ISSN 2007-9737

Classifying Roads with Multi-Step Graph Embeddings

Mohale E. Molefe*, Jules R. Tapamo

University of KwaZulu Natal,
School of Engineering,
South Africa

molefemohale@gmail.com, tapamoj@ukzn.ac.za

Abstract. Machine learning-based road-type classification is pivotal in intelligent road network systems, where accurate network modelling is crucial. Graph embedding methods have emerged as the leading paradigm for capturing the intricate relationships within road networks. However, their effectiveness hinges on the quality of input features. This paper introduces a novel two-stage graph embedding approach used to classify road-type. The first stage employs Deep Autoencoders to produce compact representation of road segments. This compactified representation is then used, in the second stage, by graph embedding methods to generate an embedded vectors, leveraging the features of neighbouring segments. Results achieved, with experiments on realistic city road network datasets, show that the proposed method outperforms existing approaches with respect to classification accuracy.

Keywords. Road type classification, road networks intelligent systems, graph embedding methods, deep autoencoder.

1 Introduction

Cities worldwide face growing traffic issues, such as congestion, accidents, and rising fuel costs. These problems are caused by increased population, vehicles in traffics, and the overall number of people using the roads.

Designing and developing smart cities is essential for better managing and reducing these traffic problems [11]. Smart city initiatives leverage information and communication infrastructure (ICT) to optimize urban living by tackling historical urban challenges through data-driven solutions and interconnected systems.

Urban transportation systems within smart cities encompass diverse applications, aiming to optimize traffic flow and minimize congestion by designing intelligent systems that rely on data captured by sensors strategically placed throughout road infrastructure. This data can be leveraged to build and train machine learning models for various transportation applications, including real-time arrival estimations and prediction of traffic flows.

Notably, the potential of machine learning for the design of intelligent transport systems extends beyond these well-established applications, with one promising yet underexplored area being the automated classification of road types. Integrating models to classify road-type within interactive maps offers valuable traffic information to users, enabling them to avoid congested routes, accident-prone areas, and intersections with high frequency.

However, leveraging machine learning for road-type classification on road network graphs presents a challenge because of the scarcity of established hand crafted based methodologies for generating feature vectors from road segments. To address this, recent research has explored graph embedding techniques, that use deep learning models to capture spatial relationships between road segments.

Features are automatically extracted within the graph network structure. Feature vectors are generated with graph embedding using their neighbouring road segments. This study represents an extension of the research initially presented at the MCPR conference [9], introducing a new multi-stage graph embedding approach

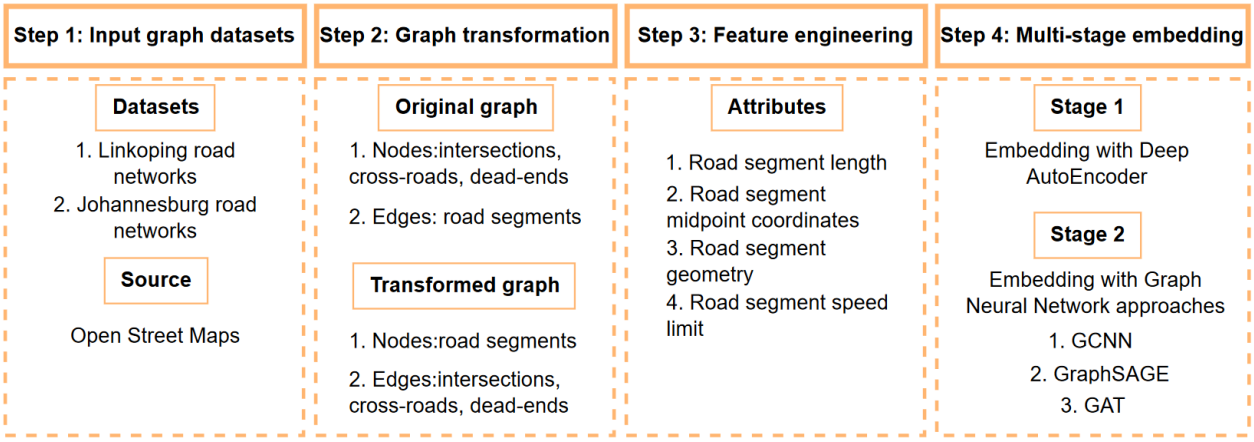

Fig. 1. System diagram of the proposed method

for classifying road types in real-world urban environments [10]. The original research addressed two key challenges associated with graph embedding methodologies for road networks modelling: a) the inability of graph embedding methods to conduct embedding raw road segments' feature vectors of , and b) the frequent assumption within graph embedding methods that road segment features are consistently robust and accurate.

To overcome these challenges, the initial stage of the original research utilized a Deep AutoEncoder (DAE) model to embed the raw road segments' feature vectors into significantly smaller dimensions while preserving essential features.

The resulting features from this first stage were then utilized as input for the second stage, where Graph Convolutional Neural Networks (GCNN) were employed to obtain the embedded features of a given road segment based on the features of its neighbouring road segments. The classification of road types was then accomplished using a MultiLayer Perceptron (MLP) classifier.

Building upon the original work's multi-stage graph embedding method, this study conducts a comparative analysis of various graph embedding techniques for road network modelling. The proposed approach is evaluated across multiple diverse road network datasets to ensure generalizability.

The rest of paper is organised as follows: Section 2 reviews recent advances in road-type classification, highlighting their techniques, models, and results achieved. Section 3 discusses the technical details of the proposed multi-stage graph embedding approach, outlining its components and implementation.

Section 4 presents the experimental setup, evaluation metrics, and obtained results, offering a comparative analysis with existing methods. Finally, Section 5 summarizes the essential findings of the study and outlines potential directions for future research.

2 Background and Related Work

Researchers can effectively model road networks using graph theory. This approach captures the complete topological structure of any road network, regardless of its size or complexity. Notably, graphs can represent not only spatial road networks but also diverse transportation systems, including highways, public transit networks, air routes, and waterways.

Furthermore, graph-based models can readily incorporate various network attributes such as speed limits, travel times, lane numbers, and traffic flow patterns. Essentially, graphs depict the topological structure of a network through nodes and edges.

