Towards ultra-latency using deep learning in 5G Network Slicing applying Approximate k-Nearest Neighbor Graph Construction

Rohit Kumar Gupta (rohit10495@gmail.com)
IIT Patna: Indian Institute of Technology Patna https://orcid.org/0000-0002-1080-2651
Praduman Pannu
IIT Patna: Indian Institute of Technology Patna
Rajiv Misra
IIT Patna: Indian Institute of Technology Patna

Research Article

Keywords: Network Slicing, k-Nearest Neighbor Graph Construction, Latency, Deep Learning

Posted Date: March 23rd, 2021

DOI: https://doi.org/10.21203/rs.3.rs-162479/v1

Noname manuscript No.
(will be inserted by the editor)

Towards ultra-latency using deep learning in 5G Network Slicing applying Approximate k-Nearest Neighbor Graph Construction

Rohit Kumar Gupta · Praduman Pannu · Rajiv Misra

Received: date / Accepted: date

Abstract The 5G Network Slicing with SDN and NFV have expended to support new-verticals such as intelligent transport, industrial automation, remote healthcare. Network slice is intended as parameter configurations and a collection of logical network functions to support particular service requirements. The network slicing resource allocation and prediction in 5G networks is carried out using network Key Performance Indicators (KPIs) from the connection request made by the devices on joining the network. We explore derived features as the network non-KPI parameters using the k-Nearest Neighbor (kNN) graph construction. In this paper, we use kNN graph construction algorithms to augment the dataset with triangle count and cluster coefficient properties for efficient and reliable network slice. We used deep learning neural network model to simulate our results with KPIs and KPIs with non-KPI parameters. Our novel approach found that at k=3 and k=4 of the kNN graph construction gives better results and overall accuracy is improved around 29%.

Keywords Network Slicing · k-Nearest Neighbor Graph Construction · Latency · Deep Learning

Rohit Kumar Gupta
Indian Institute of Technology Patna
E-mail: rohit10495@gmail.com

Praduman Pannu
Indian Institute of Technology Patna

Rajiv Misra
Indian Institute of Technology Patna

1 Introduction

The 5G and beyond networks are conceived to provide new emergence multi-services with various diverse range of specifications. Network slicing [1] is the key-enabler to provide 5G networks with its multi-services including new-verticals like intelligent transport, industrial automation, remote healthcare. Network slicing allows the transition between network-as-an-infrastructure setup and network-as-a-service to provide 5G smart services [2]. Software-Defined-Networking (SDN) and Network-Functions Virtualization (NFV) have been widely recognized as the service provisioning paradigm in the next generation networking due to its numerous advantages such as scalability, flexibility, efficiency, short deployment cycles etc [3]. Prviously, the Telecommunication Service Providers (TSPs) used to totally depend on middleware proprietary software or hardware devices for management and allocation of the network services. Although, the substantial operating expenditures (OPEX) and capital expenditures (CAPEX) was involove. In contrast, the SDN-NFV applications efficitvely reduces the cost and meet with the changing needs accordance to the users with the given Quality-of-Services (QoS), Quality of experience (QoE) and the Service-level agreement (SLA) [4].

3GPP, commonly known as the 5G global standardization communications community mainly focuses on three major 5G use cases that are comprised of Ultra-Reliable Low Latency Communications (URLLC) with low latency and high reliability, massive-MachineType Communications (mMTC) with high connection density and enhanced-Mobile Broadband (eMBB) with peak datarate [5]. The ability to serve emerging services in 5G depends on the different requirements in terms of latency, availability, reliability, throughput and specific domain related functionality [6]. The physical network of infrastructure layer has to be sliced into the multiple-logical networks of the varying structure and sizes based on the different characteristics and requirements like smartphones, industry 4.0 devices or autonomous cars etc. In the 5G networks, the softwarization using SDN-NFV are expected to fill the gap in the programmable control at the orchestration and management. 5G Network Slicing are the key enablers for network hypervisors, containers and virtual machines, SDN-NFV, cloud/fog computing, Mobile Edge Computing (MEC) [7]. The network slice provides slice-as-a-service (SlaaS) emerging public-cloud service which is introduced by new business mode of multi-tenancy network architecture. SlaaS is distinguished from classical cloud environment and services in terms of resource management complexity due to heterogeneity of network slices, while classicial cloud services are generally homogeneous [8]. The few slice admission objectives are revenue optimization, QoS and inter-slice congestion control, slice fairness accurance etc [9].

In this paper, we investigate to appropriate slice selection for a device by the Key Performance Indicators (KPIs) with deep learning neural network (DNN). Further, we apply k-Nearest Neighbour Graph Construction algorithms and derive the non-KPIs features. The current state-of-the-art models are only using these KPIs as features to train their models for slice prediction,

allocation and are not able to fully capture the relationship between device requests and associated slice. In the fact that the allocation of network resources per slice are based on previous KPIs requirements only and doesn't factor in the traffic requirements and network needs of similar requests. We computed with various k values and found that at k=3 and k=4 it gives better results. The deep learning model help to analyze the overall traffic patterns and can be able to predict the future traffic.

The remaining of the paper is organized as follows: In Section 2, we discuss the related work. In Section 3, we discuss the system model and problem statement. In Section 4, we construct kNN graph with triangle count and cluster coefficient followed by the network slice deep learning neural network (DNN) model. Further in section 5, we discuss the results, performance evaluation and analysis. Finally, in section 6, we conclude this work.

2 Related Work

In paper [10], authors have derived the slice admission-control problem and proposed Slice-as-a-Service (SlaaS) for on demand slice services. The multi-queue based controller is considered to get network slice request of the waiting tenants. In paper [11], agarwal *et. al.* seek to make resource allocation and VNF placement to support vertical services. The queuing-based system model have been used at the orchestrator to correct match with the vertical requirements optimally also, efficient with fast solution strategy author have proposed which reduces the solution complexity. In [12], authors have addressed the technical challenge of the actual virtual functions placements within the physical network and provided a real optimal approximation algorithms. For the performance evaluation authors have used two measures: the served distance between clients and the virtual functions and the setup cost of these virtual functions. In the paper [13], author have used machine learning algorithm to allocation of the network slice in 5G networks. The state of the art on the 5G use cases have been discussed with the QoS (Quality of Service) requirements like: latency, reliability, availability and throughput. In the paper [14], authors have developed deep learning neural network based deepslice model which uses network KPI to train the model to analyze the incoming traffic. The network failure scenario have also been discussed as master slice.

There are few works related to graph construction algorithms which we used to add non-KPIs features to the existing dataset for better overall accuracy. The authors Wang *et. al.* [15], have proposed an approach for approximate kNN graph construction. To achieve the approximate neighborhood graph the division of data points on hierarchical and random basis have been done. For more accurate neighborhood graph sereval time this process have been repeated. The authors have shown both theoretical and empirical efficiency of the kNN graph construction and demonstrate some speed-up to deal with large scale data. In the paper [16], authors have present *greedy filtering,* with an efficient algorithm to find approximate k-NN graph by filtering node

pairs. A faster and scalable version of greedy filtering have been introduced based on node prefixes of inverted indices. The authors have conduced extensive experiments with the seven different datasets to compare the approach to the state of the art algorithms, the result shows that greedy filtering guarantees the high level of accuracy. In the paper [17], authors have presented an efficient algorithm of approximate k-nearest neighbor graph construction with different similarity measures like jaccard, cosine, Earth Mover's Distance. This method has minimal space overhead and works based on local search that need not rely on global share index hence it is suitable for applications where mostly the data is distributed over the different network. In the paper [18], authors take a step to extend semi-supervised machine learning for unsupervised learning domain adaptation. The authors analyze the conditions to achieve better propagation based on affinity graph according to their proposed scheme. The authors Prokhorenkova *et. al.* [19] theoretically analyze graph-based nearest neighbor search and evaluate the performance. The authors also analyze the two heuristics beam search and shortcut edges that widely used in practice. In the paper [20], authors propose the latent-variable model that represent low dimensional of high-dimensional data. The authors aim to preserve neighbor's neighborhood relationships by sparse graph by computing low-dimensional embedding. For large datasets, authors develop an approximate graining procedure that avoid negative sampling for nonadjacent nodes.

3 System Model

3.1 Network Slicing Model

The combination of network slicing in 5G networks permit the customers for data processing with high speed connectivity to the business requirements. The LTE network which describes "All size fits all" scenario that used shared resources. In 5G, network slicing have introducing diverse services like: smart home, smart city, self-automation, mission-critical applications, self-driving cars. 5G also is able to provide different business many business type models such as MVNO, business vs consumers, third-party services, CSP services etc. The single network fit for all is too complex and network slice can optimize the network differently to meet the diverse requirements of different applications. Network slice takes one physical network end-to-end and form virtual slices with each slice can have able to manage independently. The network slice can have its own dedicated resources that may not used by other slices and therefore we can not optimize those slices to meet this very diverse performance requirements. There are few 5G business requirements with service quality like data rate with low latency, energy efficiency, context-aware network, QoE-aware billing and pricing, QoE with user experiance, availability security and mobility [7]. The Next-Generation Mobile Networks (NGMN) defines network slicing as the central role in 5G networking architecture that comprises of 3-layers architecture as physical layer, network slice and service instance layer.

