Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

UNIVERSITY OF PADUA

INFORMATION ENGINEERING DEPARTMENT (DEI)

MASTER'S DEGREE IN COMPUTER ENGINEERING

Comparative Analysis Between ML Models and GCNN For Accident Severity Classification In London Intersections

Contents

1 Abstract **2**

2 Introduction **2**

3 Graph and dataset **2**

4 Methods **4**

4.1 Support Vector Machine 5

4.2 Random Forest 5

4.3 K Nearest Neighbors 5

4.4 Graph Convolutional Neural Network 5

5 Results **6**

6 Conclusions **7**

7 Observations **7**

8 References **8**

1 Abstract

Accidents have become a significant global concern, leading to increased mortality rates. This project aims to develop a system that can estimate the severity of accidents near the intersections in London. The research focuses on evaluating and comparing the effectiveness of conventional machine learning techniques versus graph neural networks with the goal of identifying the most suitable model for the problem. The graph G = (V, E) derived from OpenStreetMap represents nodes as road intersections (V) and edges as roads (E). The features include node information (longitude, latitude, highway type, street count, degree centrality, name) and edge information (highway type, length, oneway, maxspeed, bridge, junction, tunnel, lanes). The dataset combines features from the graph and UK accident data, incorporating calculated features. Instances in the dataset (v_j) include graph features (latitude, longitude, street count, degree centrality) and external features from the UK dataset (average speed limits, average number of casualties, average accident severity). The severity prediction task is transformed into a classification problem, with severity categories labeled as low risk, medium risk, and high risk. To efficiently construct the dataset, a Ball Tree data structure is utilized for proximity searches, providing statistics for accidents nearest to each intersection. Four models are employed for severity prediction: Support Vector Machine (SVM), Random Forest, K Nearest Neighbors (K-NN), and Graph Convolutional Neural Network (GCN). The models are trained, validated, and tested on a split dataset. Performance evaluation metrics include F1 score, accuracy, precision, and recall. Results indicate that SVM, Random Forest and K-NN models outperform GCN. The unexpected performance of GCN may be attributed to imbalanced class distribution in the dataset.
You can find the code for this project and instructions on how to run it at the following link: `https://github.com/matteocarpentieri/LFN-Project`

2 Introduction

Accidents are becoming more frequent nowadays, emerging as one of the primary contributors to global mortality [6]. The objective of this project is to develop a system able of estimating the severity of accidents in proximity to intersections in London. This work can help institutions to understand which areas to strengthen to reduce this phenomenon. This paper aims to assess and compare the effectiveness of conventional machine learning techniques in contrast to Graph Neural Networks (GNN) [5] with the goal of indentifying the model that best fits the problem. The comparison will be done by using a dataset combining the United Kingdom road accident dataset and road data retrieved from OpenStreetView (OSM) [3] as well as different train-validation dataset-split.

3 Graph and dataset

Node features (top) and edge features (bottom)

Graph feature	Description
longitude	The node longitude.
latitude	The node latitude.
highway	Node type related to road.
street count	the number of roads that intersect the node.
degree centrality	centrality measure that counts how many neighbors a node has
name	The name of the road.
highway	The type of a road (tertiary, motorway, etc.)
length	The length of a road.
oneway	Indicates whether a road is a one-way street.
maxspeed	The maximum legal speed limit of a road.
bridge	Indicates whether a road represents a bridge.
junction	Describes the junction type of a road.
tunnel	Indicates whether a road runs in a tunnel.
lanes	The number of lanes of a road.

For the analysis, we consider

1. The graph $G = (V, E)$ obtained by using OpenStreetMap where:

   - V is the set of nodes that represent the crosses among the roads.
   - E is the collection of edges representing the roads.

   that provides the features in the table above.