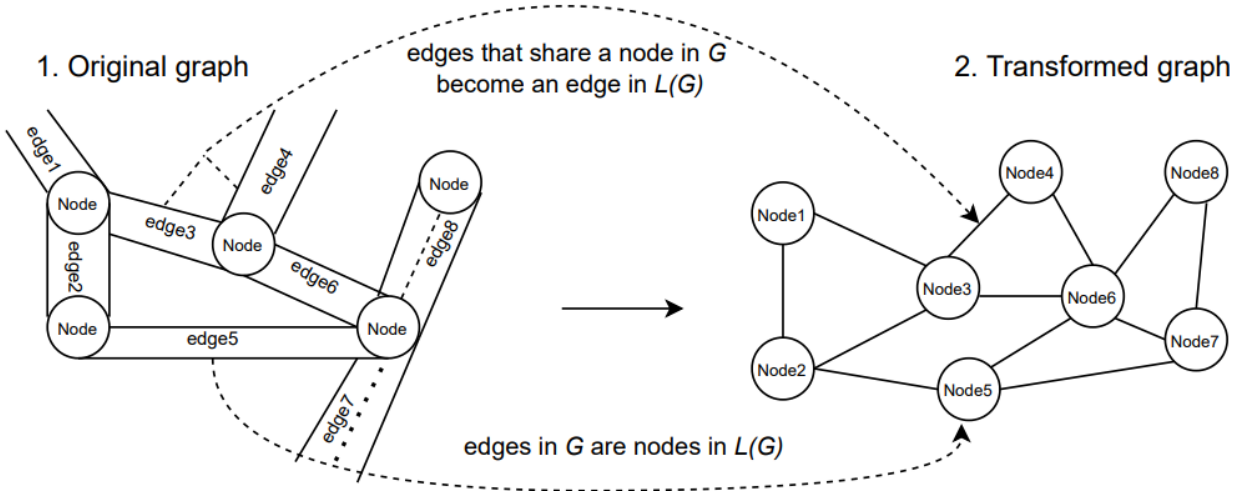

Fig. 2. Transformation of original graph to line graph

Nodes, represented by points, correspond to key locations such as intersections, dead-ends, and points of interest along the roads. Edges, represented by lines, connect these nodes and indicate the road segments between them.

Applications like traffic forecasting [1, 18, 14], speed limit optimization [12, 16, 7], and the estimation of travel time [6, 10] have witnessed successful implementations using machine learning techniques.

However, representing road networks as feature vectors for machine learning models poses a significant challenge due to the limited availability of suitable feature extraction methods in existing literature.

While recent advancements in deep learning offer an attractive avenue for automatically learning network structure and representing individual road segments based on their spatial connections to neighbouring segments, applying such techniques to graph-structured data presents unique difficulties.

Unlike commonly used data types like images and text, which are Euclidean and have fixed dimensions, the underlying connectivity patterns within graph-structured data (such as road networks) are inherently complex and non-Euclidean, posing challenges for directly applying existing deep learning methods.

Recent efforts in applying deep learning to complex, non-Euclidean graph data often rely on a fundamental approach: embedding high-dimensional graph features into a lower-dimensional Euclidean space using graph embedding techniques.

This process reduces the complexity of the data while preserving its essential relationships. By capturing these relationships in a simplified form, the model can tackle various graph-related tasks, such as predicting node attributes or connections between nodes.

Ultimately, graph embedding aims to represent each node (e.g., a road segment) with a lower-dimensional vector. This vector retains the node's similarity to the node in the original, allowing researchers to leverage standard metrics for similarity comparisons in the embedded space. Several studies have explored various graph embedding techniques for modelling road network tasks like traffic flow prediction.

While some, such as the Hybrid Graph Convolution Neural Network (HGCN) proposed in [17], capture the spatial connectivity of the network by representing toll stations as nodes and road segments as edges, however, the authors neglected to incorporate node-specific features beyond location, time, and weather.

Algorithm 1 Road segment feature extraction

Input: G and $L(G)$.
Output: Road segment feature vector.

1: **for** $s \in L(G)$ **do**
2: **From** G obtain l_s.
3: **From** s obtain $(x_s,\ y_s)$.
4: **From** s obtain segment geometry.
5: **if** geometry exist **then**
6: **Obtain** 20 equally distanced points of l_s: $(lx_i,\ ly_i)_{i=1,\ 2,\ ...,\ 20}$ by divid l_s.
7: **for** $i = 1$ to 20 **do**
8: **Subtract** $(lx_i,\ ly_i)$ from $(x_s,\ y_s)$.
9: **end for**
10: **else**
11: **Convert** to line geometry.
12: **Repeat** steps 6 to 8.
13: **end if**
14: **Obtain** S of the speed limits with m standard values.
15: **Concatenate** features generated from steps 2 to 14.
16: **end for**

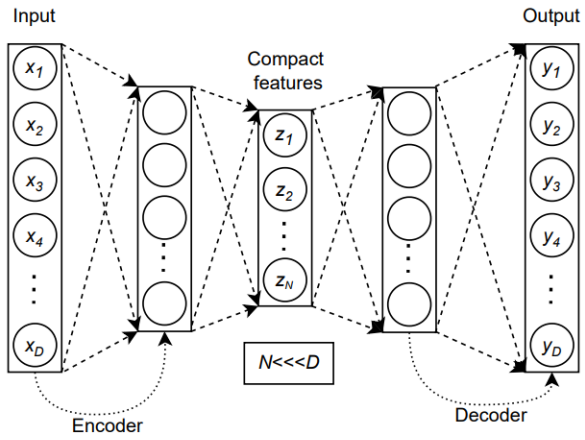

Fig. 3. Stage 1 embedding: Deep AutoEncoder

This gap is addressed in Relational Fusion Networks (RFN) introduced in [4] for speed limit classification and estimation. RFN leverages a novel graph convolution operator to effectively integrate edge information (e.g., road type, speed limits) into the representation learning process. This leads to a more comprehensive understanding of the network dynamics.

The study presented in [2] investigates the classification of different road types within realistic cities using a graph dataset extracted from Open Street Maps (OSM). Inspired by the Relational Fusion Network (RFN), the authors incorporate edge features into the learning process by transforming the initial graph into a line graph.

