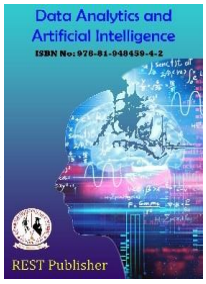
Data Analytics and Artificial Intelligence
Vol: 3(7), 2023
REST Publisher; ISBN: 978-81-948459-4-2
Website: http://restpublisher.com/book-series/daai/



Classification Of Intelligent Transportation System (Its) From Generated and Real Traffic Data

*** E. Kalaivanan, S. Brindha**
SPIHER, Chennai , Tamilnadu ,India
*Corresponding Author Email: Chennaikalaivanan.research@gmail.com

Abstract: *In this paper, we use flow-based data to evaluate the traffic condition in ITS, however, the data required for evaluation is very minimal. The study hence uses a synthetic data generation via Generative Adversarial Network (GAN) to generate more of flow-based traffic data in ITS. The study then uses Convolutional Neural Network to classify the generated and real samples in an unsupervised manner. The simulation is conducted to test the efficacy of the model. The results show that the proposed CNN model predicts well the instances and classifies well the traffic than the existing methods.*
Keywords: *Real Network Traffic, ITS, Generative Adversarial Network, Convolution Neural Network*

1. INTRODUCTION

Traffic classification has many uses [1], such as in network monitoring, quality of service management, and security, to name just a few. The traffic flows can be categorized according to the type of application, which can be used to establish a framework for service differentiation (such as streaming or Voice over IP) [2]. As a result, applications can specify their own bandwidth and latency needs, which can subsequently be utilized to allocate resources in a way that ensures QoS. The TCP/IP header is not required to be changed in order to classify traffic in any of the numerous ways possible. Various method [3] classifies the traffic into useful subsets based on the ports used. However, modern software uses dynamic ports and tunneling, making this approach obsolete. However, deep packet inspection organizes information based on the contents of individual packets, which are then compared to a database of previously recognized signature. However, this approach is not without its drawbacks [4], including privacy concerns, incompatibility with encrypted content, and high processing costs.Traffic categorization methods grounded in machine learning have received a lot of study due to the promise of increased efficiency and accuracy. The process of switching to a supervised learning approach in the field of machine learning is complex and involves a number of distinct phases. To begin, we must determine which aspects of the traffic can stand in for the flow actual characteristics (such as the length of a packet, for example). The next thing to do is to build the machine learning model. The third step is to provide the classifier with a list of predefined traffic types and teach it to draw links between those types and the features it has been given. At the end of the day, the model is put to use to classify existing data traffic and forecast future data traffic kinds [5].When training a machine learning system through unsupervised methods, the data used to do so must be unlabeled. Therefore, it is impossible to foretell the outcomes of a particular sample. However, unsupervised learning seeks to classify samples according to the similarity or statistical correlations between features without any external supervision [6]. In contrast to supervised learning, which requires an instructor oversight, unsupervised learning is done independently. If the data points have a regular pattern, we can classify them into clusters. While the model excels at seeing and labeling patterns, it has no way of knowing whether or not the classifications it makes are accurate. New groups can be created using unsupervised learning algorithms [7], such as k-means clustering and self-organizing maps.In this paper, we use flow-based data to evaluate the traffic condition in ITS, however, the data required for evaluation is very minimal. The study hence uses a synthetic data generation via Generative Adversarial Network (GAN) to generate more of flow-based traffic data in ITS. The study then uses Convolutional Neural Network to classify the generated and real samples in an unsupervised manner

2. RELATED WORKS

In this section, the study examines several published examples of related research and evaluates the differences and similarities between the approaches taken by different authors [8]-[9]. Others have explored QoS support for smart city applications across many levels or employed machine learning for traffic classification (for example, the data link layer and the transport layer). However, others have demonstrated a comparison of how machine learning may be used to classify traffic across various datasets (including the backbone network) [9].
In order to identify traffic patterns, they collected data and created features to feed into a Support Vector Machine. Then, the computer was fed the data and features. Accuracy rates of 99.31% while utilizing biased test samples and 96.12% when using unbiased test samples attest to the trustworthiness and precision of the SVM output. While accuracy in algorithm performance is important, the authors focus solely on the support vector machine (SVM) to the exclusion of all other machine learning techniques. The impact of latency is far larger than the impact of accuracy in real-time applications. This highlights the need of considering the time required to execute various machine learning algorithms [10].learning-based Incoming Internet Protocol (IP) packets can be automatically classified by type thanks to Differentiated Services. They utilized machine learning strategies, thinking about packet features including the disproportional distribution of traffic between classes (e.g., linear discriminate analysis, k-means clustering) [11]. The proposed technique dynamically altered the classification outcomes. Both our approach and the author aim to categorize traffic. On the other hand, our method generates a larger number of subclasses from the Differentiated Services labels than the semi supervised method used by the authors. However, we use a different approach, classifying network traffic into 11 distinct types via four supervised machine learning techniques. Smart city applications can benefit from the cloud robotics framework developed by [12]. To boost service quality and system performance, a robotic agent within the framework can utilize cloud services by shifting work to remote servers. The optimization issue is posed in the form of a directed acyclic network optimization problem, which is then solved with a genetic algorithm. After determining the best offloading strategies, the optimization issue is resolved. To counteract this advancement, we implement machine learning-based traffic classification to boost QoS in smart city networks. Because of this, we can arrange our network resources more efficiently.

