Classifying Car Crashes Using Neural Networks

From Raw Data to Severity Prediction

Alazar Gebremehdin, Hannibal Mussie, Feruz Seid, Yassin Bedru, Samir Bahru

2026-01-04

Outline

Introduction

Objective & Overview

Goal: Predict the severity of road traffic accidents (**Fatal, Serious, Minor, PDO**) based on accident characteristics.

The Workflow:

1. **Data Understanding:** Handling massive missing data and inconsistencies.
2. **Preparation:** Cleaning, Imputation, and Feature Engineering.
3. **Modeling:** Designing a Multi-Layer Perceptron (MLP).
4. **Training:** Managing class imbalance and overfitting.
5. **Evaluation:** F1-Scores and Confusion Matrices.

Data Understanding (EDA)

Data Quality Challenges

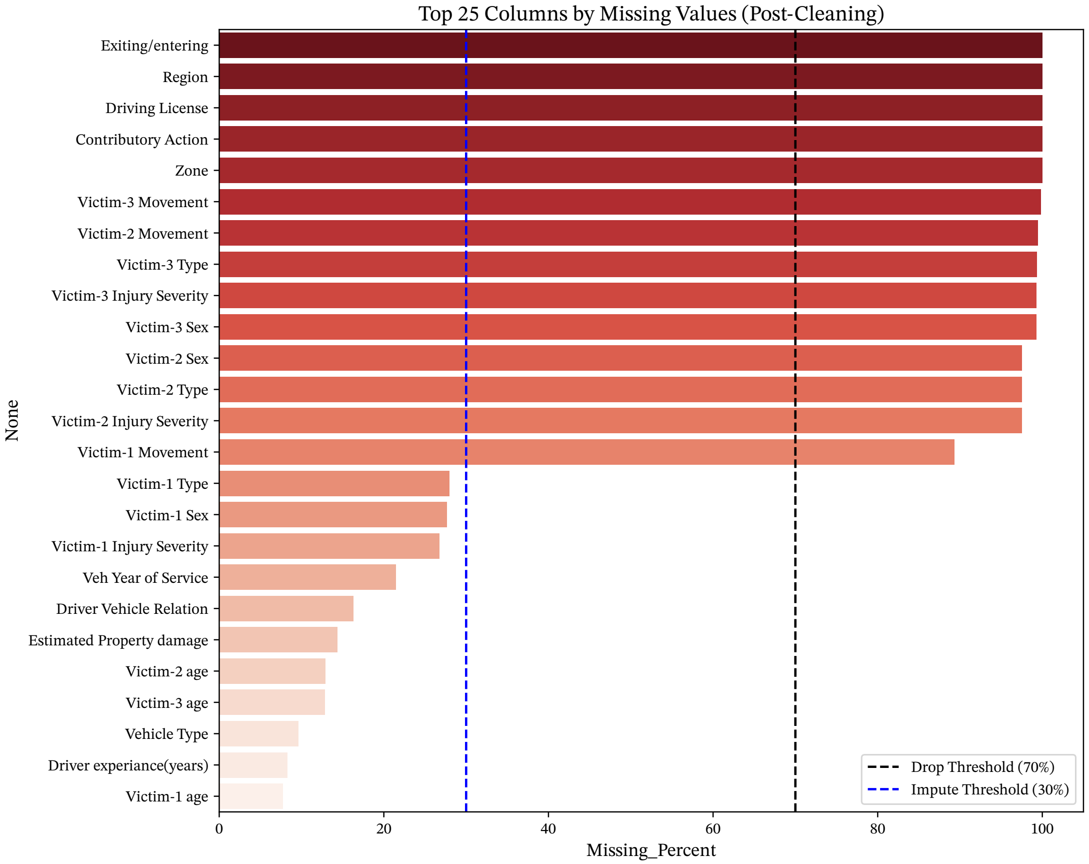
Initial Inspection:

- The raw dataset contained over 30 columns but suffered from severe quality issues.
- **Missing Values:** Columns like `Zone`, `Region`, and Victim details had $> 70\%$ missing data.
- **Inconsistencies:** Typos (e.g., “Augest”, “Privategg”) and impossible values (Age > 90).