The further proposes a novel method for generating road segment features based on readily available attributes, such as road segment length, speed limit, and geometric characteristics. The authors then compare the performance of various embedding methods, including Graph Convolutional Neural Networks (GCCNs) [5], GraphSAGE [3], Graph Attention Networks (GATs) [13], and Graph Isomorphism Networks (GINs) [15], across different learning settings.

This comprehensive evaluation aims to identify the most effective approach for classifying road types within complex urban environments. A method to classify road types using Deep AutoEncoder (DAE) method is proposed in [2]. Similar to the work in [8], the authors generated road segment features using road attributes.

DAE was applied to reduce the dimensionality of input features while preserving the most essential features. Classification of road types was achieved using the MultiLayer Perceptron (MLP) classifier. The study proposed in [9] is an improvement to the studies presented in [2] and [8], where road segment features obtained by DAE are fed as input to GCNN before classifying road types with MLP classifier.

3 Materials and Methods

This study builds upon a novel, multi-stage graph embedding method for road-type classification tasks, originally proposed in [9]. As shown in Figure 1, the proposed method extracts road network graphs from Linkoping and Johannesburg cities using OSMnx, where nodes and edges are intersections and road segments, respectively.

The initial graph is then converted into a line graph where road segments are represented as nodes. Next, a feature extraction process is employed following the construction of both the original and transformed road network graphs.

Algorithm 2 Graph embedding with Deep Auto Encoder

Require: Original road features: $\mathrm{RSEGFS} \subset \mathbb{R}^N$
Outputs: Embedded road features: $\mathrm{RSES} \subset \mathbb{R}^M$

Parameter definition: DAE encoder and decoder.
Model definition: DAE model (encoder, decoder).
for $X \in \mathrm{RSGFS}$ **do**
    Fit input feature vectors (X) to DAE model.
    Randomly initialise weights.
    while no convergence in error difference **do**
        Produce feature vectors on the decoder (Y).
        Find the MSE between X and Y.
        Update weights.
    end while
    Obtain the embedding features vector (Z).
    $\mathrm{RSEGFS} \leftarrow \mathrm{RSEGFS} \cup \{Z\}$.
end for
Return features from the embedded space RSEGFS

Table 1. Datasets description

Dataset	Nodes	Edges	Classes
Linkoping	6,799	13,022	5
Johannesburg	17,431	39,980	5

This process leverages the structural information encoded in both graphs to extract road properties relevant for road type classification. The core innovation: a multi-stage graph embedding approach, is introduced in step 4. Similar to the original study, the first stage leverages Deep AutoEncoder (DAE) to compress high-dimensional feature vectors into lower-dimensional representations.

However, this study goes beyond the original work by investigating alternative methods for the second embedding stage. Instead of Graph Convolution Neural Networks (GCNN), it explores the use of GraphSAGE and Graph Attention Networks (GAT) to model road types based on the features obtained from stage 1. Finally, a MultiLayer Perceptron (MLP) classifier categorises the road types.

3.1 Input Graph Datasets and Transformed Graph

This study utilizes undirected road network graphs of Linkoping and Johannesburg cities extracted from OSMnx for experimentation.

The input graph is represented as $G = (V, E)$, where V represents nodes, and E represents edges. Nodes correspond to intersections, junctions, and crossroads, while edges represent road segments connecting these nodes.

This section will only refer to the Linkoping City road networks dataset for simplicity and clarity to explain how each algorithm was applied to input graph datasets.

Graph embedding techniques aim to embed node features. However, nodes in the original graphs (crossroads, junctions, and intersections) lack crucial information for road-type classification.

Therefore, transforming the original graph into a line graph is necessary to represent road segments as nodes, thus facilitating graph embedding. Fig. 2 depicts the process of converting original graph G into its corresponding line graph $L(G)$.

This transformation involves mapping each edge (road segment) in G to a distinct node in $L(G)$. Subsequently, edges in G that share a common node (e.g., junction) are transformed into connecting edges within $L(G)$.

3.2 Labelling Road Type Classes

OSMnx represents road segments with corresponding road type labels, enabling the use of supervised learning for modelling road networks.

Unfortunately, the data suffers from imbalanced classes, with certain road types rarely appearing. To address this, similar to the original work, certain road types are merged and assigned new labels as follows:

– Class 1: Highway, yes, primary, secondary, motorway-link, trunk-link, primary-link, secondary-link.
– Class 2: Tertiary-link, tertiary.
– Class 3: Unclassified, planned, road.
– Class 4: Residential.
– Class 5: Living street.

Algorithm 3 Graph embedding with Graph Convolution Neural Networks

Require: Road segment embedded graph features space: $\mathrm{RSEGFS} \subset \mathbb{R}^N$.
Outputs: Road segment embeddings: $\mathrm{RSES} \subset \mathbb{R}^M$.

Define k number of hops.
Define input and output layer dimensions at each k.
for $Z_v \in \mathrm{RSFS}$ **do**
    Construct computational graph.
    Initialise W_k.
    Set h_v^0 as Z_v.
    for $i = 1 : k$ **do**
        Using Eq 2, find h_v^i.
    end for
    Obtain embedded vector $E_v = h_v^k$.
    $\mathrm{RSES} \leftarrow \mathrm{RSES} \cup \{E_v\}$.
end for
Return RSES.

3.3 Feature Engineering

This study employs a feature engineering technique similar to the one used in [2] to ensure a fair comparison of results. In this technique, four key attributes from G and $L(G)$ are extracted to create a 58-dimensional raw feature vector for each road segment. These attributes are:

- Road segment length: Represented by a single dimension.
- Midpoint coordinates: Represented by two dimensions, one for longitude and one for latitude.
- Distances to nearby points: The midpoint is surrounded by 20 points spaced at equal distances, and the subtraction of these distances creates 20 dimensions.
- This categorical feature is represented using 15 dimensions, with each dimension corresponding to a possible speed limit.

For a given road segment s with length l_s, midpoint coordinates (x_s, y_s), and a one-hot encoded speed limit vector $S = \{s_1, s_2, s_3, \cdots, s_m\}$ (where m represents the number of possible speed limits), each road segment vector can be obtained using Algorithm 1.

3.4 Graph Embedding: The Multi-Stage Approach

The proposed multi-stage graph embedding method for classifying road types is described in this section. As previously discussed, the method employs two distinct embedding approaches. The Deep AutoEncoder (DAE) model is used in the first stage.

