



HiLCoE

School of Computer Science
& Technology

Classifying Car Crashes Using Neural Networks

Members Alazar Gebremehdin
Feruz Seid
Hannibal Mussie
Samir Bahru
Yassin Bedru

Course Artificial Intelligence (CS488)

Instructor Dr. Seyoum Abebe

Batch/Section DRB2202/A

Term Autumn 2025

Date Jan 04, 2026

1. Introduction

The objective of this project is to build, train, and evaluate a Neural Network classification model to predict the severity of road traffic accidents based on a dataset from Addis Ababa. The target variable, *Accident Type*, classifies accidents into four categories: **Fatal**, **Serious Injury**, **Minor Injury**, and **Property Damage Only (PDO)**.

This report summarizes the end-to-end workflow, from Exploratory Data Analysis (EDA) and rigorous data cleaning to model architecture design and final performance evaluation.

2. Data Understanding and Preparation

2.1. Exploratory Data Analysis (EDA)

Initial analysis revealed significant data quality challenges. The raw dataset contained over 30 features, but many suffered from high rates of missing values.

- **Missing Data:** Columns such as *Zone*, *Region*, and specific victim details (e.g., *Victim-2 Movement*) had missing rates exceeding 70%.
- **Class Imbalance:** The target variable was heavily skewed. **Minor Injuries** accounted for approximately 63% of the data, while **Fatal** accidents represented only 3%.

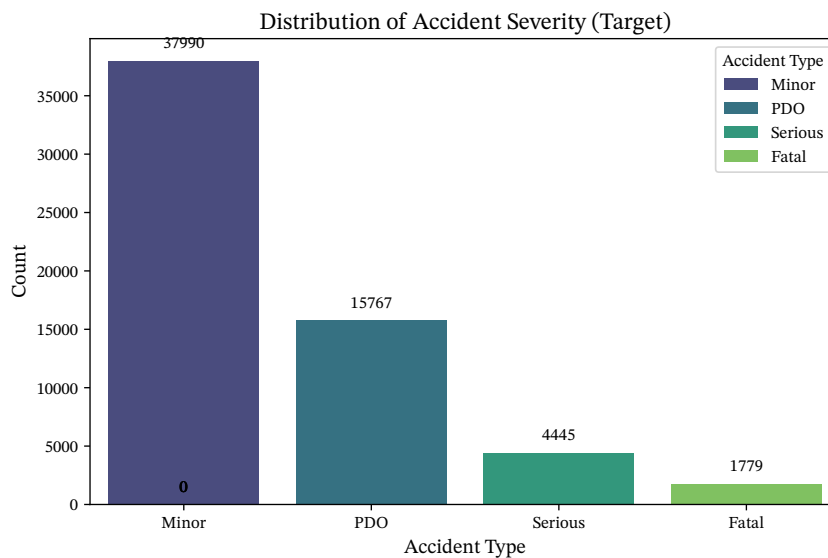


Figure 2: Distribution of Accident Severity (Target Variable). Note the severe imbalance.

2.2. Data Cleaning & Standardization

Before modeling, a custom cleaning pipeline was implemented:

1. **Standardization:** Typos were corrected (e.g., "Augest" → "August", "Privategg" → "Private"). Amharic terms like "Amet" (Year) and "Wor" (Month) were standardized.
2. **Outlier Removal:** Impossible values (e.g., Driver Age > 90, Experience > 60 years) were treated as missing.
3. **Feature Dropping:** Columns with > 70% missing data were removed to reduce noise.

2.3. Feature Engineering & Preprocessing

To prepare the data for the Neural Network:

- **Cyclical Encoding:** The time of day was converted from linear hours (0-23) into sine and cosine components to preserve the temporal proximity between 23:00 and 00:00.
- **Imputation:** Median imputation was used for numerical features (e.g., Age) and Mode imputation for categorical features.
- **Scaling & Encoding:** Numerical features were standardized using *StandardScaler*. Categorical variables were transformed using *OneHotEncoding*.

3. Model Design

3.1. Architecture

A Multi-Layer Perceptron (MLP) was designed using TensorFlow/Keras. The architecture features a “funnel” design to compress high-dimensional inputs into abstract representations.

- **Input Layer:** 230 features (resulting from One-Hot Encoding of high-cardinality categorical variables).
- **Hidden Layer 1:** 128 Neurons, ReLU activation, Batch Normalization (Param count: 29.5k).
- **Hidden Layer 2:** 64 Neurons, ReLU activation, Batch Normalization (Param count: 8k).
- **Output Layer:** 5 Neurons with Softmax activation.

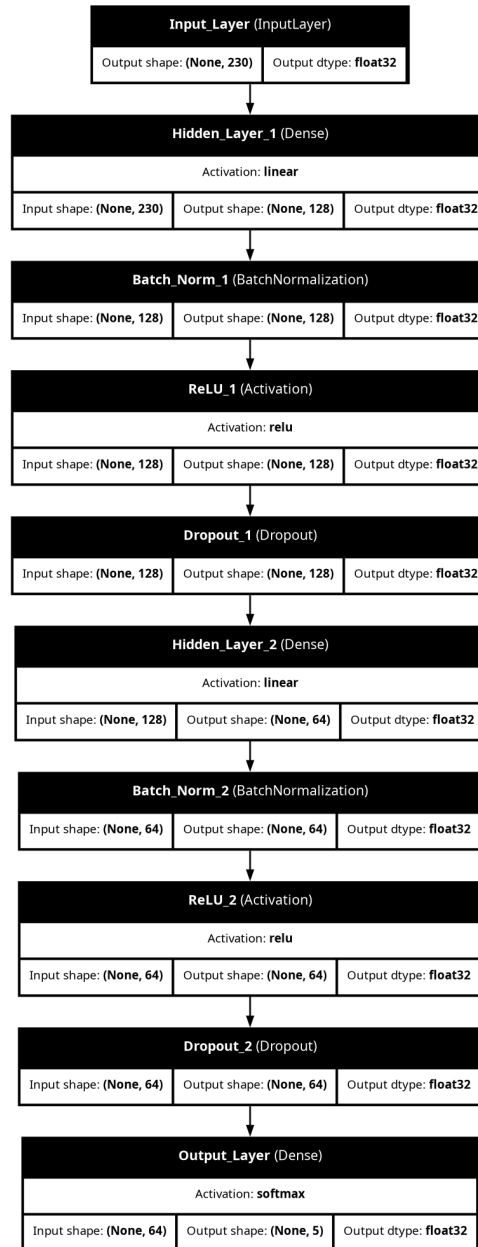


Figure 3: Neural Network Architecture Diagram. Total trainable parameters: 38,533.

3.2. Design Justification

- **ReLU Activation:** Selected to prevent the vanishing gradient problem.
- **Dropout (0.3 - 0.4):** Applied after hidden layers. Given the high dimensionality (230 features), dropout was crucial to prevent the model from memorizing specific input patterns.
- **L2 Regularization:** Added to penalize large weights, keeping the model weights small and stable.

4. Training Process

The model was trained for up to 100 epochs using the **Adam** optimizer. To address the class imbalance identified in EDA, **Class Weights** were computed and applied to the Loss Function. This penalizes the model more heavily for misclassifying rare classes (Fatal/Serious) than common ones (Minor).

Overfitting Mitigation:

- **Early Stopping:** Training halted automatically when Validation Loss failed to improve for 15 epochs.
- **Model Checkpointing:** Only the model version with the highest Validation Accuracy was saved.

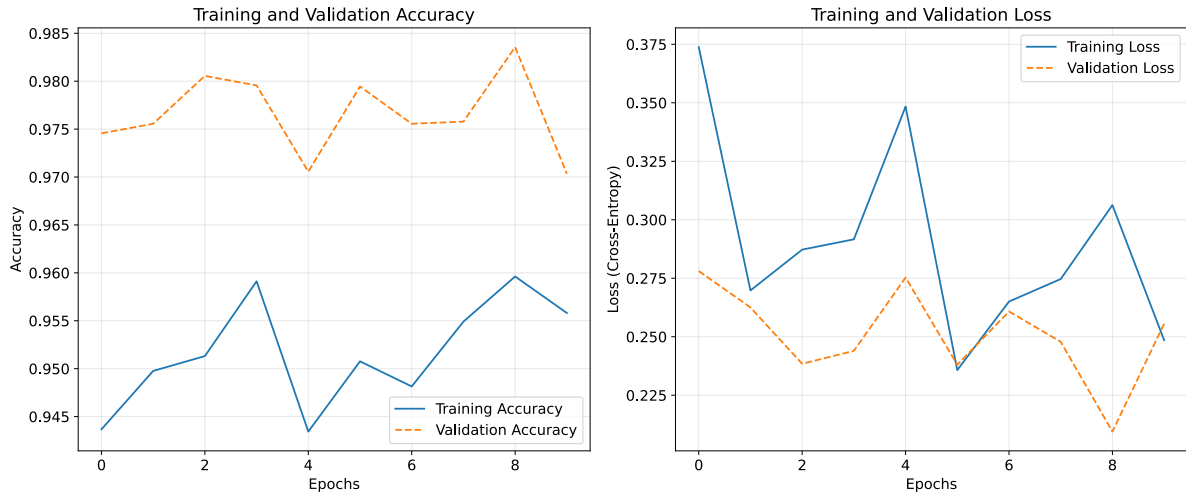


Figure 4: Training and Validation Performance Curves. Note the convergence point where Early Stopping triggers.