2. The data on accidents in London taken from the UK road accident dataset that records over 1.8 million accidents from 2000 and 2018 [2], covering 33 features for each accident. The most relevant and combinable features were selected from the London graph data. Many features were also not considered given the sparse presence within the dataset.
   Those we considered are reported in the table below:

Dataset feature	**Description**
longitude	Longitude of the accident.
latitude	Latitude of the accident.
accident severity	Accident Severity on the Scale of 1 to 3
Speed limit	Speed limit of the road
number of casualties	Number of casualties in accident

Based on the data provided by the graph and UK dataset, we constructed our dataset D combining features from both datasets and some calculated by us. Each instance $v_j \in D$ is a vector containing:

- **Graph features**: we only select the most useful features related to the node $u_j \in V$ which are: the `latitude`, the `longitude`, the `street count` and the `degree centrality` computed by us.

  Note that `degree centrality` $= \frac{\text{degree}(v)}{|V|-1}$, with $|V|$ = total number of nodes in the graph.

- Three **external features** using the UK dataset as follow:

  1) `Average speed limits`: we compute this quantity considering the average over speed limit of the street related to the accidents nearest to the node u_j.
  2) `Average number of casualties`: we compute this quantity considering the average over the number of victims related to the accidents nearest to the node u_j.
  3) `Average accident severity`: we compute this quantity considering the average over the severity related to the accidents nearest to the node

So, the final representation of the vector v_j is:

v_j ={longitude, latitude, street count, degree centrality, number of nearest accidents, average number of casualties, average speed limits, average accident severity}

Looking at the table above, one of the features of an accident is the severity measure. To perform the severity prediction task over each vector v_j in our dataset, we need to have a value of severity measure l_j representing our true label. We compute this quantity considering the average accident severity over the accident severities related to the accidents nearest to the node u_j. By checking the values of accident severities in the UK dataset we discovered that they were only from 1 to 3. Consequently, the average severity values obtained for each cross were in the range [1,3]. As the true labels are numerical values and we aim to predict the severity category rather than a continuous numerical value, we transformed a regression problem into a classification one. We utilized a `label encoder` module from sklearn, which converts labels (categories) into integer values. This way, our true label `average_accident_severity` is transformed into a categorical variable. An important thing to consider was that for the crosses that had no nearest accidents, the average severity values, average number of casualties, and average speed limit were set to 0. The categorical label assigned to these crosses was 0, so the average severity could be 0. Let's consider $s_i \in [0, 3]$ the average severity value:

- if $s_i \in [0, 1]$ is mapped in $l_1 = 0$ that corresponds to the label `low` risk.
- if $s_i \in]1, 2]$ is mapped in $l_2 = 1$ that corresponds to the label `medium` risk.
- if $s_i \in]2, 3]$ is mapped in $l_3 = 2$ that corresponds to the label `high` risk.

So our set of labels representing the severity measure is defined as $L = \{l_1, l_2, l_3\}$.
To create our dataset, it was necessary to find the nearest intersection for each incident, a time-consuming operation given the amount of nodes.
To solve this problem we first tried to use 2 approaches:

- The first one was about the use of the shortest path as it was established in the proposal report. Since the accidents were not mapped into the graph, it was needed to find the nearest nodes. Then compute the shortest paths between the nearest nodes and the crosses considered. If the shortest path were inside a certain threshold the accident was considered nearest to the cross otherwise no.
- The second one was to find the nearest accidents around the crosses by using the haversine distance which measures the distance between two points on the surface of the earth based on their longitude and latitude.

Both solutions were discarded due to the high complexity and the time required to be executed. To perform this operation efficiently, we used the Ball Tree data structure [1], which is suitable for proximity searches. The Ball Tree is then used to find the nearest nodes in the road graph for each traffic accident and then aggregate statistics based on these nodes are calculated to enrich the new dataset such as: average number of casualties, average speed limits and average accident severity as explained above

4 Methods

Let us formalize the accident severity prediction problem.
It is a supervised learning problem given that:

- a feature vector $v_j \in D$ representing a specific node (cross) in the graph where $j \in [1, N]$ where N is the cardinality of V.
- a fixed set of labels $L = \{0, 1, 2\}$ where each label corresponds to a specific accident severity category (low, medium, high).