This model acts as a dimensionality reduction technique, compressing the high-dimensional feature vectors associated with each node (representing road segments) into a lower-dimensional, compact representation.

This compressed representation captures the essential characteristics of the road segments while discarding redundant information. In the second stage, the approach leverages graph embedding techniques to extract contextual information for each road segment.

This is achieved by incorporating the feature vectors of its neighbouring segments, previously obtained in Stage 1. Through this process, the method captures the influence and relationships between individual roads within the broader network structure.

3.4.1 Stage 1: Deep AutoEncoder Embedding:

The presented method utilizes a Deep AutoEncoder (DAE) to embed road segment features from a high-dimensional space (D) into a lower-dimensional space (N), where N is significantly smaller than D ($D >>> N$).

This dimensionality reduction process aims to achieve “compact” representations of the road segments. To understand the meaning of “compact” features, it’s crucial to grasp the DAE architecture. As depicted in Figure 3, the DAE comprises three key components:

- Encoder: This component receives a feature vector ($X_i = \{x_{i,1}, x_{i,2}, x_{i,3}, \cdots, x_{i,D}\}$) containing D features and processes it through several hidden layers with progressively decreasing dimensions.

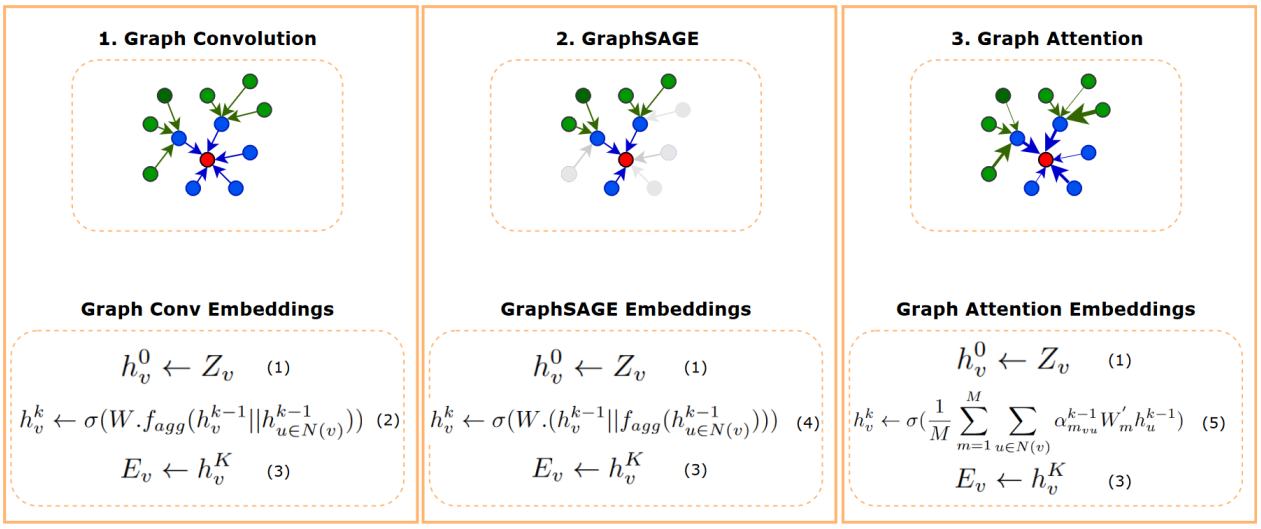

Fig. 4. Graph Neural Network approaches. 1) GCN: Target node (red) receives update from its direct neighbours (blue nodes) and neighbours of the neighbours (green nodes). 2) GraphSAGE: Target node (red) receives update only from k sampled nodes of its neighbours (blue) and m sampled nodes of neighbours of the neighbours (green). 3) GAT: Learns a scoring to weigh the influence of neighbouring nodes on the target node

– Compact Features Layer: This layer represents the heart of the dimensionality reduction. It compresses the encoded feature vector (X_i) into a lower-dimensional vector ($Z_i = \{z_{i,1}, z_{i,2}, z_{i,3}, \cdots, z_{i,N}\}$) with only N features ($N < D$).

– Decoder: This component takes the compact feature vector (Z_i) and utilizes it to reconstruct an approximation of the original feature vector ($Y_i = \{y_{i,1}, y_{i,2}, y_{i,3}, \cdots, y_{i,D}\}$) through several dense layers with increasing dimensions.

The "compactness" of the features in the compact features layer is defined by the error difference between the original feature vector (X_i) and its reconstructed counterpart (Y_i).

If this error difference is minimal; then, the compact layer features are considered "compact" as they effectively capture the essential information of the original features in a reduced dimension.

– Number and size of hidden layers: This parameter affects the capacity of the model in learning complex patterns.

– Learning rate: This parameter controls how quickly the model updates its weights during training.

– Dimensionality of compact features: This parameter determines the compression level achieved by the embedding.

DAE utilizes a fully connected neural network architecture comprising input, output, and dedicated "compact features" layers. The encoder and decoder have input and output layers, respectively, and they share similar numbers and sizes of hidden layers.

The process begins with normalizing the input feature vectors. These normalized vectors are then fed into the encoder. ReLU activation, defined as $f(x) = \max(0, x)$, is applied to introduce non-linearities. On the decoder's output layer, the reconstructed values are normalized between 0 and 1 using the sigmoid function, defined as:

$$g(y) = \frac{1}{1 + e^{-y}}. \tag{1}$$

Leveraging the Adam optimization algorithm, the encoder's weight parameters are iteratively

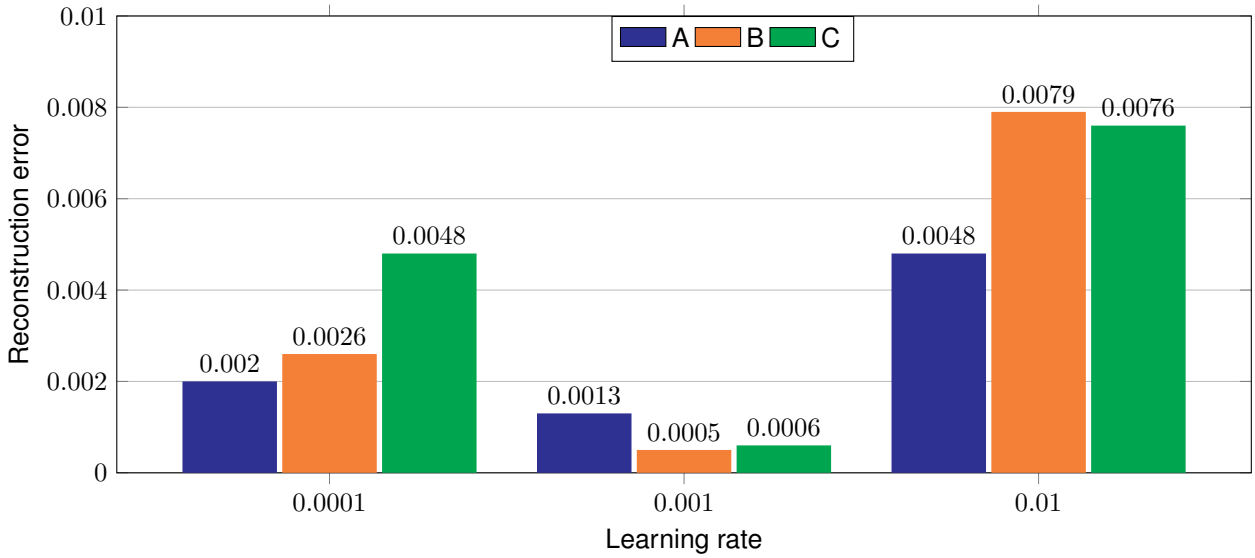

Fig. 5. Stage 1 embedding: Error difference at various DAE models and learning rates for Linkoping road network

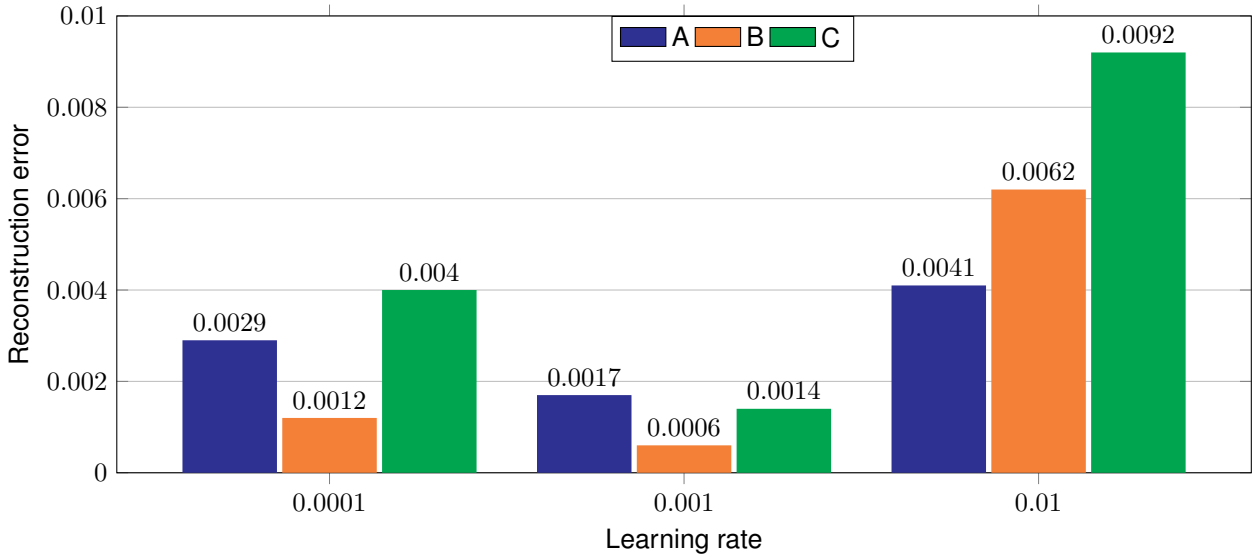

Fig. 6. Stage 1 embedding: Error difference at various DAE models and learning rates for Johannesburg road network

adjusted to minimize the reconstruction error, defined as the discrepancy between the input data and its encoded representation. Finally, the original D-dimensional road segment features are replaced with the N-dimensional compact features obtained from the model. Algorithm 2 describes the DAE model.

3.4.2 Stage 2: Embedding with Graph Neural Network Approaches:

Stage 2 leverages graph neural network (GNN) approaches to exploit the intrinsic topological and spatial relationships embedded within the graph-structured road network data.

Table 2. Description of various DAE models

Model	Hidden Layers	Sizes	N
A	5	$\{58, 49, 40, 31, 22, 13\}$	4
B	4	$\{58, 48, 38, 28, 18\}$	8
C	3	$\{58, 46, 34, 22\}$	10

This enables the extraction of richer feature representations for each road segment by incorporating information from its neighbouring segments, leading to a more comprehensive understanding of the road network's spatial context and connectivity.

As highlighted earlier, the ultimate goal of any GNN approach is to generate the embedded vector h_v^k of the sampled road segment v for each hop layer k by aggregating information from its direct neighbours $u \in N(v)$. Several GNN methods are available in the literature for generating such an embedded vector.

In this work, the comparison between GCNN, GraphSAGE and GAT is conducted. It is worth noting that inputs to each GNN are the road segment feature vectors $Z \subset \mathbb{R}^N$(from stage 1), and the outputs are the embedded vector $E \subset \mathbb{R}^M$.

3.4.3 Graph Convolution Neural Networks

A two-hop GCNN architecture [5] generates the embedded vector of a given road segment by aggregating features from its direct neighbours as well as the neighbours of the neighbours. As indicated in Equation 2 of Figure 4, GCN generates embedded vector h of target road segment v at any hop k by concatenating the embedded vectors h_v^{k-1} and $h_{u \in N(u)}^{k-1}$ of the target and neighbouring road segments, respectively at previous hop $k-1$.

It then uses some aggregator function f_{agg} to obtain the contribution of neighbouring road segments to the target road segment before applying the Sigmoid function σ. W represents the set of weights associated with the target and neighbour road segments. The experimental section of the study will investigate the performance of three GCNN aggregator functions namely, GCNN-Mean, GCNN-Max, and GCNN-Sum.

