Evaluation of a hybrid traffic routing model to handle unexpected traffic events

Abhishmita Gogoi Kalita
School Of Electronic Engineering
Dublin City University
Dublin, Ireland

***Abstract*—Traffic regularization is a growing necessity in smart cities. This paper uses shortest path algorithm and neural network to find the shortest path between source and destination node pairs. Solutions have been developed using complex deep learning models. It is implemented and compared with shortest path algorithms and the results are documented.**

***Index Terms*—Bellman-Ford Algorithm, Dijkstra's Algorithm, RNN**

I. Introduction

The advent of urbanization and increase in urban living space has increased the distance for urban commute. One of the major difficulties faced by commuters daily is traffic congestion. Traffic congestion results from high volume of vehicles on the road and overuse of roads. The road infrastructure also plays a pivotal role in it. With the introduction of Intelligent Transport System (ITS) in smart cities, the research area for finding the optimal route neural network started in 1992.

There are two types of traffic congestion recurring and non recurring congestion [**?**]. Regular instances of traffic such as traffic during rush hours are known as Recurring congestion. Non recurring congestion are caused due to unpredictable events-such as road closure, construction, accident. Figure 1 shows the average hours lost per driver due to traffic congestion. The research for non-recurrent traffic congestion is limited . This paper [1] gives a study on route planning during non-recurrent events using neural network.

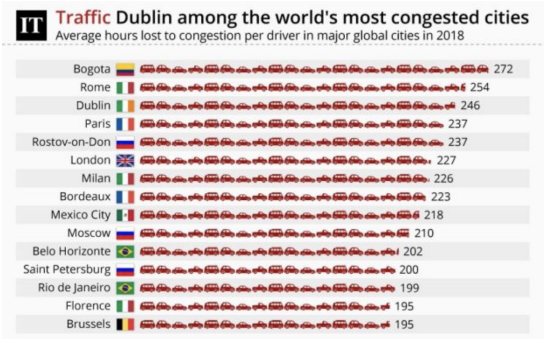

Fig. 1. Average hours lost per driver to traffic congestion [1].

II. Prior Work

Unexpected traffic events, roadside construction work, environmental factors affect traffic adversely [7]. There are two strategies to rectify it by changing given public infrastructure or by using active traffic management on current infrastructure.

Change in public infrastructure is not economically feasible but also physically difficult to attain without disturbing the current traffic conditions. As the outcome of change in infrastructure does not guarantee the predicted state.With the first-time usage of neural networks for traffic congestion forecasting and control, in the last decade the accuracy has increased significantly. Aggregation approach to short term traffic prediction developed by Man et al [4], has outstanding results. However, it still falls short, when an accident or an unlikely traffic incident takes place. In such unexpected traffic events, the prediction accuracy is affected drastically. The nature of road traffic is not static, most of the research previously done have used static methods to give a desired output. Non-recurrent road traffic is dynamic in nature, due to its unpredictability. Latest approaches of Jiaming et al [5] focuses on using both real and historical data to analyze further machine learning models and use big data to focus on prediction of traffic volumes and rerouting of traffic in case of unexpected or emergency traffic events. Lucas [8] uses Recurrent Neural Network (RNN) to find the shortest path with an updated process, the commonality among these three papers is they are for route planning for recurrent traffic congestion.

There are two approaches for traffic prediction, namely model driven approach and data driven approach. In the model driven approach, a computational model of the road network is simulated and traffic behavior is analyzed. To make the model more accurate, different road conditions are added, such a speed-limits, junctions and lanes are taken into account. A majority of these models lack the ability to assimilate real-time information. These models are more beneficial for long term prediction, such as sports events or concerts, where there is less demand for change in traffic conditions. A data driven approach means predicting traffic conditions by analysing and interpreting data, creating models and gaining valuable insights with the data in hand. The approach used for this study is data driven where graphs are generated and changed dynamically to get the most optimal path.

III. Technical Description

The approach for this study is data driven. The first step is generating graphs with nodes that resemble real traffic routes. The second step is defining the shortest path algorithms. Next,

designing a neural network considering previous research study. The weights are adjusted dynamically to resemble a non-recurrent traffic scenario. Models trained are evaluated later.