Action:

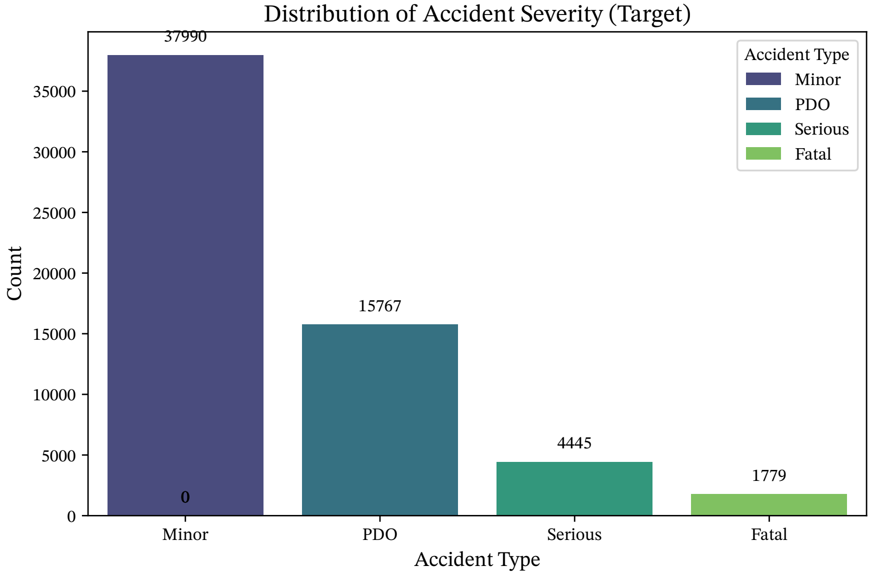
- Dropped columns with $> 70\%$ missingness.
- Standardized categorical labels (e.g., mapping `P.D.O`, `pdo` $\rightarrow$ `PDO`).

Missing Values Analysis



Class Imbalance

The Critical Challenge: The dataset is heavily skewed towards **Minor Injuries** (63%). **Fatal** accidents represent only 3%.



Data Preparation

Feature Engineering: Time

Problem: Time is cyclical. 23:00 is close to 00:00, but numerically (23 vs 0) they are far apart.

Solution: We encoded time using Sine and Cosine transformations.

```python
# Feature Engineering Code Snippet
def feature_engineering(df):
    # Extract Hour
    df['Hour'] = df['Time'].apply(extract_hour)

    # Cyclical Encoding
    df['Hour_Sin'] = np.sin(2 * np.pi * df['Hour'] / 24)
    df['Hour_Cos'] = np.cos(2 * np.pi * df['Hour'] / 24)
```

Feature Engineering: Time


```

    return df.drop(columns=['Time'])
```

Preprocessing Pipeline

Before feeding data into the Neural Network:

1. **Imputation:**
   - Numerical (Age, Experience) → **Median**
   - Categorical (Road Surface, Light) → **Mode**
2. **Scaling:**
   - `StandardScaler` applied to numerical inputs to normalize variance.
3. **Encoding:**
   - `OneHotEncoder` for categorical variables.
4. **Splitting:**
   - Train (70%) / Validation (15%) / Test (15%).

Model Design

Architecture Summary

Based on the execution results:

- **Input Shape:** 230 Features (High dimensionality due to One-Hot Encoding).
- **Total Parameters:** 38,917 (Lightweight model).
- **Trainable Params:** 38,533.

Layer Structure:

- Input (230)
- Dense (128) → BN → ReLU → Dropout
- Dense (64) → BN → ReLU → Dropout
- Output (5 Classes)

Implementation Code

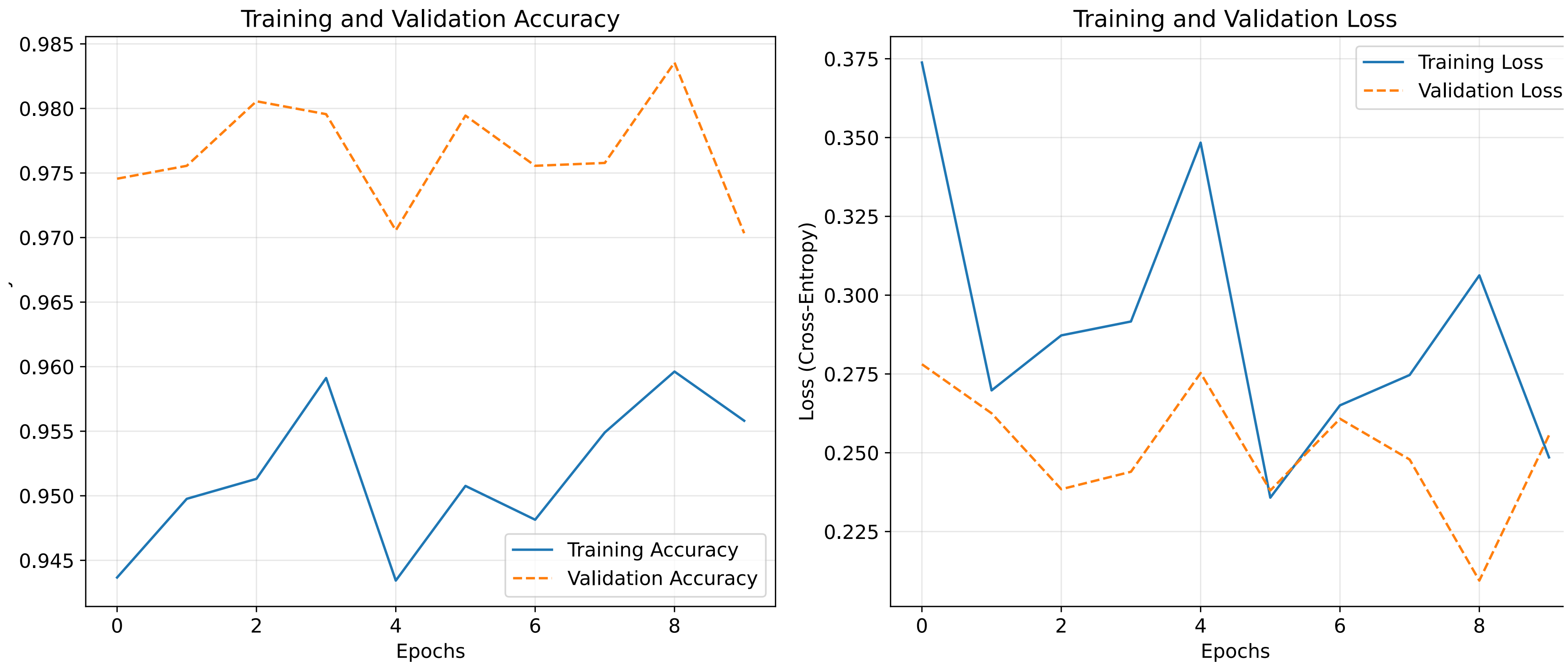
Python

```
# Actual Model Summary Output
Layer (type)                 Output Shape          Param #
=================================================================
Input_Layer (InputLayer)     (None, 230)           0
Hidden_Layer_1 (Dense)       (None, 128)           29,568
Batch_Norm_1                (None, 128)           512
Dropout_1 (Dropout)          (None, 128)           0
Hidden_Layer_2 (Dense)       (None, 64)            8,256
Output_Layer (Dense)         (None, 5)             325
=================================================================
Total params: 38,917
```


Input_Layer (InputLayer)
Output shape: (None, 230)
Output dtype: float32
Hidden_Layer_1 (Dense)
Activation: linear
Input shape: (None, 230)
Output shape: (None, 128)
Output dtype: float32
Batch_Norm_1 (BatchNormalization)
Input shape: (None, 128)
Output shape: (None, 128)
Output dtype: float32
ReLU_1 (Activation)
Activation: relu
Input shape: (None, 128)
Output shape: (None, 128)
Output dtype: float32
Dropout_1 (Dropout)
Input shape: (None, 128)
Output shape: (None, 128)
Output dtype: float32
Hidden_Layer_2 (Dense)
Activation: linear
Input shape: (None, 128)
Output shape: (None, 64)
Output dtype: float32
Batch_Norm_2 (BatchNormalization)
Input shape: (None, 64)
Output shape: (None, 64)
Output dtype: float32
ReLU_2 (Activation)
Activation: relu
Input shape: (None, 64)
Output shape: (None, 64)
Output dtype: float32
Dropout_2 (Dropout)
Input shape: (None, 64)
Output shape: (None, 64)
Output dtype: float32
Output_Layer (Dense)
Activation: softmax
Input shape: (None, 64)
Output shape: (None, 5)
Output dtype: float32