3.4.4 GraphSAGE

A two-hop GraphSAGE architecture [3] generates the embedded vector of a given road segment by aggregating information from only a set of sampled neighbouring road segments. As indicated in Equation 4 of Figure 4, GraphSAGE generates embedded vector h of target road segment v at any hop k by first applying the aggregator function f_{agg} to the embedded vector $h_{u \in N(u)}^{k-1}$ of neighbouring road segments, at previous hop $k-1$.

The aggregated vector is then concatenated to the embedded vector h_v^{k-1} before applying the Sigmoid function σ. The experimental section of the study will investigate the performance of three GraphSAGE aggregator functions: GSAGE-Mean, GSAGE-Max, and GSAGE-Sum.

3.4.5 Graph Attention Networks

Similar to GCNN, a two-hop GAT architecture [13] generates the embedded vector of a given road segment by aggregating features from its direct neighbours as well as the neighbours of the neighbours. However, GAT further learns the attention weights that describe the influence of each neighbouring road segment towards the target road segment.

As indicated in Equation 5 of Figure 4, GAT generates the average weighted embedded vector h of target road segment v at any hop k over multiple heads by applying attention weights α_{vu}^m to the corresponding neighbours shown in Algorithm 3 is the the pseudo-code for achieving the embedding task using GCNN. GraphSage and GAT follow the same pseudo-code with the only exception being the generation h_v^k.

3.5 Classifying Road Types With MLP Classifier

The study employs a Multilayer Perceptron (MLP) classifier, characterized by its non-linear activation functions and multiple hidden layers, for road type classification.

The MLP is trained, validated, and tested on feature vectors generated using a multi-stage graph embedding method. A concise summary of MLP parameters is provided, instead of

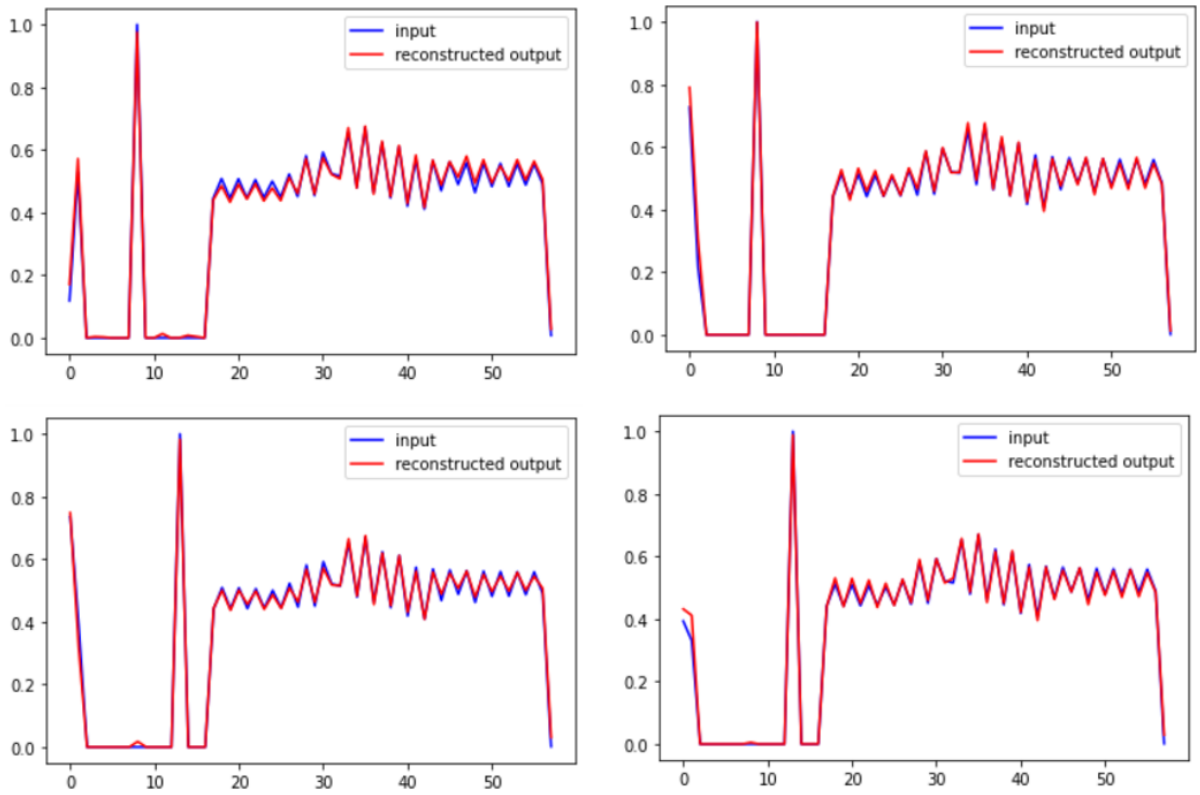

Fig. 7. Stage 1 embedding: Examples of actual and reconstructed road segment feature vectors for Johannesburg road network based on the test dataset

delving into the intricate mathematical framework of the MLP, which isn't crucial to this method's originality. To ensure a fair comparison with the approach presented in [9], a single hidden layer MLP classifier equipped with the Adam optimizer is employed.

The input layer size matches the dimensionality of the input road segment, while the output layer size aligns with the number of road type labels (five classes). Road segment feature vectors are fed through the input and hidden layers of the MLP classifier. The output layer leverages the softmax activation function to generate probability values for each road-type class.

The cross-entropy loss function measures the discrepancy between predicted and true class labels. The Adam optimizer then utilizes this calculated loss to update the MLP's weight parameters.

4 Experimental Results

Datasets of road network of Linkoping and Johannesburg cities, with 6761 and 17431 road segments (nodes), respectively, are used to carried out the experiments. Algorithm 1 is used to generate 58-dimensional feature vector from each road segment.

4.1 Stage 1: Graph Embedding with Deep Autoencoder

The aim of this stage is to embed road segment features from D dimensional space ($D = 58$) into N dimensional space with compact road segment features. Section 3.4 describes how to produce compact features. Road segments data was split into 50% for the training set, 20% for the validation set used to obtain the optimal DAE parameters and 30% for the testing set.