A. Graph Traversal

The most common types of graph traversal is based on the order of node exploration. They are breadth first search and depth first search. As the name suggests, breadth first search traverses the graph horizontally from left to right or vice versa, based on the problem statement. Depth first search traverses the graph edge wise and it is more efficient in terms of computing [9]. The use of heuristics is prevalent to find the nearest node for exploration.

B. Shortest Path Algorithm

Shortest path algorithms are used to find the nearest nodes between source node and destination node by finding the smallest weight for each route. The most commonly used shortest path algorithm is Dijkstras algorithm.

1) Dijkstras Algorithm: It solves the shortest route problem on directed, non-negative weighted graphs. Its always chooses the closest or the lightest node from the origin. A representation of Djikstra can be shown in figure [1]below

Algorithm 1: Dijkstra Algorithm

input : A Graph Q having v nodes.
output: Distance between the nodes

```
Function Dijkstras Algorithm(Source, Graph)
    Q ← the set of all nodes in Graph i
    for each vertex v in Graph : do
        dist [v] ← ∞
        prev [v] ← ∅
        dist [source] ← 0
        while Q is not empty do
            u ← node in Q with smallest dist [ ]
            remove u from Q
        alt ← dist[u] + dist_between(u,v)
        if alt < dist [v] then
            dist [v] ← Alt
            prev [v] ← u
        return prev []
```

Fig. 2. Dijkstra Algorithm [2].

Dijkstras algorithm relaxes all the nodes in the graph and sets the initial distance source as zero and the distance to other nodes as infinity. This is a greedy algorithm and it looks for the smallest weight as it traverses from one node to the other. Dijkstras algorithm applies breadth first search for node traversal. Dijkstras performance on bi-directional roads is limited [13].

2) Bellman-Ford Algorithm: Bellman-Ford Algorithm is a dynamic programming algorithm. It is similar to Dijkstra's algorithm but its computationally more efficient than it. The algorithm starts from bottoms up and it takes the edges with shortest distance. It then calculates nodes with two edges and so on. This algorithm acknowledges negative cycles.

Algorithm 2: Bellman Ford Algorithm

```
Function Bellman Ford Algorithm(Source, Graph)
    for each vertex v in Graph : do
        dist [v] ← ∞
        prev [v] ← ∅
        dist [source] ← 0
        for each vertex v in Graph : do
        temp_dist ← dist [u] + edge_weight(u,v)
        if temp_dist < dist [v] then
            dist [v] ← temp_dist
            prev [v] ← u
    foreach edge (U,V) in Graph do {
    if dist [u ] + edge_weight(u,v) < dist [v] then
        dist [v] ← Alt
        prev [v] ← u
    return dist [],prev []
    }
```

Fig. 3. Bellman Ford Algorithm [2].

C. Neural Network

Neural network are computing systems that vaguely resemble the biological neural network consisting of neurons of animal brains. Neurons are multilayered for better performance and accuracy. In a neural network inputs are fed to the system and the system trains on the given input and gives the desired output. Based on the type of problem they are binary or multi class neural network.

A basic neural network architecture consists of an input layer, a number of hidden layers and an output layer. The ideology behind neural networks is the application of weights on the layers. There can be a million parameters that can be used as weights. It will not yield the desired output. The process for selecting weights is called backpropagation and its given in details below. The correct weight value matters as it alters all the layers involved in a neural network.

D. Recurrent Neural Network

The neural network used for this study is recurrent neural network (RNN). The most important aspect of it is the use of historical output to be used as inputs in the hidden layers. This is due to the use of back propagation. It is a variation to feed-forward network (FF). The purpose of a neural network is to find the error and to minimise it. The input from one layer is fed to the next layer in a Feed-forward network. A feed forward network is given in fig. 4 It can be seen there is no loop to transfer the output back to the input in a feed-forward network. Vanishing gradient are often encountered in case of recurrent neural network.