The 5G performance target by ITU [21] is given as Down-Link (DL) 20 Gbps, Up-Link (UL) 10 Gbps with less than 1 ms latency in RAN and connection density upto 1,000,000 devices per km^2, three usage scenario is shown in Fig. 2. The eMBB is intended to provide video streaming, web browsing, email application type of traffic with very high data rate. The second category of slicing is called URLLC with certain class of applications like self-driving cars, virtual reality, augmented reality that have very low round trip time to operate effectively. Some of the URLLC applications also need high degree of reliability like in health care application. The third slice is mMTC and in this category of usage scenario a huge number of devices like Internet of Things (IoT). A huge number of devices in a given area with those devices may not necessary extremely high data-rate and extremely low latency.

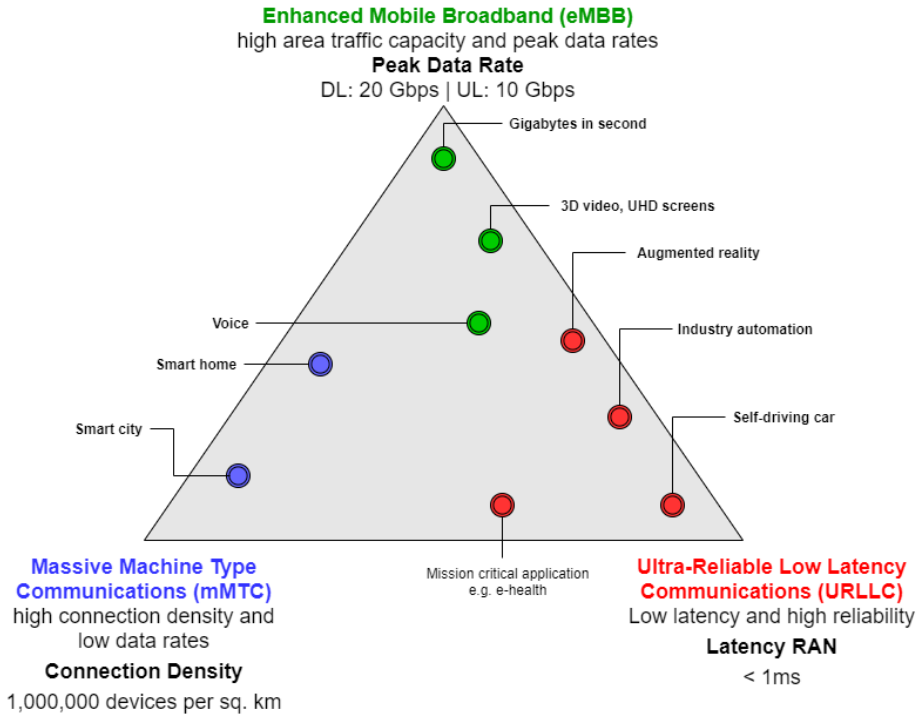


Fig. 1: ITU usage scenario for 5G

In Fig. 1, we illustrate the 5G Network Slicing Model which has eMBB, URLLC and mMTC slices at the each triangle node. The various services have shown in the triangle with their requirements based on QoS and QoE like very high data-rate per second, 3D videos and Ultra-HD screens, voice, smart-home, augmented reality, industry automation, smart-city, self-drivng cars and critical application.

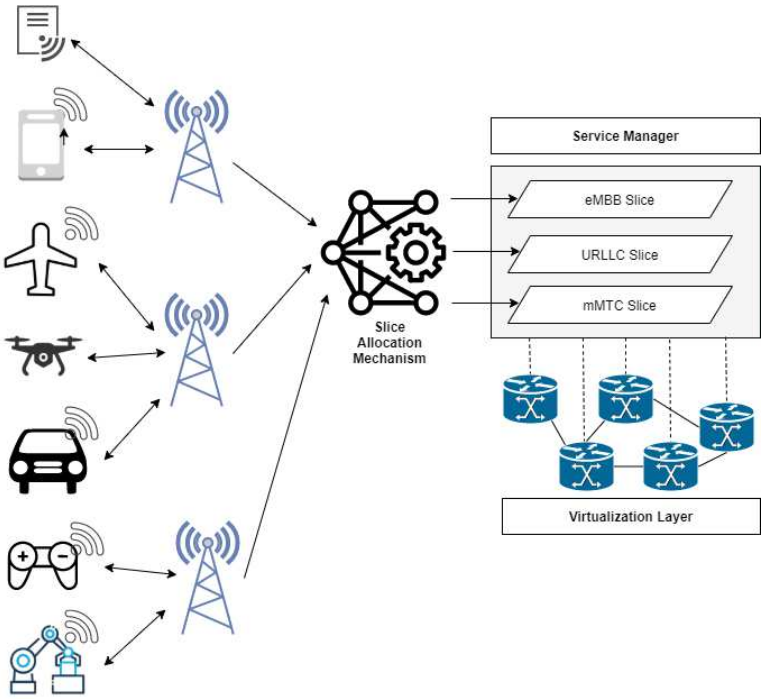


Fig. 2: Overview of Network Slice architecture

In 5G network slicing there are diverse range on devices connected to it and based on their requirements it assign to a suitable slice. The network slice architecture overview is shown in the Fig. 2. The three slices eMBB, URLLC and mMTC are on the top of virtualization layer and linked with Application Programming Interfaces (APIs). The virtualization layer consists of the abstraction of all the physical network resources that are composed of connection between server, virtual links and forwarding elements [22]. The service manager is placed on the top of the network slice layer that has network orchestrator which manage the entire network traffic by defined Service-Level Agreement (SLA). The SLA is described by the each type of services based on latency, reliability, availability, throughput etc.

The service manager works as the network control and orchestrator which composed of several modules and treats every received users request. The user demand is well communicated to the service manager and the admission control mechanism maps to the users demand to the deployed slices. We use deep learning neural network (DNN) to allocate the users to a specific slice based on the KPIs. When the framework receives the users request, DNN analyses its requirements and push to a slice that suits perfectly to the user demand. DNN makes a smart decision by recording all the user information and in future it efficiently allocate network resources.

3.2 Problem Formulation

The network slicing allows service providers to built their own slicing-model for current use cases and some emerging applications. There are few challenges in the 5G network slicing like appropriate selection to the slice for a user based on incoming traffic, slice monitoring based on traffic prediction, slice analytics based on slice assignments. We model the slice allocation mechanism with DLL based on initial KPIs of the dataset. With the DLL model based on KPIs the slice can be predicted but, in order to increase the overall accuracy the k-Nearest Neighbour Graph Construction algorithms has been introduced. By the introduction of kNN graph construction some non-KPI could be added like triangle count, cluster coefficient etc.

3.2.1 kNN Graph Construction

We have considered V as dataset where distance metric d is defined as: $d : V \times V \rightarrow [0, +\infty)$. The smaller distance have higher similarity, so let us consider similarity $\varsigma = -d$. For any radius r, defined in $r \in [0, +\infty)$, the scope of r-ball is defined around $v \in V$ as $A_r(v) = \{u \in V | d(u, v) \leq r\}$. The V be a matic space is said to be growth restricted if there exist a constant k, such that

$$|A_{2r}(v)| \leq k|A_r(v)|, \qquad \forall v \in V \tag{1}$$

We consider the cosine similarity for this work as the similarity measure ς in kNN graph construction. The cosine similarity is given as:

$$similarity(u_i, u_j) = \frac{u_i \cdot u_j}{|u_i||u_j|}, \qquad \forall u_i, u_j \in V \tag{2}$$

4 k-NN Graph Construction for Network Slicing

4.1 Constructing k-NN graph

k-Nearest Neighbor graph construction is known to be important operation with the web applications where network traffic data has been analysed like similarity search and other machine learning problems. kNN graph construction algorithm is based on the simple principle: *the neighborhood of a neighbor is also likely to be a neighbor*. It means, if we are having approximate kNN for each node, then we can further improve the kNN approximation by exploring for each nodes neighbor's neighbor by current approximation. The kNN graph for a set of objects i.e. dataset V is considered as directed graph that has an edge defined from $v \in V$ to its most similar k objects with the given similarity measure. To construct the neighborhood graph the straight-forward strategy is to apply method of approximate nearest neighbor search. For better understanding, we have given table of notations in Table 1.

Table 1: Notations

V, N	dataset V of size N, where $N = \|V\|$
u, v	objects in V
A, B	kNN list, B is reverse of A
ς	similarity measure
r	radius
k	growing constant of V
M	large object belong to V
α	sample rate
β	precision parameter
T(G)	triangle count
C(G)	cluster coefficient

Consider the dataset V of size N where $N = |V|$, and let the similarity measure $\varsigma : V \times V \rightarrow \mathbb{R}$. We used cosine similarity in this paper discussed in section 3.2.

Lemma 1: Let for the whole dataset the diameter is D. As long as we select a large object M in V (other than v) for each $v \in V$, $A_M(v)$ is considered as v's k-nearest neighbor. The each object of M items are likely to found within a radius $D/2$ by exploring its neighbors' neighbor [17].

Proof: We have approximate kNN graph A so, $\bar{A}[v] = \cup_{\bar{v} \in A[v]} A[\bar{v}]$ are set of points, A could be improve by exploring. Suppose that accuracy of A is reasonably good for a certain radius r. Let, k be as growing constant in dataset V and consider $M = k^3$.

For any $u_i \in A_{r/2}(v)$, since, $\bar{v}$ is also in the set $A_r(v)$ so,

$$Pr(\bar{v} \in A[v]) \geq M/A_r(v) \tag{3}$$

distance$(u, \bar{v}) \leq$ distance(u, v) + distance$(v, \bar{v}) \leq$ r so,

$$Pr(u \in A[\bar{v}]) \geq M/A_r(\bar{v}) \tag{4}$$

The k neighbors are uniformly distributed in $A_r(v)$ in $A[v]$.

$$|A_r(v)| \leq k|A_{r/2}(v)| \tag{5}$$

For certain event $k << |A_{r/2}(v)|$ because $M = k^3$.

$$|A_r(\bar{v})| \leq k|A_{r/2}(\bar{v})| \leq k|A_r(v)| \leq k^2|A_{r/2}(v)| \tag{6}$$

Combining equations 3-6, we get

$$Pr(\bar{v} \in A[v] \wedge v \in A[\bar{v}]) \geq M/|A_{r/2}(v)|^2 \tag{7}$$

Total there are $|A_{r/2}(v)|$ candidates for such $\bar{v}$, then

$$Pr(u \in \bar{A}[v]) \geq 1 - (1 - M/|A_{r/2}(v)|^2)^{|A_{r/2}(v)|^2} \approx M/|A_{r/2}(v)| \tag{8}$$

The heuristic argument discussed above suggest that each object of M items are likely to found withina radius $D/2$ by exploring its neighbors' neighbor.

Algorithm 1: kNN Graph Construction algorithm

1. **Input:** Dataset V, Similarity ς, M
2. **Output:** k-NN list in the form of heap A
3. Initialization;
4. A[v] $\leftarrow$ INSTANCE(V, M), $\forall v \in V$
5. iteration
6. B $\leftarrow Reverse(A)$
7. [v] $\leftarrow A[v] \cup B[v]$, $\forall v \in V$
8. count $\leftarrow 0$
9. **for** $v \in V$ **do**
10. **for** $u_1 \in [v], u_2 \in \bar{A}[u_1]$ **do**
11. $p \leftarrow \varsigma(v, u_2)$
12. count $\leftarrow count +$ UPDATEKNN$(A[v], < u_2, p >)$
13. **END for**
14. **END for**
15. **return** A **if** count=0;
16. **END**

function INSTANCE(Q,m)
return Sample m items for the set Q

function REVERSE(A)
do
B[v] *in* u $|< v_1, v_2, .. > \in A[u]$, $\forall v \in V$
return A

function UPDATEKNN(Q,$< v, p >$)
UPDATE heap Q; return 1 if any changed else return 0;

The basic-kNN graph construction algorithm flow is shown in Algorithm 1. The input of the algorithm is the dataset V with similarity measure and output gives kNN list in the form of heap. For each object, we start selecting a random kNN approximation that iteratively improved the approximation by comparing each item of its current neighbor's neighbor. The reverse-kNN and kNN both usages to improve the approximation and terminates when there is no improvement.

The Algorithm 2 is the optimized version of Algorithm 1 which uses some optimization techniques. In this two list *used* and *now* have been introduced which uses the list with flag. Initially, all items of the *used* list is copied with the list A and list *now* has copied $\alpha \cdot M$ items, where α is the sampling rate. Both the list have been used to update the counter and terminated with list A where count is less than $\beta \cdot VM$, where β is the precision parameter. The precision parameter is the fraction of true kNN that are missed due to early termination. We considered the default precision parameter which is $\beta = 0.001$.

Algorithm 2: Efficient kNN Graph Construction algorithm

1. **Input:** Dataset V, Similarity ς, M, sample rate α, precision parameter β
2. **Output:** k-NN list in the form of heap A
3. Initialization;
4. A[v] $\leftarrow$ INSTANCE(V, M), $\forall v \in V$
5. iteration
6. **for** v $\in$ V **do**
7. $used[v] \leftarrow$ all values of A[v] with FALSE flag
8. $now[v] \leftarrow \alpha \cdot M$ values of A[v] with TRUE flag
9. remaining items in A[v] with FALSE flag
10. $\overline{used} \leftarrow$ REVERSE$(used)$;
11. $\overline{now} \leftarrow$ REVERSE(now);
12. count $\leftarrow 0$; //counter
13. **for** v $\in V$ **do**
14. $\overline{used}$[v] $\leftarrow \overline{used}$[v] $\cup$ INSTANCE$(\overline{used}[v], \alpha \cdot M)$;
15. $\overline{now}$[v] $\leftarrow \overline{now}$[v] $\cup$ INSTANCE$(\overline{now}[v], \alpha \cdot M)$;
16. **for** $u_1 < u_2$ where $u_1, u_2 \in now[v]$ or $u_1 \in now[v], u_2 \in used[v]$ **do**
17. $p \leftarrow \varsigma(u_1, u_2)$
18. count $\leftarrow$ count + UPDATEKNN$(A[u_1], < u_2, p, \text{TRUE} >)$;
19. count $\leftarrow$ count + UPDATEKNN$(A[u_2], < u_1, p, \text{TRUE} >)$;
20. **END for**
21. **END for**
22. **return** A **if** count $< \beta \cdot VM$
23. **END**

This optimizations can have multiple evaluation of similarity between two objects. The optimized algorithm can have a complexity of $O(N^2)$ in worst case running time. This complexity can expensive for the large dataset but at the same time it can have optimal time on map reduce implementation. A record of list consists of key-object and a list which measures the distances of the neighbors of the key-object. The realization of the above algorithm can be done in two MapReduce operations. First, the mapper linked with the input list and reverse of the kNN list. The reducer merges with the kNN list and does the reverse kNN operation. Second, the mapper does the local join operation and linked with the input input list. The reducer does the merge operation with the neighbors of each key-object and keeps the only top K items.

The one of the most fundamental substructures in the graph is triangle. The two fundamental measurements are triangle count $T(G)$ and clustering coefficient $C(G)$. The triangle count formula is given in the Eq. 9 where, $\lambda(v)$ is the total number of triangles in the graph that contain vertex v. To count all the triangles, $\lambda(v)$ counts for all V. Since, each triangle has been counted three times so, the final sum is divided by 3 to obtain triangle count T [23].

$$T(G) = \frac{1}{3} \sum_{v \in V} \lambda(v) \tag{9}$$

The cluster coefficient C_i of the graph is calculated based on the triplets of nodes given in Eq. 10. A triplet is a set of three vertices that can have two or three edges. If there three edges then it is a triangle and it is called as *closed triplet*. If there are two edges then it is known as *open triplet*. The triplet are counted for each vertices and added together. The three closed triplets will count as triangle because there will be one closed triplet associated with each of the three vertices [24].

$$C(G) = \frac{1}{n} \sum_{i=1}^{n} Ci \tag{10}$$

4.2 Network slicing deep learning model

The deep learning neural networks (DNN) works well with good accuracy when the data is large and unstructured. We use to train the multiple neurons of the model with the dataset KPI and non-KPIs so that it can predict a slice for a device from incoming traffic. DNN model learns what type of device allocated to which slice and predict accurately for the future traffic requests. In Fig.3, we show the DNN model overview which has two hiden layers ReLU, tanh, SoftMax activation functions has been used.

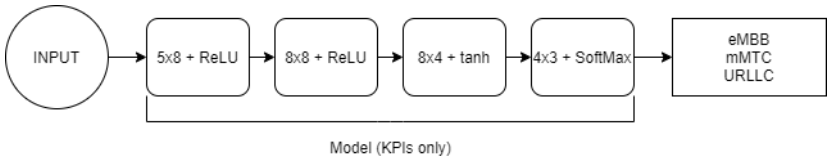


Fig. 3: DNN Model Overview

The Fig. 4 shows the augmented DNN model which simulated with the augmented dataset that contains KPIs and non-KPIs features. Our augmented DNN model can predict very accurately.

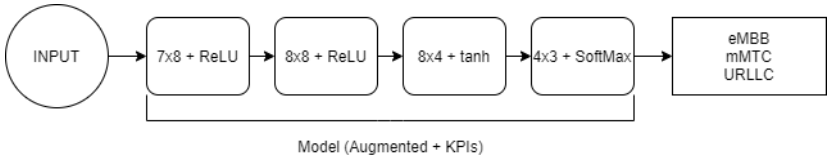


Fig. 4: Augmented DNN Model Overview

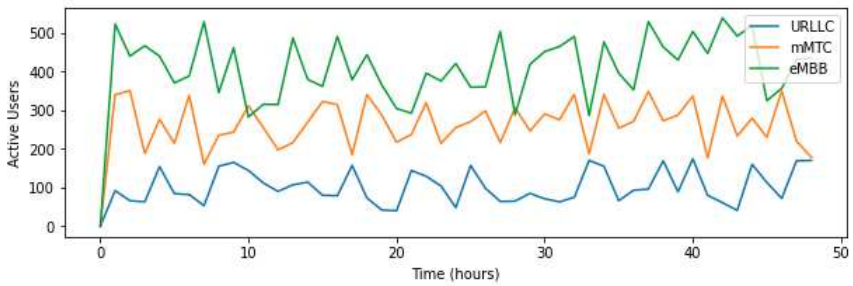


Fig. 5: The observed active user in the network

The DNN model run with the dataset for the long time to improve the overall model accuracy. With the approximate half million connection request traffic has been generated for 48 hours and observed every hours of eMBB, mMTC and URLLC traffic as shown in Fig. 5. The deep learning neural network (DNN) model automatically can learn and understand the device type that allocated to which slice. further over the time DNN will able to predict the connection for device that requires network slice or a specific service.

5 Performance Evaluation and Analysis

In this section, we evaluate our model and discuss the results. We used google co-lab for the simulation which provide 12.72 GB of RAM and 107.77 GB of disk storage. We use epoch-size 50, batch-size 128 and split ratio of training vs testing as 70:30 that means 70% of the dataset to train the model and 30% was used for predicting classifier accuracy. The original dataset has 8 different KPIs as Input features. We describe the KPI in details with the statistics and show the results of slice selection by using deep learning neural network. We applied kNN graph construction algorithm on KPIs with different k values and elaborated 2 non-KPIs features as triangle count and cluster coefficient. Now, the Input features has 8 KPIs and 2 non-KPIs. We evaluated the model with existing and derived input features.

5.1 Prediction based on network KPIs only

The analysis of the dataset based on the network key performance indices is as follows given in the Table 2. The descriptive statistics have been described in summarize shape of the dataset's distribution, central tendency, dispersion. The percentiles we shown in the table as [.25, .5, .75], which returns the 25th, 50th and 75th percentile. We denoted as the first quartile (Q1) for the 25th percentile, quartile (Q2) for the 50th percentile and third quartile (Q3) for the 75th percentile. There are total eight KPIs shown as KPI-I to KPI-VIII with some statistics.

Table 2: network KPIs distribution

Statistics	KPI-I	KPI-II	KPI-III	KPI-IV	KPI-V	KPI-VI	KPI-VII	KPI-VIII
mean	3.41	10.968	1.468	4.00	11.50	1.56	2.16	3.89
std	2.55	6.06	0.498	2	6.92	0.49	0.84	2.29
min	1	1	1	1	0	1	1	1
first quartile (Q1)	1	6	1	2	5.75	1	1	2
second quartile (Q2)	3	11	1	4	11.5	2	2	4
third quartile (Q3)	5	16	2	6	17.25	2	3	6
max	8	22	2	7	23	2	3	7

The formula for *precision*, *recall* and $f1score$ is given in the Equations 11-13 respectively where, TP is the number of true positives, FP is the number of false positives and FN is the number of false negatives. Precision talks about how the model is accurate with how many are actual positive and precision is found to be good measure, when false positive costs are high. Recall calculates the actual positive of the model through labeling as positive and recall is important when the high cost associated with the false negative. F1-score is must needed to seek a good balance between precision and recall.

$$precision = \frac{TP}{TP + FP} \tag{11}$$

$$recall = \frac{TP}{TP + FN} \tag{12}$$

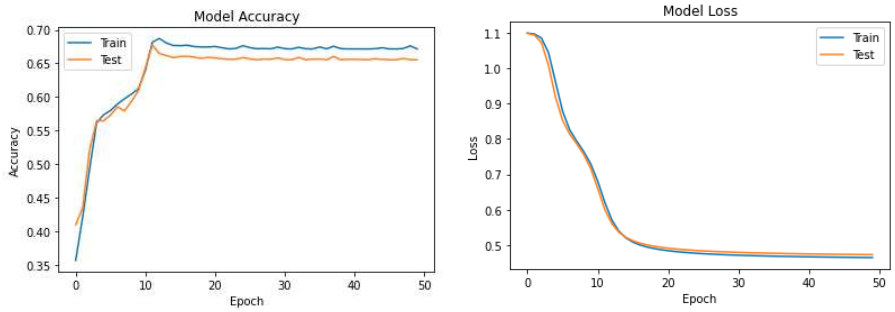
$$F1\ score = 2 \cdot \frac{precision \cdot recall}{precision + recall} = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{13}$$

The performance evaluation with the KPIs only is given in the Table 3 with the *precision*, *recall* and $f1 - score$ parameters corresponding to eMBB, mMTC and URLLC slices. There has been found for eMBB slice the precision, recall and f1-score have good performance measure. For the mMTC and URLLC slice this model with KPIs only does not give good performance metrices.

Table 3: Performance evaluation with KPIs only

Performance matrices	eMBB	mMTC	URLLC
Precision	0.958	0.508	0.487
Recall	0.963	0.504	0.492
F1-Score	0.96	0.506	0.490

The training the model using only the network KPIs to predict slices for new or unknown device has a accuracy of 68.89% . The training and loss function evaluation of the model is shown in the Fig. 6. The success rate and loss function evaluation are two ways to define that models are well trained.



(a) Success Rate Evaluation (b) Loss Function Evaluation

Fig. 6: Performance Evaluation based on KPIs

5.2 Prediction based on network KPIs and non-KPIs features

The analysis of the augmented dataset based on the network non-KPIs as triangle count and cluster coefficient is given in the Table 4. The descriptive statistics have been described in summarize shape of the dataset's distribution, central tendency, dispersion. There are total four k values as $k = 2$, $k = 3$, $k = 4$ and $k = 5$ with non-KPI I and non-KPI II features with some statistics. The dataset have now previous KPIs and new derieved non-KPIs features from this augmented dataset can be applied on the deep learning neural network (DNN) to get more precisely predictions. DNN model have applied for the all four k-values augmented datasets to compare the results.

Table 4: network non-KPIs features distribution

	k=2		k=3		k=4		k=5	
Statistics	non-KPI I	non-KPI II	non-KPI I	non-KPI II	non-KPI I	non-KPI II	non-KPI I	non-KPI II
mean	1.468085	0.626729	3.827	0.629	6.412709	0.594603	9.843892	0.684804
std	1.710995	0.429074	3.51	0.39	4.909001	0.318046	5.272491	0.218393
min	0	0	0	0	0	0	2	0.2
first quartile	1	0.285714	2	0.4	4	0.5	6	0.6
second quartile	1	1	3	0.6	6	0.666667	7	0.6
third quartile	1	1	4	1	6	0.714286	18	0.857143
max	6	1	11	1	15	1	18	1

The performance evaluation with the non-KPIs features for $k = 2$, $k = 3$, $k = 4$ and $k = 5$ are given in the Table 5 with the precision, recall and f1-score parameters corresponding to eMBB, mMTCand URLLC slices. The DNN model has been applied to get prediction analysis on the all four augmented datasets. There has been found for eMBB slice the precision, recall and f1-score have good performance measure. The mMTC and URLLC slices performance metrices have also been improved in terms of *precision*, *recall* and $f1 - score$.

Table 5: Performance evaluation with non-KPIs features

Performance matrices	eMBB	mMTC	URLLC
k=2			
Precision	0.92513	1	0.7
Recall	1	0.57142857	1
F1-score	0.96110912	0.72727273	0.82352941
k=3			
Precision	0.958	0.90909091	0.76470588
Recall	0.96375461	0.71428571	0.92857143
F1-score	0.960868689	0.8	0.83870968
k=4			
Precision	0.91052	0.81818182	0.70588235
Recall	0.97012	0.64285714	0.85714286
F1-score	0.939375598	0.72	0.77419355
k=5			
Precision	1	1	0.7
Recall	0.90065	0.57142857	1
F1-score	0.94772841	0.72727273	0.82352941

The *precision*, *recall* and $f1 - score$ have been shown in the Fig. 7-9 respectively with the eMBB, mMTC and URLLC slices. The augmented DNN model with also non-KPIs features have given better results for $k = 3$ and $k = 4$ as compare to DNN model with KPIs only. For $k = 2$ and $k = 5$, the precision and recall is compromised and gives sometimes value one but for f1-score both are good. For k=3 and k=4, overall found to be better model and it gives better overall accuracy but, out of k=3 is the best model for the augmented dataset. Our training dataset included KPIs and non-KPIs input features to determine the service requests and to allocate the appropriate slice. This needs to be very essential since there are diverse range of devices like industry 4.0, mobile devices, self-driving cars, AR devices etc. are connected to get an appropriate slice. The AR device, industry automation, self-driving cars and healthcare required very low latency approx to 1ms and high reliability whereas eMBB, mMTC can be somewhat compromise with the latency but should be below 10ms.

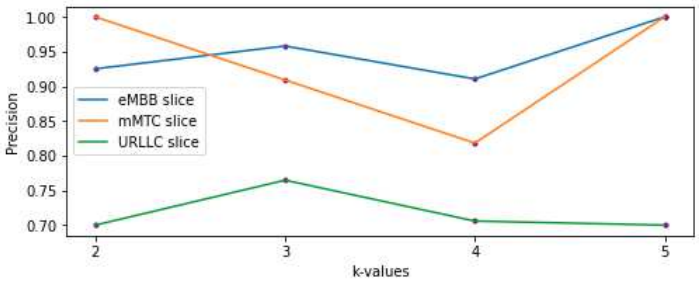


Fig. 7: Precision

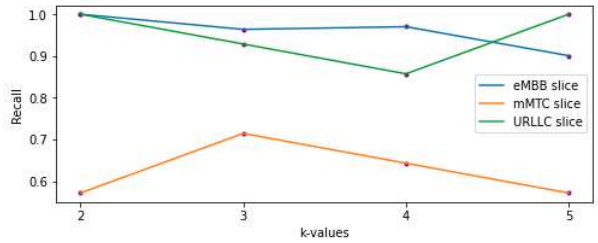


Fig. 8: Recall

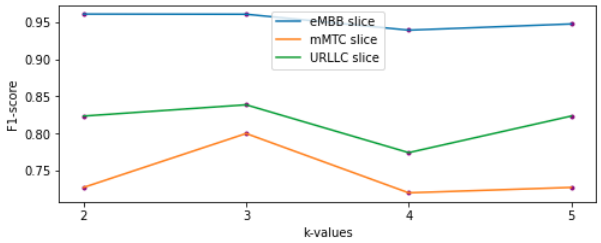
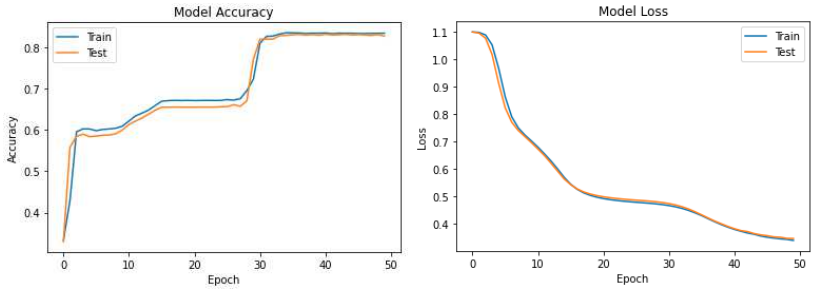


Fig. 9: F1-score

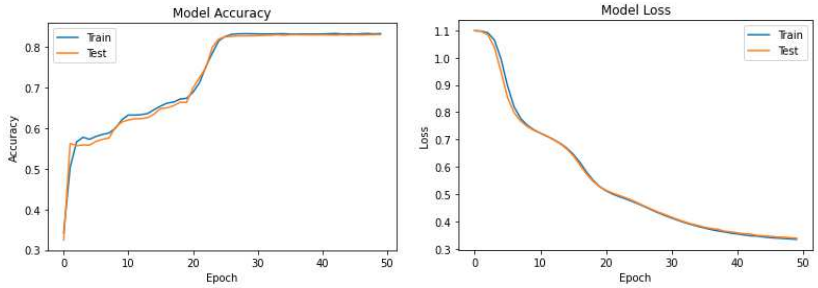
The accuracy of the model are as 86.67% for $k = 2$, 88.89% for $k = 3$, 84.44% for $k = 4$ and 86.67% for $k = 5$. The training and loss function evaluations of the model for $k = 2$, $k = 3$, $k = 4$ and $k = 5$ is shown in the Fig. 10. The overall accuracy is found to be good after the using of non-KPIs features by kNN graph construction with triangle count and cluster coefficient. This non-KPIs features improve more to the URLLC slice because of very-low latency and high throughput requirements whereas the eMBB and mMTC slice have also more impact for the peak data rate and high connection density requirements.

From this experiments, we get that including the augmented features along with the network KPIs improves the model performance significantly. As we increase the value of k (nearest neighbours), the values of triangle count and clustering coefficient increase because increasing the neighbours means increasing the connectedness between the device requests. For lower k values e.g. $k = 1$ both the features are 0 because of very less connectivity. Similarly for higher k values e.g. K = 10, the features tend to skew towards 1. Therefore the optimal value for k is $k = 3$ or $k = 4$ when the dataset is not under/over-connected with around 29% improved in accuracy. This is also reflected in the model performance as the evaluation parameters as precision, recall and f1-scores for all slices are better for $k = 3$ and $k = 4$ whereas the model performs poorly for mMTC and URLLC slices and is overfitting the eMBB slice.



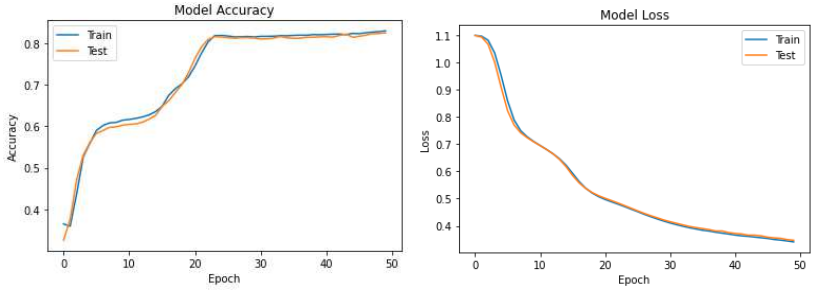
(a) Accuracy for k=2

(b) Loss for k=2



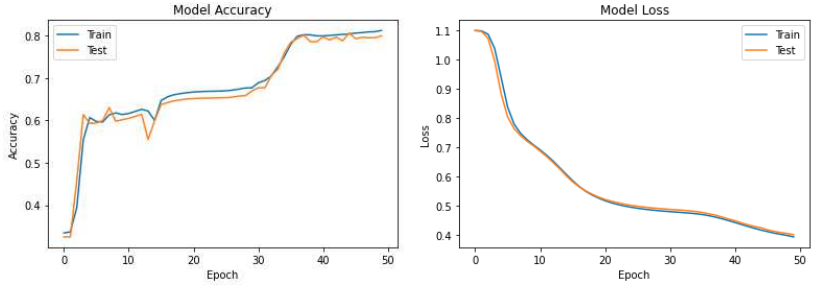
(c) Accuracy for k=3

(d) Loss for k=3



(e) Accuracy for k=4

(f) Loss for k=4



(g) Accuracy for k=5

(h) Loss for k=5

Fig. 10: Success rate vs loss function evaluation for different k values

6 Conclusions

The Network slicing paradigm in 5G play the key role in the next generation wireless networks, vertical industries, mobile operators. In this paper, we demonstrated k-Nearest Neighbour Graph Construction algorithm in 5G Network Slicing KPIs and derived non-KPIs as triangle count and cluster coefficient. We use deep learning neural network (DNN) models to predict the slice for the incoming traffic with KPIs and non-KPIs. Our results shows that at k=3, k=4 of kNN graph construction gives overall better accuracy. The deep learning model help to analyze the overall traffic patterns and canbe able to predict the future traffic.

7 Declarations

7.1 Funding

Not applicable

7.2 Conflicts of interest/Competing interests

The authors declare that they have no competing interests.

7.3 Availability of data and material

The datasets generated during and/or analysed during the current study are available in the adtmv7/DeepSlice repository at github.com/adtmv7/DeepSlice/tree/master/Source.

7.4 Code availability

software application or custom code could be available whenever will be required.

References

1. GSMA,"An Introduction to Network Slicing," 2020.
2. Khan, Latif U., et al. "Network Slicing: Recent Advances, Taxonomy, Requirements, and Open Research Challenges." IEEE Access 8 (2020): 36009-36028.
3. Yang, Song, et al. "Recent Advances of Resource Allocation in Network Function Virtualization." IEEE Transactions on Parallel and Distributed Systems 32.2 (2020): 295-314.
4. Sun, Gang, et al. "Energy-efficient provisioning for service function chains to support delay-sensitive applications in network function virtualization." IEEE Internet of Things Journal (2020).

5. Dogra, Anutusha, Rakesh Kumar Jha, and Shubha Jain. "A Survey on beyond 5G network with the advent of 6G: Architecture and Emerging Technologies." IEEE Access (2020).
6. Messaoudi, Farouk, Philippe Bertin, and Adlen Ksentini. "Towards the quest for 5G Network Slicing." 2020 IEEE 17th Annual Consumer Communications Networking Conference (CCNC). IEEE, 2020.
7. Barakabitze, Alcardo Alex, et al. "5G network slicing using SDN and NFV: A survey of taxonomy, architectures and future challenges." Computer Networks 167 (2020): 106984.
8. Han, Bin, and Hans D. Schotten. "Machine Learning for Network Slicing Resource Management: A Comprehensive Survey." arXiv preprint arXiv:2001.07974 (2020).
9. Ojijo, Mourice O., and Olabisi E. Falowo. "A Survey on Slice Admission Control Strategies and Optimization Schemes in 5G Network." IEEE Access 8 (2020): 14977-14990.
10. Han, Bin, et al. "A utility-driven multi-queue admission control solution for network slicing." IEEE INFOCOM 2019-IEEE Conference on Computer Communications. IEEE, 2019.
11. Agarwal, Satyam, et al. "VNF placement and resource allocation for the support of vertical services in 5G networks." IEEE/ACM Transactions on Networking 27.1 (2019): 433-446.
12. Cohen, Rami, et al. "Near optimal placement of virtual network functions." 2015 IEEE Conference on Computer Communications (INFOCOM). IEEE, 2015.
13. R. K. Gupta and R. Misra, "Machine Learning-based Slice allocation Algorithms in 5G Networks," 2019 International Conference on Advances in Computing, Communication and Control (ICAC3), Mumbai, India, 2019, pp. 1-4, doi: 10.1109/ICAC347590.2019.9036741.
14. Thantharate, Anurag, et al. "Deepslice: A deep learning approach towards an efficient and reliable network slicing in 5G networks." 2019 IEEE 10th Annual Ubiquitous Computing, Electronics Mobile Communication Conference (UEMCON). IEEE, 2019.
15. Wang, Jing, et al. "Scalable k-nn graph construction for visual descriptors." 2012 IEEE Conference on Computer Vision and Pattern Recognition. IEEE, 2012.
16. Park, Youngki, et al. "Greedy filtering: A scalable algorithm for k-nearest neighbor graph construction." International Conference on Database Systems for Advanced Applications. Springer, Cham, 2014.
17. Dong, Wei, Charikar Moses, and Kai Li. "Efficient k-nearest neighbor graph construction for generic similarity measures." Proceedings of the 20th international conference on World wide web. 2011.
18. Zhang, Yabin, et al. "Label propagation with augmented anchors: A simple semi-supervised learning baseline for unsupervised domain adaptation." European Conference on Computer Vision. Springer, Cham, 2020.
19. Prokhorenkova, Liudmila, and Aleksandr Shekhovtsov. "Graph-based nearest neighbor search: From practice to theory." International Conference on Machine Learning. PMLR, 2020.
20. Saul, Lawrence K. "A tractable latent variable model for nonlinear dimensionality reduction." Proceedings of the National Academy of Sciences 117.27 (2020): 15403-15408.
21. Navarro-Ortiz, Jorge, et al. "A survey on 5G usage scenarios and traffic models." IEEE Communications Surveys Tutorials 22.2 (2020): 905-929.
22. Kammoun, Amal, et al. "Admission control algorithm for network slicing management in SDN-NFV environment." 2018 6th International Conference on Multimedia Computing and Systems (ICMCS). IEEE, 2018.
23. Wu, Bin, Ke Yi, and Zhenguo Li. "Counting triangles in large graphs by random sampling." IEEE Transactions on Knowledge and Data Engineering 28.8 (2016): 2013-2026.
24. Bhatia, Siddharth. "Approximate Triangle Count and Clustering Coefficient." Proceedings of the 2018 International Conference on Management of Data. 2018.

Figures

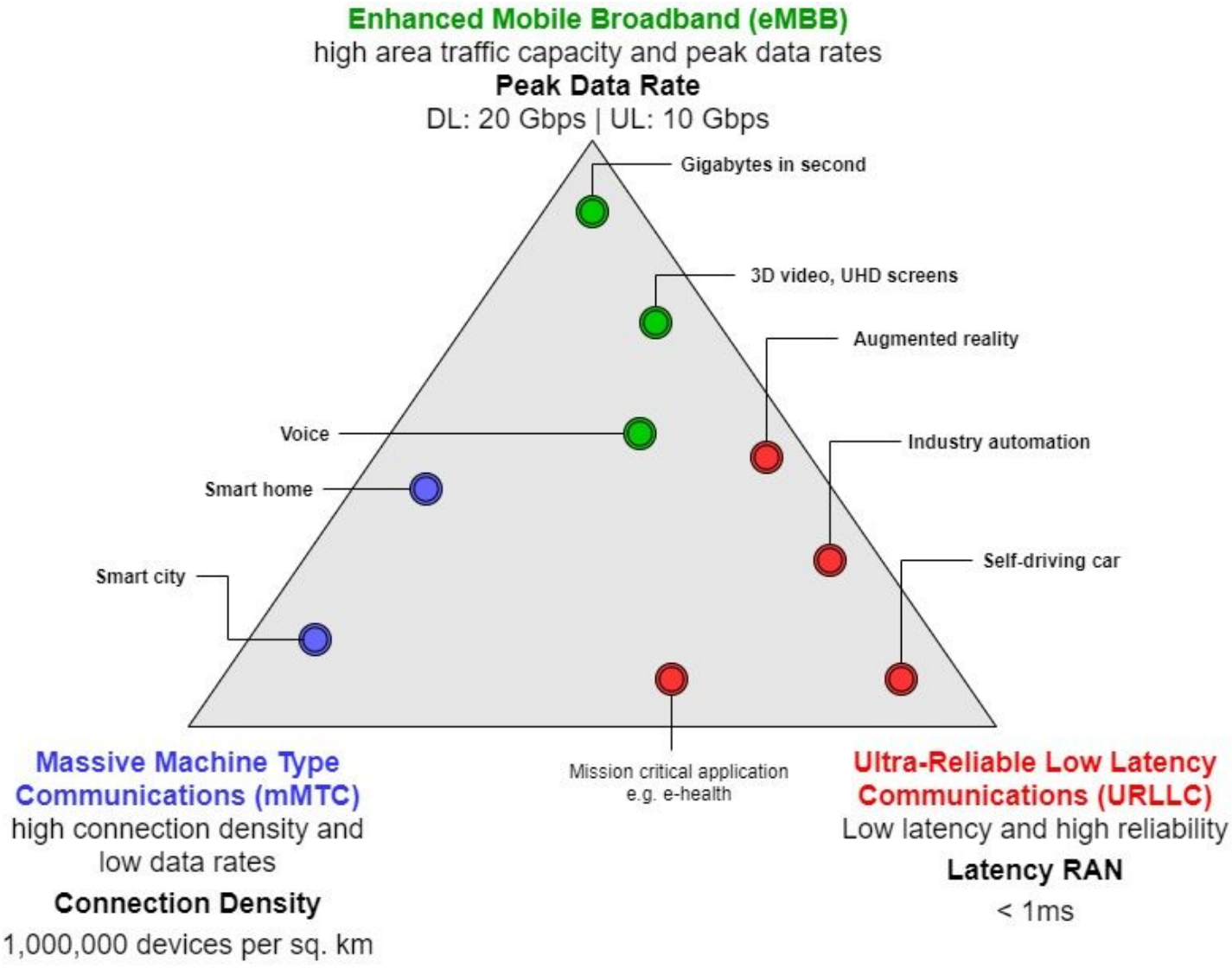


Figure 1

ITU usage scenario for 5G

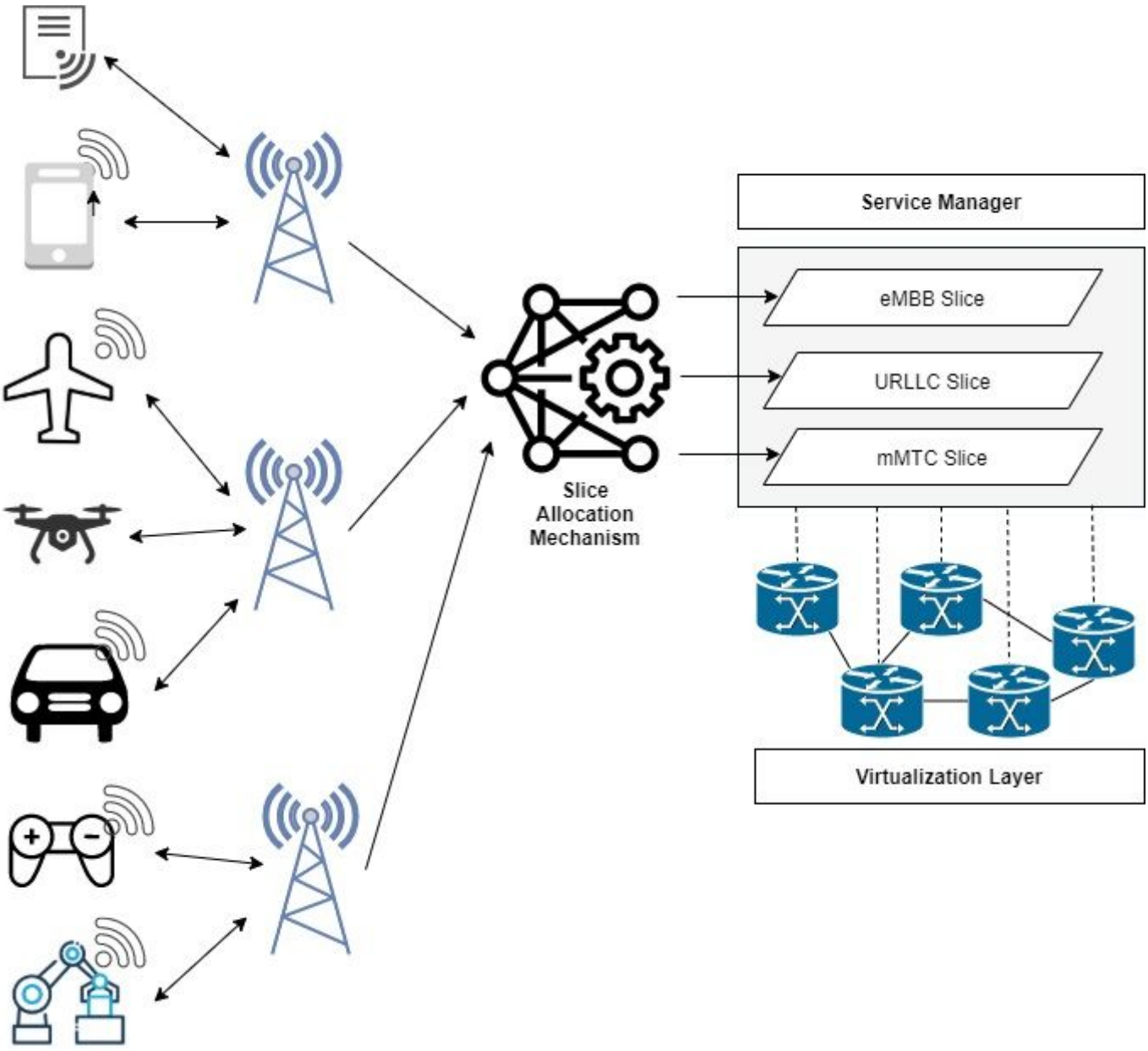


Figure 2

Overview of Network Slice architecture

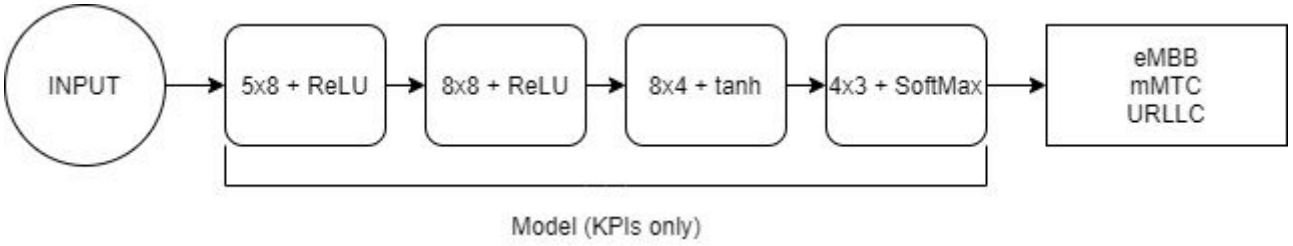


Figure 3

DNN Model Overview

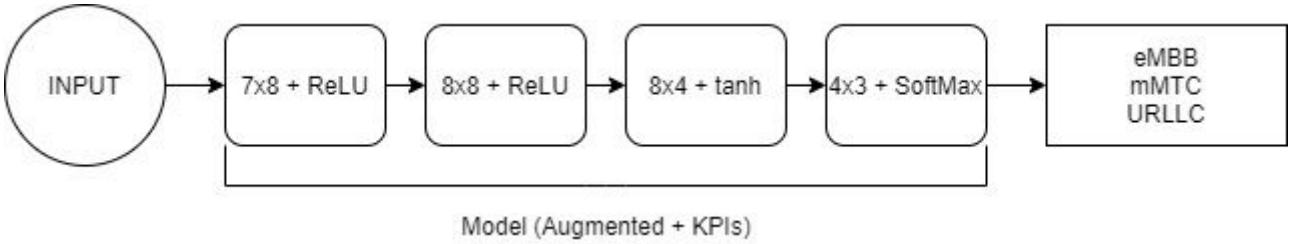


Figure 4

Augmented DNN Model Overview

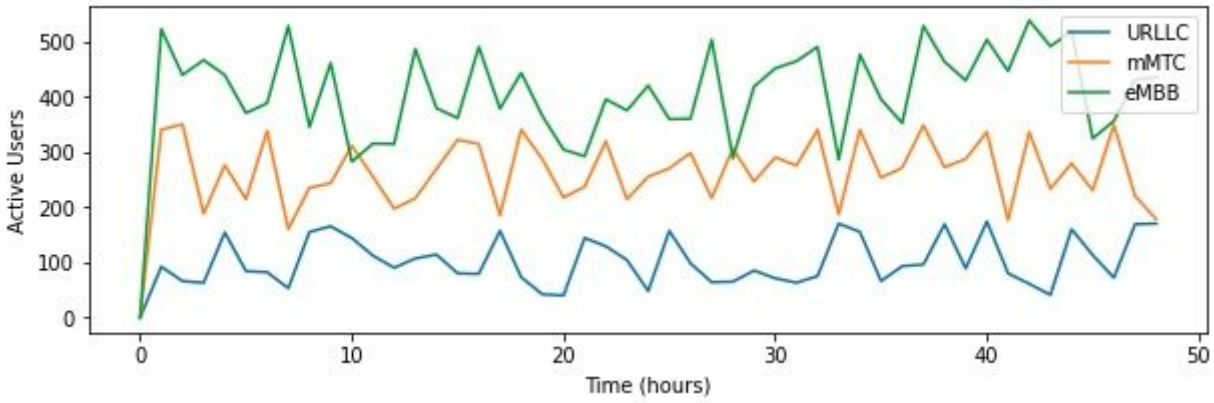
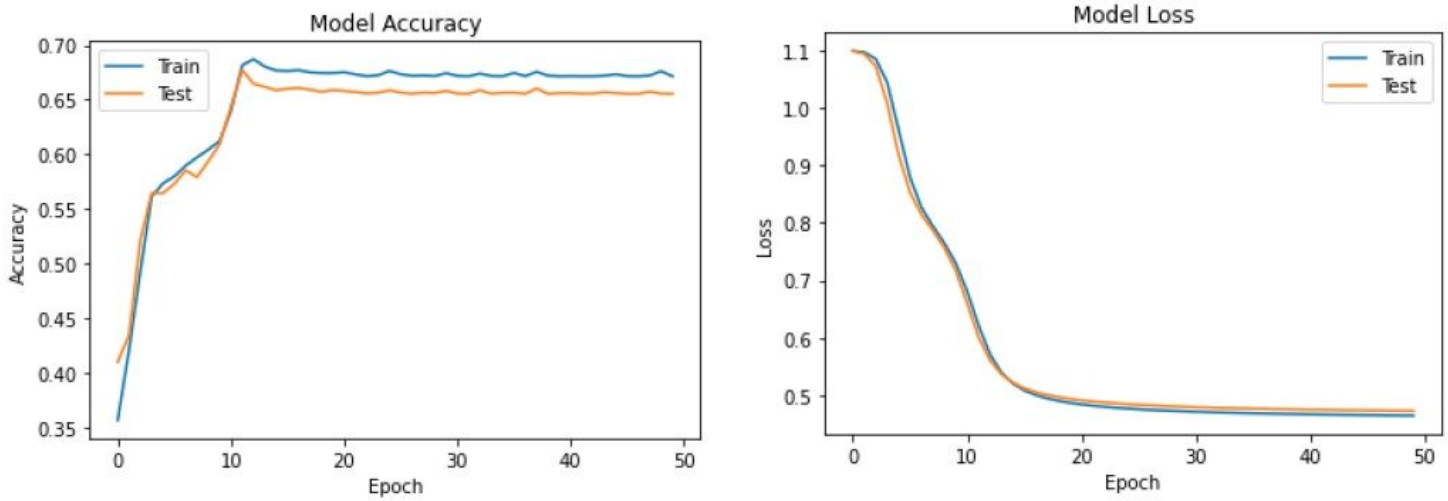


Figure 5

The observed active user in the network



(a) Success Rate Evaluation (b) Loss Function Evaluation

Figure 6

Performance Evaluation based on KPIs

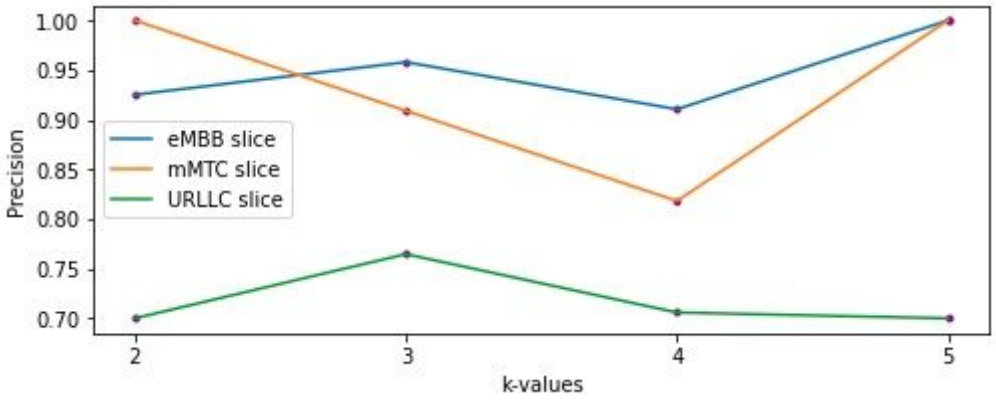


Figure 7

Precision

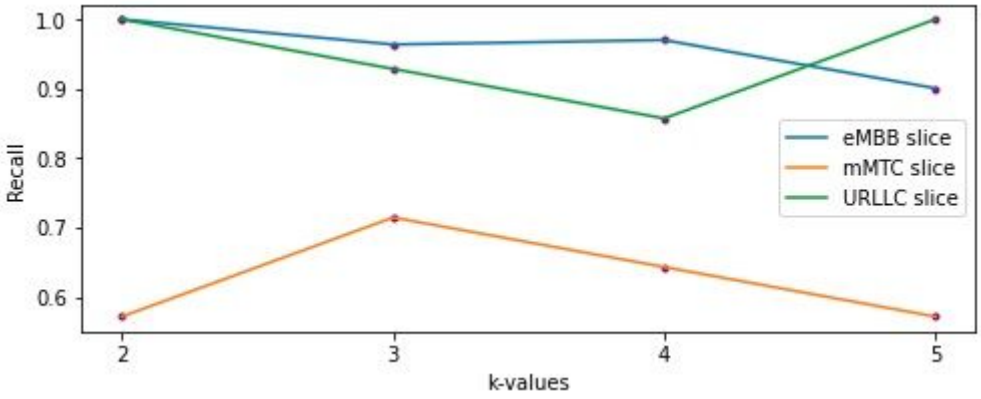


Figure 8

Recall

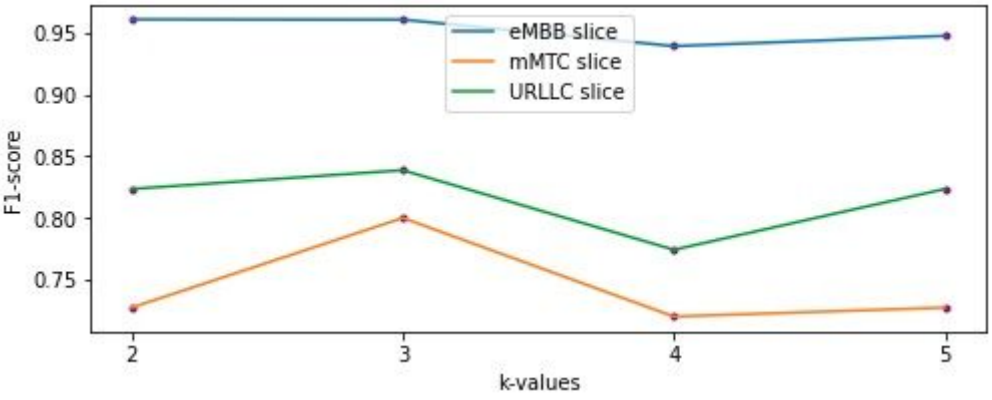


Figure 9

F1-score

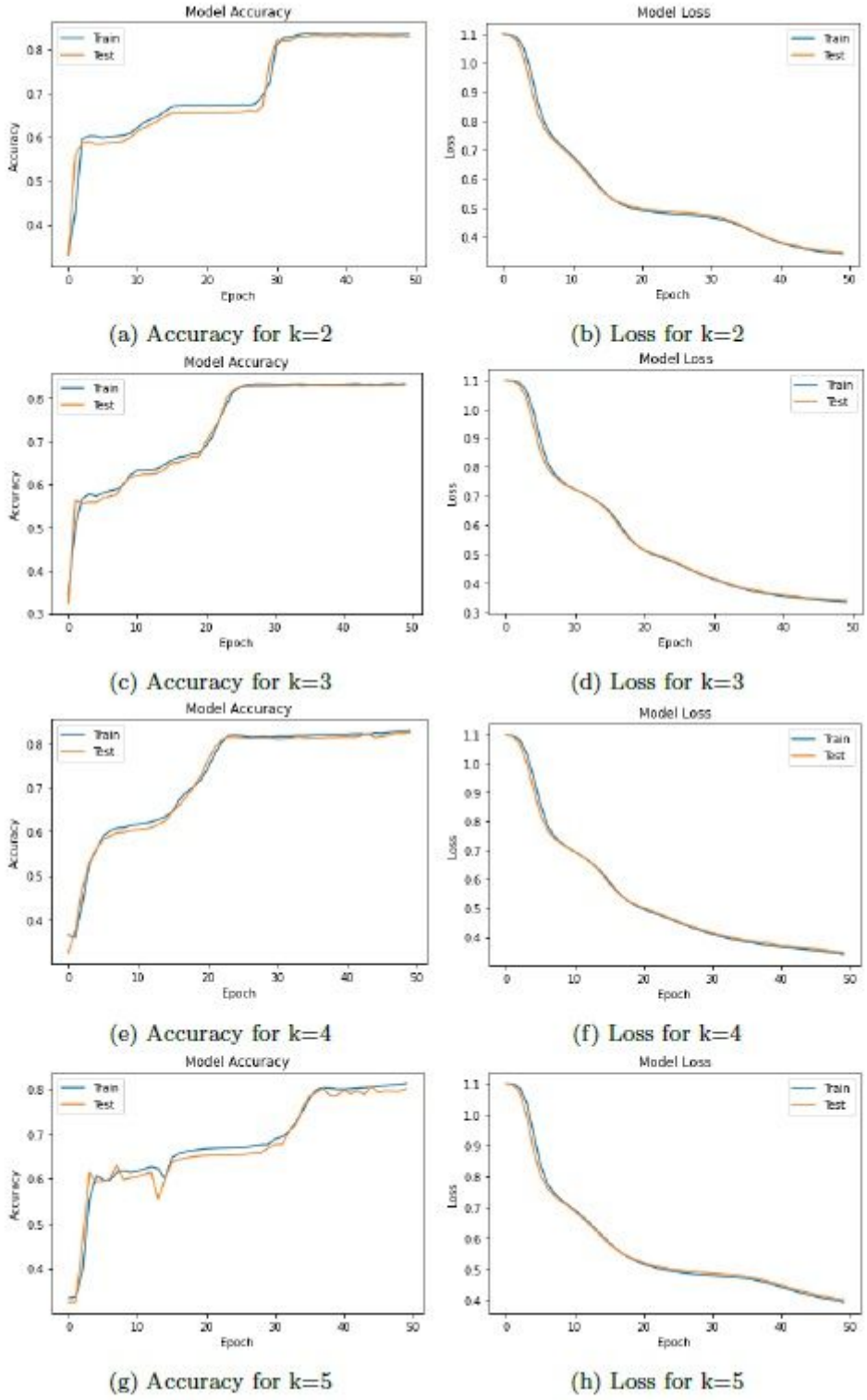


Figure 10

Success rate vs loss function evaluation for different k values