Table 3. Parameter settings required for the experiments

Parameters	
Learning rate	{0.001, 0.01, 0.1}
Output dimension	{16, 32, 64}
Epochs	1000
Batch size	1024
Dropout	0.2

Table 4. Prediction results for Linkoping road network datasets: Results for different graph embedding methods are shown in terms of micro F1-Score. Training time for every 50 epochs is also shown

Approach	Training Time (s)	Val. F1	Test F1
Raw features	04	62	59
DAE	02	66	64
GCN-Sum	26	77	72
GCN-Mean	31	76	70
GCN-Max	32	79	75
GSAGE-Sum	20	78	77
GSAGE-Mean	23	77	76
GSAGE-Max	29	79	78
GAT	29	78	76

Table 2 describes the parameters of several DAE models at varying numbers and sizes of hidden layers, respectively. For instance, model A has 5 hidden layers of size {49, 40, 31, 22, 13} on the encoder and decoder component, while the compact layer size (N) is 4.

Figure 5 shows the error difference obtained by each DAE model at varying learning rate parameters for the Linkoping road network based on the test set.

Figure 6 shows the error difference obtained by each DAE model at varying learning rate parameters for the Johannesburg road network based on the test set. It can be seen that the lowest possible error difference is achieved with model B at 0.001 learning rate parameter.

This shows that DAE successfully performs the embedding of road segments 58-dimensional feature space into 8-dimensional space for both Linkoping and Johannesburg road network datasets. Figure 7 depicts the examples of actual and reconstructed road segment feature vectors obtained by the DAE model for the Johannesburg road network using a test dataset.

4.2 Stage 2. Embedding with Graph Convolution Neural Network

In this stage, the aggregation of information from neighbouring road segments is used to generate road segment embedded vectors using methods discussed in section 3.4.2. As in [9], the dataset was split into 70% for training, 15% for validation and 15% for testing. Each embedding model comprises a definition of a two-hop layer, in which inputs of the first layer are the 8-dimensional road segment feature vectors generated from stage 1. The M-dimensional output of the second layer is fed into the MLP classifier. The experiments are designed mainly to investigate the performances of various graph embedding methods for modelling both Linkoping and Johannesburg road networks.

4.2.1 Hyperparemeter Settings

As discussed in section 3.4.2, experiments are conducted using 7 different graph embedding approaches namely GCNN-Mean, GCNN-Max, GCNN-Sum, GSAGE-Mean, GSAGE-Max, GSAGE-Sum and GAT.

For each approach, the model that achieves the lowest micro F1 score during the validation process is selected as the best-performing model, and it is further tested on the test set. Furthermore, various learning rates (0.001, 0.01 and 0.1) and output dimension M parameters (16, 32, and 64) are investigated on each approach to obtain optimal parameters. Batch normalization is applied after each layer as a regularizer.

Combining different graph embedding approaches, output dimensions and learning rates yield 126 models for both Linkoping and Johannesburg road network datasets. Table 3 illustrates the parameters required to conduct the experiments.

Table 5. Prediction results for Johannesburg road network datasets: Results for different graph embedding methods are shown in terms of micro F1-Score. Training time for every 50 epochs is also shown

Approach	Training Time (s)	Val. F1	Test F1
Raw features	08	67	64
DAE	03	74	70
GCN-Sum	91	78	74
GCN-Mean	88	78	73
GCN-Max	116	79	73
GSAGE-Sum	65	84	81
GSAGE-Mean	81	86	84
GSAGE-Max	70	85	82
GAT	67	79	76

Table 6. Comparison of impact of method used on classification performance: The proposed method uses DAE features as input to graph embedding methods, while the method proposed in [2] uses raw features as input. The results compare the micro F1 score of GCNN, GraphSAGE and GAT using DAE features and raw features

Approach	Proposed method F1	Other method F1
GCN-Mean	70	58
GSAGE-Mean	76	62
GAT	76	76

4.2.2 Linkoping Road Networks

Table 4 shows the micro F1 score achieved by the best model on each approach based on the test set for road type classification on the Linkoping road network graph dataset. The training time after 50 epochs is also presented.

The results obtained by each embedding method are compared with the performance of DAE features and only when raw features are given to the classifier. Raw features refer to the 58-dimensional road segment features generated by Algorithm 1.

DAE features are embedded features obtained in the first stage of the proposed method, where Algorithm 2 generates 8-dimensional road segment features. As indicated, using only raw features yields the micro F1 score of 59%.

Also, using DAE features slightly improves the micro F1 score to 64%. This slight improvement is understandable given that DAE features are much more robust and accurate compared to raw features. It can be observed that all 7 graph embedding methods in the second stage of the proposed method outperform both raw features and DAE features.

However, GSAGE-Max outperforms the rest of the methods with 22% improvement compared to the performance of raw features. It can further be observed that GAT and GraphSAGE approaches have much shorter training time compared to GCNN approaches, this observation is reasonable given that GCNN uses all the neighbouring road segments to generate an embedded vector, whereas GraphASAGE and GAT only take a sample of neighbours.

4.2.3 Johannesburg Road Networks

Table 5 shows the micro F1 score achieved by the best model on each approach based on the test set for road type classification on the Johannesburg road network graph dataset.

The training time after 50 epochs is also presented. The results obtained by each embedding method are compared with the performance of DAE features and only when raw features are given to the classifier.

Raw features refer to the 58-dimensional road segment features generated by Algorithm 1. DAE features are embedded features obtained in the first stage of the proposed method, where Algorithm 2 generates 8-dimensional road segment features. As indicated, using only raw features yields the micro F1 score of 64%.

Also, using DAE features slightly improves the F1 score to 70%. It is further observed that all 7 graph embedding methods in the second stage of the proposed method outperform both raw features and DAE features.

However, GSAGE-Mean outperforms the rest of the methods with 20% improvement compared to the performance of raw features.

4.3 Comparison with Existing Works

Performance of the proposed method was compared to the model presented in [2] for classification of road types of Linkoping City. There are similarities between and GCNN methods: (1) they use the same road network graph dataset. (2) They use similar classifier parameters.

They differ on the embedding approach. In fact, Graph embedding methods proposed in [2] apply embedding using raw data as opposed to the proposed method that will initially use DAE to generate the compact version of road segments before graph embedding methods is applied.

Table 6 shows the comparison of the methods in terms of micro f1 score. It can be observed the method proposed in this study outperforms the method proposed in [2] for road type classification when GCNN-Mean and GSAGE-Mean are used as graph embedding methods.

The results achieved for GAT are similar for both studies. However, it can be observed that the use DAE embedding approach significantly improves the performance of graph embedding methods for road-type classification tasks.

5 Conclusion

A multi-stage graph embedding method for the classification of road has been presented. Experiments are conducted using the Linkoping and Johannesburg road networks dataset extracted from OpenStreetMaps. Similar to [2], road attributes such as length, mid-point coordinates, geometry and speed limit are used to generate raw features for each road segment.

Embedded road segment feature vectors are produced from raw features using a two state graph embedding method. GCNN (Sum, Mean and Max), GraphSAGE (Sum, Mean, and Max) and GAT methods were used in this study to investigate their performance for road type classification on the obtained road network graph datasets.

The results indicated that all seven methods outperform both raw and DAE features. Furthermore, GraphSAGE-Sum and GraphSAGE-Mean outperform other methods for classifying road types in Linkoping and Johannesburg cities, respectively. The results obtained by GCNN-Mean, GraphSAGE-Mean and GAT on the Linkoping road dataset were compared to the methods proposed in [2], where a similar dataset was used to solve the same tasks when only raw features were input to the graph embedding methods.

Results further indicate that the use DAE embedding method to generate compact road segment features significantly improves the performance of graph embedding methods for modelling road types. Future work of the study will generate more road segment features using attributes such as lane count.

Furthermore, replacing the one-hot encoding method with deep neural network embedding for representing categorical features is worth attempting. Additionally, the F1-score metrics used in this study have been found to exhibit a bias influenced by the imbalanced degree of the imbalanced dataset. Therefore, future work will utilise metrics such as the Matthews Correlation Coefficient (MCC).

References

1. **Deekshetha, H. R., Madhav, A., Tyagi, A. (2022).** Traffic prediction using machine learning. Evolutionary Computing and Mobile Sustainable Networks Lecture Notes on Data Engineering and Communications Technologies, pp. 969–983. DOI: 10.1007/978-981-16-9605-3_68.
2. **Gharaee, Z., Kowshik, S., Stromann, O., Felsberg, M. (2021).** Graph representation learning for road type classification. Pattern Recognition, Vol. 120, pp. 108174. DOI: 10.1016/j.patcog.2021.108174.
3. **Hamilton, W. L., Ying, R., Leskovec, J. (2017).** Inductive representation learning on large graphs. Proceedings of the 31st

Conference on Neural Information Processing Systems, pp. 1–11.

4. **Jepsen, T. S., Jensen, C. S., Nielsen, T. D. (2022).** Relational fusion networks: Graph convolutional networks for road networks. IEEE Transactions on Intelligent Transportation Systems, Vol. 23, No. 1, pp. 418–429. DOI: 10.1109/TITS.2020.3011799.
5. **Kipf, T. N., Welling, M. (2017).** Semi-supervised classification with graph convolutional networks. Proceedings of the 5th International Conference on Learning Representations, pp. 1–14.
6. **Masiero, L., Casanova, M., Tilio, M. (2011).** Travel time prediction using machine learning. Proceedings of the 4th ACM Special Interest Group on Spatial Information and International Workshop on Computational Transportation Science, Vol. 11509. DOI: 10.1145/2068984.2068991.
7. **Modi, S., Bhattacharya, J., Basak, P. (2022).** Multistep traffic speed prediction: A deep learning based approach using latent space mapping considering spatio-temporal dependencies. Expert Systems with Applications, Vol. 189, pp. 116140. DOI: 10.1016/j.eswa.2021.116140.
8. **Molefe, M., Tapamo, J. R. (2023).** Road-type classification with deep autoencoder. Computational Intelligence and Neuroscience, Vol. 2023, pp. 1–14. DOI: 10.1155/2023/1456971.
9. **Molefe, M. E., Tapamo, J. R. (2023).** A new approach for road type classification using multi-stage graph embedding method. Proceedings of the Mexican Conference on Pattern Recognition, Vol. 13902, pp. 23–35. DOI: 10.1007/978-3-031-33783-3_3.
10. **OpenStreetMap Contributors (2022).** Planet osm. planet.osm.org.
11. **Sahoo, J., Rath, M. (2017).** Study and analysis of smart applications in smart city context. International Conference on Information Technology (ICIT), pp. 225–228. DOI: 10.1109/ICIT.2017.38.
12. **Szwed, P. (2019).** Speed limits can be determined from geospatial data with machine learning methods. International Conference on Artificial Intelligence and Soft Computing, pp. 431–442. DOI: 10.1007/978-3-030-20915-5_39.
13. **Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y. (2018).** Graph attention networks. Proceedings of the 6th International Conference on Learning Representations, pp. 1–12. DOI: 10.48550/ARXIV.1710.10903.
14. **Vázquez, J. J., Arjona, J., Linares, M., Casanovas-Garcia, J. (2020).** A comparison of deep learning methods for urban traffic forecasting using floating car data. Transportation Research Procedia, Vol. 47, pp. 195–202. DOI: 10.1016/j.trpro.2020.03.079.
15. **Xu, K., Hu, W., Leskovec, J., Jegelka, S. (2018).** How powerful are graph neural networks? Proceedings of the 6th International Conference on Learning Representations, pp. 1–17.
16. **Yan, M., Li, M., He, H., Peng, J. (2018).** Deep learning for vehicle speed prediction. Energy Procedia, Vol. 152, pp. 618–623. DOI: 10.1016/j.egypro.2018.09.220.
17. **Yang, F., Zhang, H., Tao, S. (2022).** Hybrid deep graph convolutional networks. International Journal of Machine Learning and Cybernetics, Vol. 13, No. 8, pp. 2239–2255. DOI: 10.1007/s13042-022-01520-y.
18. **Yin, X., Wu, G., Wei, J., Shen, Y., Qi, H., Yin, B. (2022).** Deep learning on traffic prediction: methods, analysis, and future directions. IEEE Transactions on Intelligent Transportation Systems, Vol. 23, No. 6, pp. 4927–4943. DOI: 10.1109/tits.2021.3054840.

Article received on 04/07/2023; accepted on 13/10/2023.
** Corresponding author is Mohale E. Molefe.*