From fig. 4, it can be deduced that feed forward networks are simple neural networks that never forms a loop. Delta rule

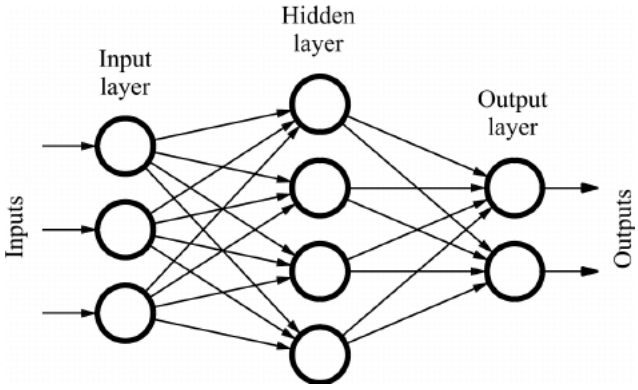

Fig. 4. Architecture of a feed forward network [10].

is applied by neural networks to compare the weights with the expected weight values, this weight values are used by the model to give the desired output. To minimise error during prediction backpropagation algorithm is applied. This algorithm takes the error values obtained from previous iterations and fine tunes it to find the suitable weight.

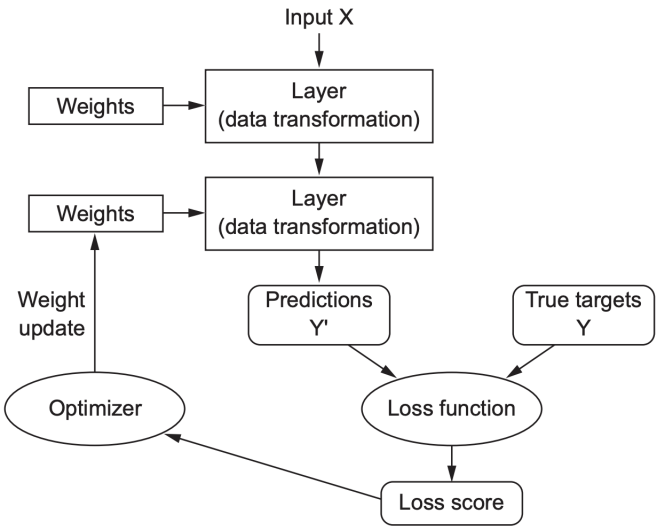

Fig. 5. Overview of the backpropagation algorithm [11].

There is a wide range of neural network models. There is very less background knowledge except for research by Tan et al [3] and Jiaming et al [5]. This study does not consider their design due to the following reasons:

1) Due to computational limitation, the time for training the data based on the models is very high.
2) The model used are feed-forward networks, the process of choosing weights is not clearly stated.
3) The data which is used are from their country of Origin, which is not on public domain and is difficult to get access to.

Recurrent neural network (RNN) model is used for this study due to benefits of backpropagation and computational efficiency. A basic RNN model is shown in fig 6.

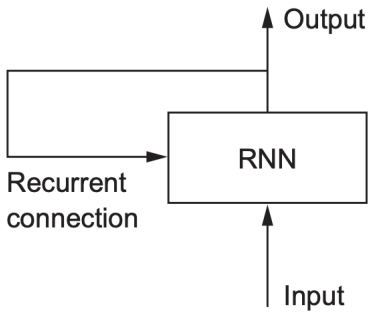

Fig. 6. Overview of the backpropagation algorithm [11].

E. Data Augmentation

Training data used for neural network needs to be as large as possible to give desired outputs. In this study, data augmentation is implemented to prevent overfitting. Data Augmentation generates more training samples from the input samples provided. It augments the data by a number of random transformations [11]. As the data used for this study is simulated data, with data augmentation the number of samples is increased. Data augmentation helps the model to generalise data in a better way.

This method is widely used in video, audio and text classification due to hints of spatial nature in them [12]. The data simulated also showcases some spatial feature that can be utilised for data augmentation.

F. Learning Algorithm

Supervised learning algorithms is used to make a decision based on the input-output pair. There are two types of problems. They are regression or classification. The motivation for using neural network is to decrease the edge weights for finding the shortest path.

1) Regression : The estimator used is mean squared error. It minimises sum of weight directly
2) Classification: The estimator used is cross entropy error. Its output is the accuracy of the model performance.

For this study, the problem is solved using classification. Regression estimator, mean squared error gives the output as a real number and its not accommodating for the model performance. To access the working of a neural network, model performance is a more concrete criteria. Cross entropy error is given by1.

$$CE(y, y^{\wedge}) = -\sum_{i=1}^{n} y_i log y_i^{\wedge} \tag{1}$$

The cross entropy loss finds the distance between probability probability distributions. In this study, it is the distance between the ground truth and the predicted output. [11]

The second reason is classification is a more consistent, computationally and with the number of data wrangling processes involved for this study. Classification is the best option.