The goal is to train a model with a training set $T = \{(v_1, l_i), \ldots, (v_n, l_i)\}$ where n is the number of labeled instances in the set T and $i \in [1, 3]$.
All the models were trained taking into account the following division of the dataset:

- 70 % Training set
- 20 % Validation set
- 10 % Test set

Performance evaluation of individual models is based using the metrics F1 score, accuracy, precision and recall
The methods used for development and analysis related to our problem are:

- Support Vector Machine SVM
- Random Forest
- K Nearest Neighbors K-NN
- Graph Convolutional Neural Network

4.1 Support Vector Machine

The SVM classifier involves the use of an RBF (Radial Basis Function) kernel and an adjustment of hyperparameters using grid search. The grid of hyperparameters includes variations of the regularization parameter C and the gamma parameter for the RBF kernel. The SVM model was trained on the training data with different combinations of hyperparameters, and the best performing model was selected based on performance by cross-validation. Experiments have also been performed with other kernels: sigmoid or polynomial but with inferior results compared with RBF.

4.2 Random Forest

The Random Forest classifier needs a hyperparameters tuning using grid search. The hyperparameters are: number of estimators, maximum depth, minimum samples split, minimum samples leaf, maximum features.

4.3 K Nearest Neighbors

The k-nearest neighbors was developed with three different values of k: 3, 5, 7. It does not use hyperparameter tuning using grid search. We select the model that obtained the best accuracy value on the validation set.

4.4 Graph Convolutional Neural Network

For our problem instead of using a classical graph neural network, we chose a graph convolutional neural network
The GCN model that we create, consists of two convolutional graph layers.
The first layer takes an input feature and transforms it into a hidden representation of fixed size and the second layer further transforms the hidden representation into an output corresponding to the number of unique classes, which in our case is 3 (0,1,2).
This architectural design allows the model to learn complex representations from the input data through the application of graphical convolutions, culminating in a final prediction distributed over three distinctive classes.
The model introduces non-linearity using the rectified linear unit activation function (ReLU) and uses the dropout regularization technique to avoid overfitting by randomly "dropping out" a fraction of neurons during the training phase.
Optimization is performed using the Adam optimizer with a learning rate of 0.01 and a decay of weights of $5 \cdot e^{-4}$.
The loss function used is cross-entropy loss suitable for multiclass problems.

In order to pass the data to the model, it was necessary to convert the traning set, validation set and test set into three different PyTorch Geometric objects [4]. Before doing this, the dataset D was transformed into a graph G' in which the nodes had the same features considered for the vectors of D, and the edges among the nodes were the same as in G. Therefore, G' maintained the same structure as G. From this graph, we built 3 subgraphs by applying the function `train_test_split` to the list of nodes in G' . In this way, we obtained 3 different lists of nodes: one for training, one for validation, and one for testing. From these lists, we retrieved the corresponding subgraphs.
Two training versions of GCN were also created with the goal of identifying the best approach:

- In the first version did not split the training nodes into batches.
- In the second version, the nodes of the training subgraph were divided into bacths of 133 samples

5 Results

Table with data on the metrics used for the evaluation of each model on the test set

Table 1: Evaluation

Model	Accuracy	Precision	Recall	F1 Score
SVM	0.9797	0.9616	0.9797	0.9704
Random Forest	0.9797	0.9616	0.9797	0.9704
K-NN_k_3	0.9763	0.9636	0.9763	0.9693
K-NN_k_5	0.9791	0.9672	0.9791	0.9708
K-NN_k_7	0.9796	0.9681	0.9796	0.9707
GCN	0.9469	0.9072	0.6189	0.7231
GCN_with_batch	0.9469	0.9069	0.6189	0.7232

We considered the following hyperparameters for the classifiers in which the Grid Search was applied:

- `SVM`
  - C: 0.1
  - gamma: 0.1
  - kernel: rbf
- `Random Forest`:
  - max_depth: 10
  - max_features: sqrt
  - min_samples_leaf: 2
  - min_samples_split: 2
  - n_estimators: 100