Training & Evaluation

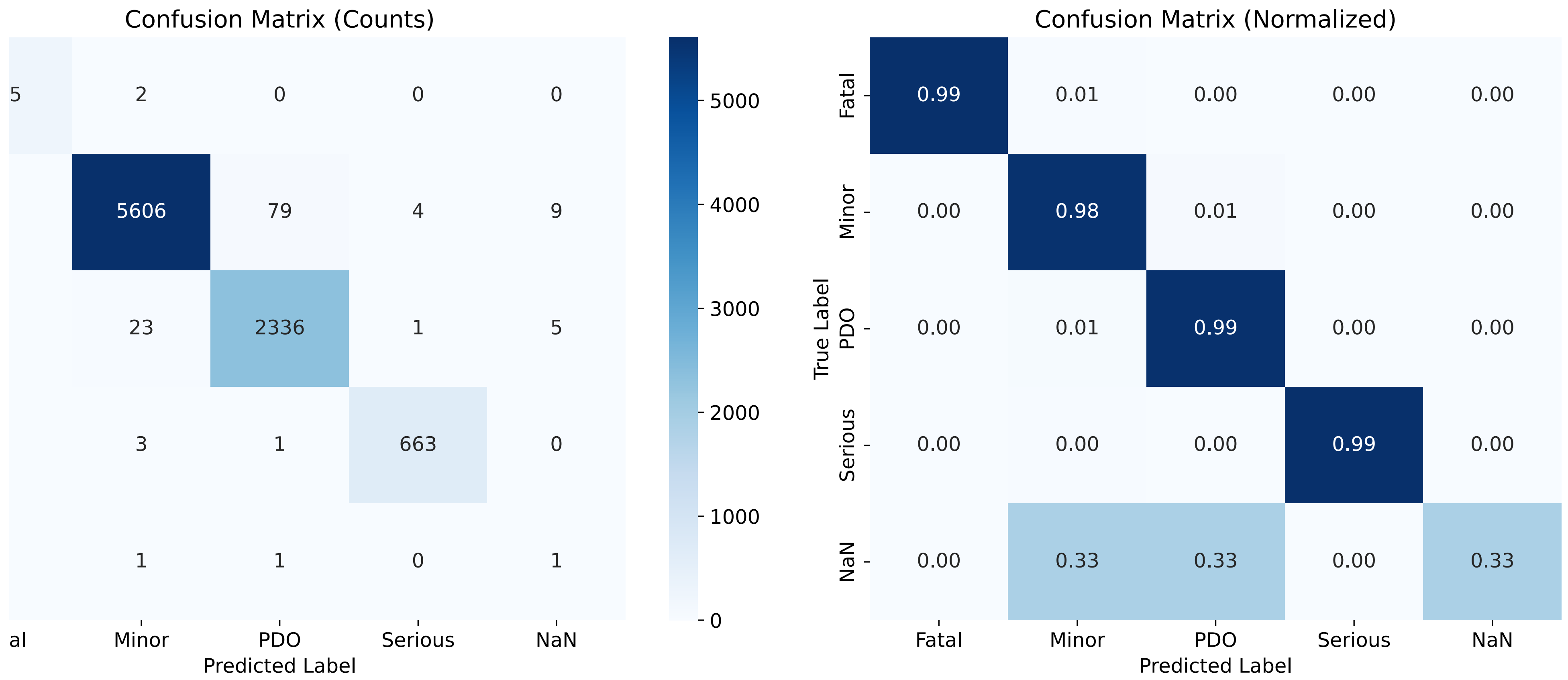
Training Performance



Training Performance

Insight: The model converges quickly. Validation accuracy peaks around **98%**, indicating robust learning without significant overfitting.

Confusion Matrix Results



Analysis of Results

Exceptional Performance:

- **Fatal Class:** 99% Recall (265 Correct, 2 Missed).
- **Minor Class:** 98% Recall.

Critical Analysis (The "Why"):

- The high accuracy suggests the model effectively utilized casualty count features (e.g., `Number of fatalities`) present in the dataset.
- While excellent for **classifying** historical records, this indicates that accident outcomes (casualties) are the strongest predictors of the severity label.

Conclusion

Summary

1. **Data Quality:** Cleaning and encoding resulted in 230 clean input features.
2. **Model:** A 38k parameter MLP was sufficient to capture the relationships.
3. **Results:** The model achieved 98% test accuracy.

Recommendation:

- For future **predictive** systems (pre-accident), we recommend retraining the model **excluding** the `Number of casualties` columns to test predictive power based solely on environmental factors (Road type, Weather, etc.).

Summary

Thank You!

Questions?