IV. Experimental Setup

A. Graph creation

In this section, the generation of data has been discussed. Initially, the data graph is generated randomly according to the number of pair nodes, shown in the above figure. Once the nodes are created, they are connected with a pre-defined number of pairs with edges. For the designing of neural network, local heuristics such as edge hub is considered.

Global heuristics capture the information of edges calculated through Manhattan distance and cosine similarity. Finally, the generated nodes and edges are divided randomly into training for neural network and testing to get a realistic outcome. With the use of Greedy strategy, random graphs are created having n number of nodes, there are some added limitations to the nodes that replicate road networks. Each node has at most four neighbours and the edge weights are increased on a particular node to simulate an accident or a non recurrent traffic congestion scenario. The other nodes have a uniform weight. A randomly generated graph is shown in fig. 7

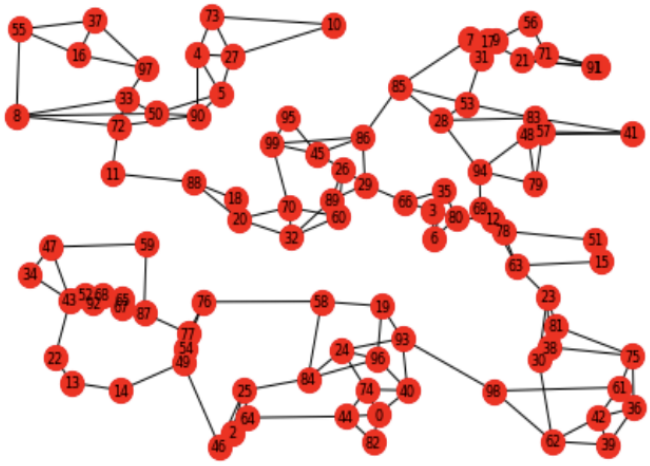

Fig. 7. Random graph generation with number of nodes n = 100 and weights of (0,1).

Manhattan distance and cosine distance are very useful in clustering. Manhattan distance is generalised for higher dimensions such as integrated circuits [14]. Due to the above reasons its chosen as one of the global heuristic. The equation for Manhattan distance for distance between points a1 (x1,y1) and a2 (x2,y2) is given by eqn. 2

$$Manhattan\_dist = |x^1 - x^2| + |y^1 - y^2| \tag{2}$$

Cosine index on the other hand measures the angle between the two nodes to predict the right direction [14]. The equation for cosine index between two vectors A and B is given by eqn.3. The combination of both the above distance and direction indexes helps the neural network in navigating throughout the graph.

$$Cos\theta = \frac{A.B}{||A|.|B||} \tag{3}$$

B. Algorithms Used

Edge hub is chosen based on the number of times the route repeats across the same edge or node. The data sampled from the graph is random in nature, to replicate dynamic road networks. Dijkstra's algorithm and Bellman Ford algorithms are considered fed to the neural network.

The architecture of the neural network design is given in fig. 8 In this section, an overview of the neural network discussed. To obtain the result, for this study, two fully connected layers are used to map the input vectors with output. Activation layer is used to add non-linearity into the network for complex mapping. Dropout layer is used to prevent the network from being saturated and finally, soft max function is added to get the desired outcome. Before processing, the input vector passes through data augmentation, where the volume of data increases. This method helps the network to shuffle the input vector iteratively until it does not find the next node with the shortest path.

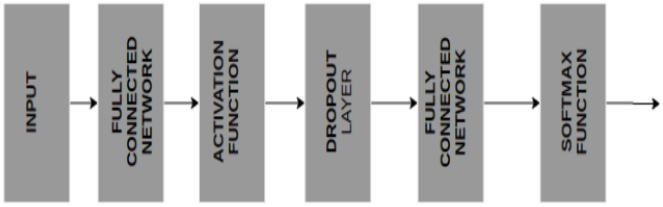

Fig. 8. Neural network design.

1) Rate of Arrival: Arrival rate (AR) for the model is given by the following eqn. 4. The higher the arrival rate, the more accurate is the model performance.

$$AR = \frac{No.of nodepairsinthepath}{No.of sampled nodepairs} \tag{4}$$

V. Result

This section demonstrates the results obtained from the neural network approach. For this study, to get the effectiveness of the model arrival rate calculated, which stated as 4 Here N(p) is number of pairs, and N (s) number of sampled node.