The following image displays, for each accident in the UK_Accident dataset, the severity of the accident categorized according to the following criteria::

- green = low
- yellow = medium
- red = high

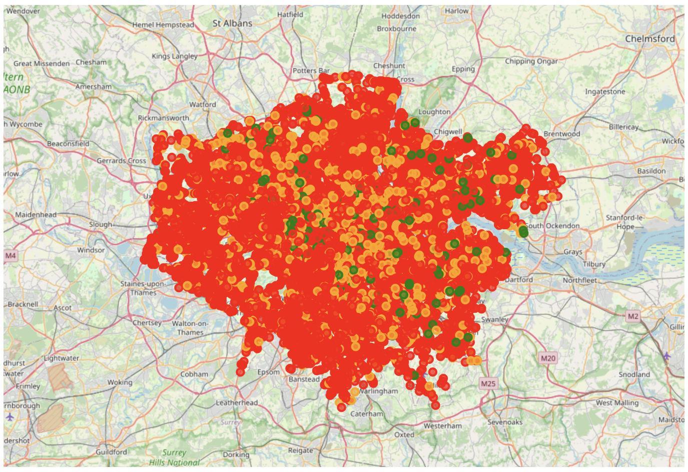

Figure 1: Traffic accident severities in London

To perform the analysis we use different machines:

- HP Pavilion Laptop: AMD Ryzen 7 5825U , 16GB RAM, AMD Radeon (TM) Graphics, Windows 11
- Macbook Pro 2020: 2 GHz Intel Core i5 quad-core, 16GB, Intel Iris Plus Graphics 1536 MB, macOS
- Macbook Air 2022: Apple M2, 8GB RAM, no GPU, macOS
- Google Colab

6 Conclusions

The model that proved to fit our problem best turned out to be KNN.
Unexpectedly, GCN is not the model that performed best despite being the structure of the problem suitable for such a network.
One hypothesis that might explain why is that when a graph is used for classification tasks as in our case, it is preferable that the data available for training belong to different classes so that the model can best learn the differences between them. In our case the number of available classes is small and with a large majority of the nodes belonging to one particular class that is $l_3 = 2$ (as reported in figure 1).
As much as gnn has been shown to be less accurate than traditional machine learning models, we found that for the same amount of time, with the exception of KNN, it is the model that takes the least amount of time in the various training phases. Therefore, with a larger amount of data that would further increase the execution time , its use could be valid and interesting

7 Observations

We summarize some observations:

- At the website where we downloaded the UK_Accident dataset, it was specified that the feature `Accident_Severity` was on a scale from 1 to 5. However, upon verification (by identifying the minimum and maximum values of that feature), we found that the scale actually ranged from 1 to 3.

Additionally, two changes have been made compared to the midterm/proposal:

- We opted for the BallTree algorithm instead of the shortest path to assign the nearest incident to each node for computational efficiency (significantly faster).
- We introduced some additional features (average speed limits and degree centrality) to enhance the performance of various models.
- We changed the title from ”Integrating GNN And Graph Analysis For Estimating Accident Severity At Road Intersections” to ”Comparative Analysis Between ML Models and GCNN For Accident Severity Classification In London Intersections” because it seemed more appropriate to us.

8 References

[1] Ball Tree and KD Tree Algorithms. https://www.geeksforgeeks.org/ball-tree-and-kd-tree-algorithms/.

[2] Road accident united kingdom (uk) dataset. https://www.kaggle.com/datasets/devansodariya/road-accident-united-kingdom-uk-dataset.

[3] Geoff Boeing. Osmnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks. *Computers, Environment and Urban Systems*, 65:126–139, 2017.

[4] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. 2019.

[5] William L. Hamilton. *Graph Representation Learning*. Morgan & Claypool Publishers, San Rafael, 2020.

[6] World Health Organization. Road Traffic Injuries. https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries.