 2022. “Analisis Data Mining Klasifikasi Berita Hoax Covid 19 Menggunakan Algoritma Naive Bayes.” *Jurnal Sistem Komputer Dan Informatika (Json)* 4(1): 132. <https://doi.org/10.30865/Json.V4i1.4852>.

Ridwan, Asep Saepul, Yulison H Chrisnanto, And Ridwan Ilyas. 2021. “Klasifikasi Kalimat Pada Berita Olahraga Secara Otomatis Menggunakan Metode Artificial Neural Network.” *Jurnal Komputer Dan Informatika* 9(1): 88–97. <https://doi.org/10.35508/Jicon.V9i1.3708>.

Sathyanarayanan, S. 2024. “Confusion Matrix-Based Performance Evaluation Metrics.” *African Journal Of Biomedical Research* 27(4): 4023–31. <https://doi.org/10.53555/Ajbr.V27i4s.4345>.

Siddique, M. N.H., And M. O. Tokhi. 2001. “Training Neural Networks: Backpropagation Vs Genetic Algorithms.” *Proceedings Of The International Joint Conference On Neural Networks* 4: 2673–78. <https://doi.org/10.1109/Ijcn.2001.938792>.

Syahrudin, Syahrudin, Fatmawati Fatmawati, And Herry Suprajitno. 2022. “The Formula Study In Determining The Best Number Of Neurons In Neural Network Backpropagation Architecture With Three Hidden Layers.” *Jurnal Resti (Rekayasa Sistem Dan Teknologi Informasi)* 6(3): 397–402. <https://doi.org/10.29207/Resti.V6i3.4049>.

Takawane, Gauri, Abhishek Phaltankar, Varad Patwardhan, And Aryan Patil. 2023. “Language Augmentation Approach For Code-Mixed Text Classification.” *Natural Language Processing Journal* 5(May): 100042. <https://doi.org/10.1016/J.Nlp.2023.100042>.

Vilares Ferro, Manuel, Yeraí Doval Mosquera, Francisco J. Ribadas Pena, And Víctor M. Darriba Bilbao. 2023. “Early Stopping By Correlating Online Indicators In Neural Networks.” *Neural Networks* 159: 109–24. <https://doi.org/10.1016/J.Neunet.2022.11.035>.

Widhiyasana, Yudi, Transmissia Semiawan, Ilham Gibran Achmad Mudzakir, And Muhammad Randi Noor. 2021. "Penerapan Convolutional Long Short-Term Memory Untuk Klasifikasi Teks Berita Bahasa Indonesia." *Jurnal Nasional Teknik Elektro Dan Teknologi Informasi* 10(4): 8. [Http://Dx.Doi.Org/10.22146/Jnteti.V10i4.2438](http://dx.doi.org/10.22146/jnteti.v10i4.2438).