VI. Analysis

In this section, the analysis of results as described. Figure 9 shows the arrival rate of the neural network with an accident and without an accident. Where there is no accident, the neural network shows high performance. However, this performance degrades after a certain number of pair nodes. Neural network touches at the peak value of 77% when there are 20 nodes paired with a node. On the other hand, when there is an accident, the neural network shows the high performance of 67% with ten nodes in pair. After 20 pair nodes, the result shows almost a constant trend as there is little drop

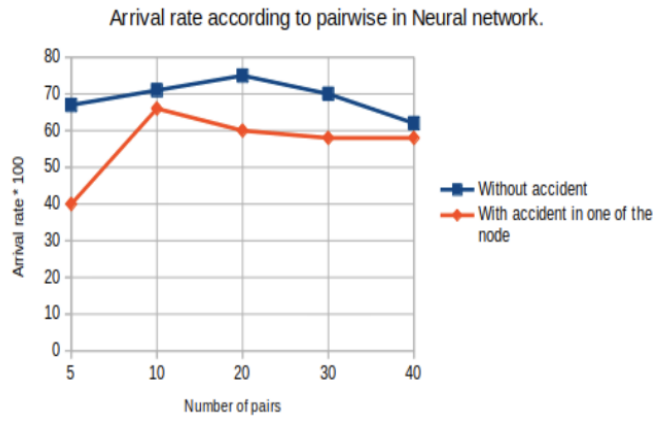

Fig. 9. Arrival rate of model according to pair-wise nodes, without accident and with accident in one edge.

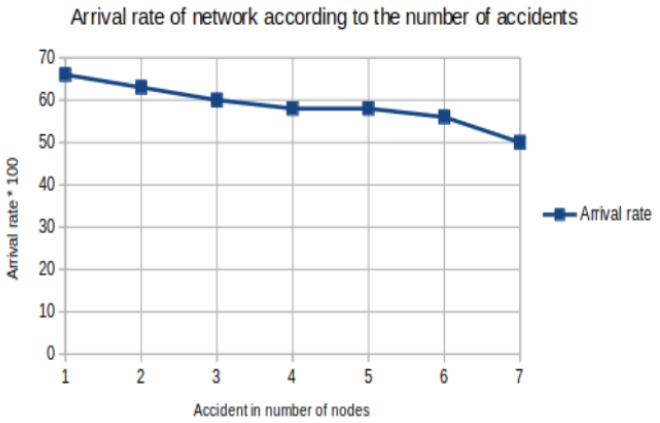

Fig. 10. Arrival rate of model according to number of accidents occurred

in the arrival rate. Figure 10 shows the performance of the model when the incident increases. There is a clear trend that the arrival rate degrades when the accident increases. The performance drops from 67% arrival rate with one accident to 49% when the accident occurs in seven nodes.

A. Comparison between neural network and Bellman-Ford Algorithm

In this section, comparison has made between the performance of Bellman-Ford and neural network. Figure 13 shows the arrival rate of the neural network and Bellman-Ford algorithm. As the number of pair increases, the neural network shows almost similar trends. Whereas, Bellman-Ford performance falls drastically on increasing the pair number of nodes. Belman-Ford algorithm shows 100% arrival rate when a single node is connected with five pairs. On the other hand, the neural network shows highest arrival rate of 77% when a single node connected to 20 pairs.

Figure 14 shows the performance of algorithms when the number of accident increases on different nodes. There is a slight difference in the arrival rate of both algorithms. Bellman-Ford shows 65% arrival rate when there is an accident in a single node; whereas, neural network shows 67%. On increasing the count of accidents, Bellman-Ford drops from 65% to 60%. On the other hand, the neural network degrades from 67% to 58%.