5. Evaluation and Interpretation

5.1. Performance Metrics

The model was evaluated on an unseen Test Set. The performance was exceptional, achieving an overall accuracy of approximately **98%**.

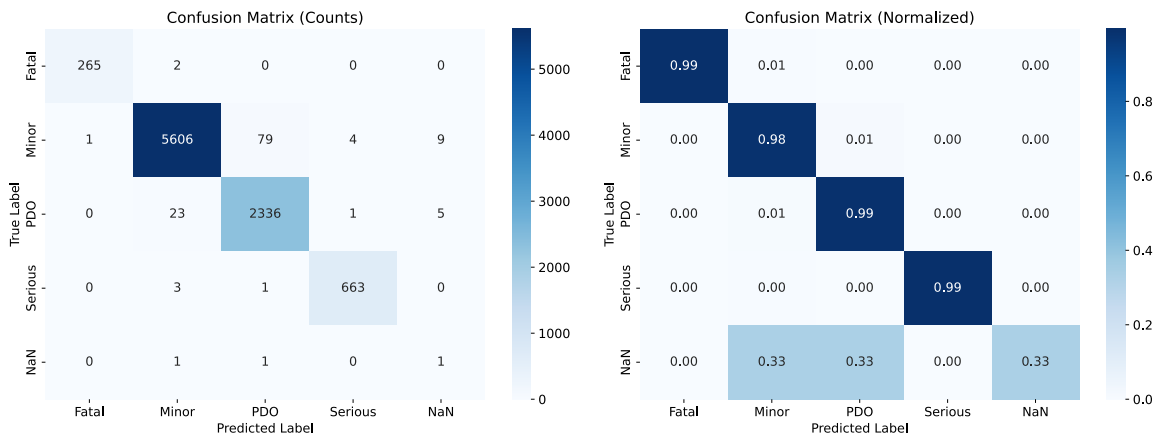


Figure 5: Confusion Matrix. Note the near-perfect diagonal, indicating high classification accuracy across all classes.

5.2. Analysis of Results

1. Strengths:

- **High Recall on Fatal Cases:** The model correctly identified 265 out of 267 fatal accidents (99% Recall). This is a significant achievement, as “Fatal” is usually the hardest class to predict due to its rarity.
- **Robustness:** The validation loss remained stable alongside training loss, indicating that the regularization techniques effectively prevented overfitting despite the model’s high capacity.

2. Critical Reflection (Data Insights):

- Upon analyzing the feature importance, the high accuracy suggests the model utilized strong predictors present in the dataset, specifically the **post-accident** casualty counts (e.g., Number of fatalities, Number of severe injuries).
- Since the target variable (Accident Type) is directly derived from these counts, the model effectively learned the rule-based definitions of the categories (e.g., if Fatalities > 0, then Type = Fatal).
- While this makes the model a highly effective **classifier** for historical records, it relies on data that would not be available before an accident occurs.

6. Recommendations & Conclusion

6.1. Potential Improvements

1. **Predictive Modeling:** To build a model that predicts severity **before** an accident happens (based purely on road conditions, weather, and driver demographics), a second iteration of this project should explicitly exclude the Number of fatalities/injuries columns from the input.
2. **Advanced Architectures:** For the predictive (non-leakage) version, implementing **TabNet** or **Wide & Deep** networks would be necessary to capture complex interactions between environmental factors without relying on casualty counts.

6.2. Conclusion

The project successfully demonstrated the end-to-end Neural Network workflow. The data preparation phase successfully transformed a messy dataset into a clean, 230-feature input space. The resulting model achieved 98% accuracy in classifying accident severity, proving that the Neural Network can effectively map input features to accident outcomes when casualty data is available.

6.3. Reflection on Workflow

The project demonstrated that **Data Preparation** is the most critical phase. The raw data required extensive cleaning before any modeling could succeed. The transition from a raw, noisy Excel sheet to a structured Neural Network pipeline highlights the importance of robust feature engineering (like cyclical time encoding) and rigorous evaluation beyond simple accuracy.