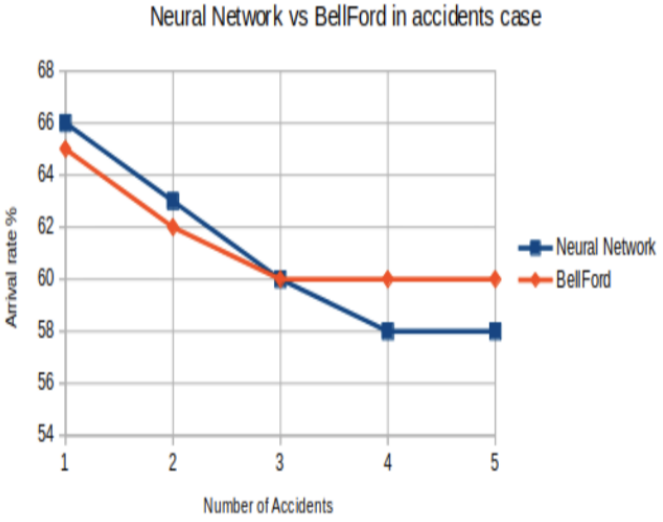

Fig. 11. Comparison of neural network model and Bellman-Ford model with accident

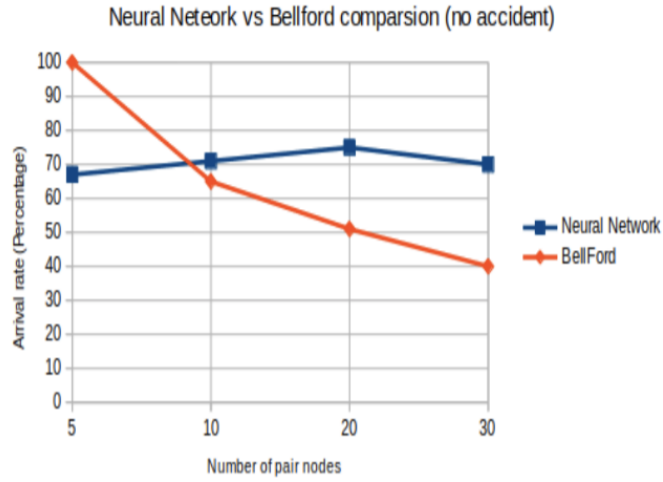

Fig. 12. Comparison of neural network model and Bellman-Ford model without accident

B. Comparison between neural network and Bellman-Ford Algorithm

VII. Conclusion

The results obtained from the given approaches above in relation to this project are obtained and the following conclusions are made:

- It can be found that neural network outperforms both Bellman-Ford and Dijkstra in a scenario without accidents. It can be seen that Dijkstra's Algorithm performance decreases with increase in nodes. This proves that

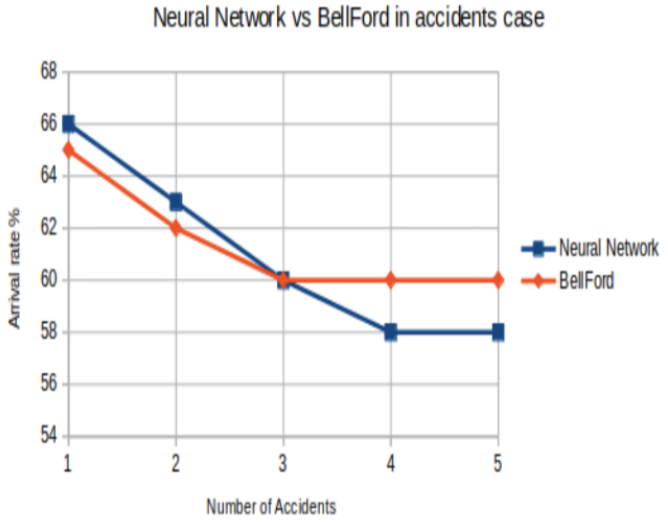

Fig. 13. Comparison of neural network model and Bellman-Ford model with accident

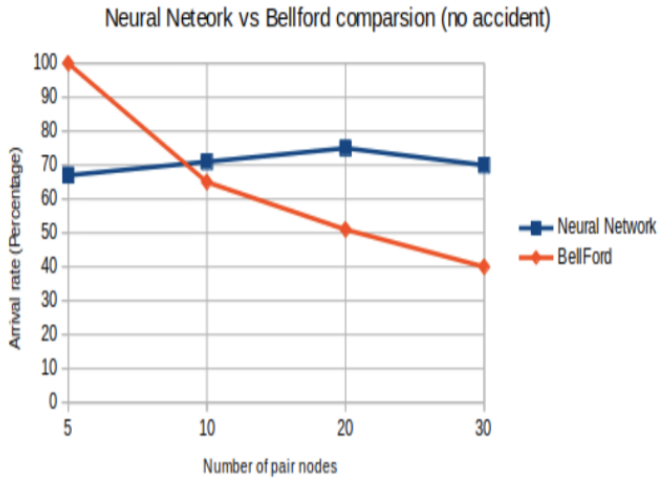

Fig. 14. Comparison of neural network model and Bellman-Ford model without accident

Dijkstra's algorithm is more helpful for finding shortest path in a static routing system.

- Bellman-Ford algorithm on the other hand gives a better performance than Dijkstra's algorithm in the case of no accidents. It even outperforms the neural network when it comes to adding accidents. This shows that Bellman-Ford algorithm is best suited for dynamic routing system and its performance does not degrade to a greater extent with increase in nodes in comparison to Dijkstra's algorithm.
- In all, neural network scores the best performance in the three scenarios with the use of a distance and direction index, except in the scenario with accidents. Bellman-Ford algorithm outperforms it.

VIII. Future Work

- This model can be improved by working on graphs having more nodes and increasing the number of hidden layers in the neural network.In future, this model can be used on a real map and with availability of real time data, a cloud based platform can be made to make actual real time routing possible.
- Environmental data [7] also reflects traffic conditions. Understanding these conditions will help in active traffic management across Smart cities across the globe.

References

[1] Abhishmita GK, MPECE Project Portfolio, Main Research paper.
[2] Abhishmita GK, MPECE Project Portfolio, Appendix A, Literature Review Report (LRR). Submitted via LOOP
[3] C. Pope and M. Hilliard, "Dublin one of worst cities in world for traffic congestion", The Irish Times, 2020. [Online]. Available: https://www.irishtimes.com/news/ireland/irish-news/dublin-one-of-worst-cities-in-world-for-traffic-congestion-1.3791651.
[4] M. Tan, S. C. Wong, J. Xu, Z. Guan and P. Zhang, "An Aggregation Approach to Short-Term Traffic Flow Prediction," in IEEE Transactions on Intelligent Transportation Systems, vol. 10, no. 1, pp. 60-69, March 2009, doi: 10.1109/TITS.2008.2011693.
[5] Xie, J. and Choi, Y.-K. (2017) 'Hybrid traffic prediction scheme for intelligent transportation systems based on historical and real-time data', International Journal of Distributed Sensor Networks. doi: 10.1177/1550147717745009.
[6] Falcocchio, J. C., Levinson, H. S. (2015). Road traffic congestion: A concise guide. (Springer tracts on transportation and traffic; Vol. volume 7). Springer. https://doi.org/10.1007/978-3-319-15165-6
[7] Thomas et al, "Whether weather matters to traffic demand, traffic safety, and traffic operations flow", Sage Journals, vol. 1948, no. 1, pp. 170-176, Jan 2006
[8] Valatka, Lukas. (2019). Recurrent Neural Network Models for Route Planning in Road Networks. 10.13140/RG.2.2.24440.49928.
[9] Kozen D.C. (1992) Depth-First and Breadth-First Search. In: The Design and Analysis of Algorithms. Texts and Monographs in Computer Science. Springer, New York, NY. https://doi.org/10.1007/978-1-4612-4400-4$_4$
[10] DeepAI. (2019, May 17). Feed Forward Neural Network. Retrieved August 29, 2020, from https://deepai.org/machine-learning-glossary-and-terms/feed-forward-neural-network
[11] CHOLLET, F. (2018). Deep learning with Python.
[12] S. C. Wong, A. Gatt, V. Stamatescu and M. D. McDonnell, "Understanding Data Augmentation for Classification: When to Warp?," 2016 International Conference on Digital Image Computing: Techniques and Applications (DICTA), Gold Coast, QLD, 2016, pp. 1-6, doi: 10.1109/DICTA.2016.7797091.
[13] Sunita, Deepak Garg, Dynamizing Dijkstra: A solution to dynamic shortest path problem through retroactive priority queue, Journal of King Saud University - Computer and Information Sciences, 2018.
[14] Pandit, S., amp; Gupta, S. (2011). A Comparative Study on Distance Measuring Approaches for Clustering. International Journal of Research in Computer Science, 2(1), 29-31. doi:10.7815/ijorcs.21.2011.011