2.1 Proposed Method: Figure 1 depicts the four-step process we use to classify traffic using machine learning. Data collection and feature selection, followed by pre-processing, DL modelling, and finally visualization of results. A dataset is built utilizing samples of actual traffic flows after extensive data collection and the identification of key aspects. After filtering out any data that is not statistically significant, the samples are then assigned a label indicating to which group they belong. By using a standardization technique on the data during pre-processing, the features take on a more uniform appearance. Next, the DL algorithms utilize the training dataset to establish the model for traffic classification. Next, the algorithms are put to the test by being run on the test dataset. However, given the dearth of real-world datasets, we use GAN to generate generated data samples for the sake of evaluation. The evaluation is subsequently performed using these fabricated data samples.

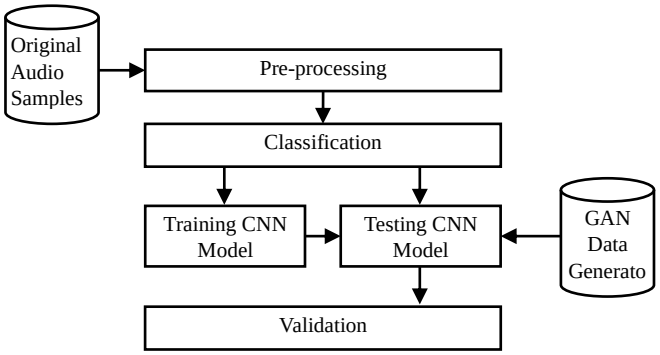


FIGURE 1. Proposed Model

2.1 GAN Traffic Data Generation: For their ability to classify incoming data, discriminative models are commonly used in intrusion detection. Generic models, as opposed to discriminative ones, are used extensively when generating new data. As a result, many generative models depend critically on the maximization of the likelihood for a parametric probability distribution. However, other models get around this issue by never even

considering a likelihood representation. General adversarial networks (GANs) use a game-theoretic technique to estimate the data distribution in place of Markov chains. Differentiating real data from fake data is the task of the discriminator network (denoted by the letter D), while the generator network (denoted by the letter G) works to recreate samples of data distribution. Both networks are trained in a recursive manner until the discriminator D can no longer tell the difference between the genuine and fake input. Not only does the generator G save down on computation time, but it also never needs to be refreshed with new samples. Instead, the generator network G takes in noise in the form of a z vector as an input. Back propagation is used to train the generator using only the gradients from the discriminator. This reduces the likelihood that the generator G will be over fit to a particular data set through the process of learning and recreating actual data.

2.2 Generator:

$$\min_{G}\max_{D\in L}\underset{x\sim P_r}{E}\left[\log\left(D(x)\right)\right]+\underset{z\sim P_f}{E}\left[\log\left(1-D\left(G(z)\right)\right)\right]$$

2.3 Discriminator:

$$\min_{G}\max_{D\in L}\underset{x\sim P_r}{E}\left[D(x)\right]-\underset{z\sim P_f}{E}\left[D\left(G(z)\right)\right]$$

where

P_r - real distribution of data and

P_f - generated distribution of data.

$D(x)$ - x probability that shows as a real data, and

z - random noise.

L - 1-Lipschitz function set to restrict D.

We begin by using a GAN structure that is highly similar to the one shown on the left side of Figure 1 to construct our generative model. The generator takes as input a vector selected at random from a normal distribution. The generator creates an FBANK feature-laden feature map as its final product. The discriminator is educated to tell the difference between the generator fabricated feature maps and the real ones that were extracted from the noisy source data. Expression 2 details the WGAN framework we utilized; with this, we applied the loss function and parameters we had customized. Previous studies on GAN hypothesize that the model performance heavily depends on the structural configuration and training variables. Our architecture, shown in Figure 2, employs a simple GAN in the manner shown. The discriminator employs a series of 3 CL followed by 2 FCL to determine which data points are authentic and which are not. The discriminator uses 2 FCL for transferring the input random noise, whereas the generator uses 3 transposed CL to build the feature maps. When conducting experiments, we employ the Leaky ReLU not only as a classifier but also as a generator. In this model, the negative slope is fixed at a value of 0.2.From Figure 2, it is sees that the proposed generator and discriminator, which is built on the GAN framework. For example, there is the convolutional layer (CL), the transposed convolution layer (ConvT), the fully connected layer (FC), and the batch normalization layer (BN).

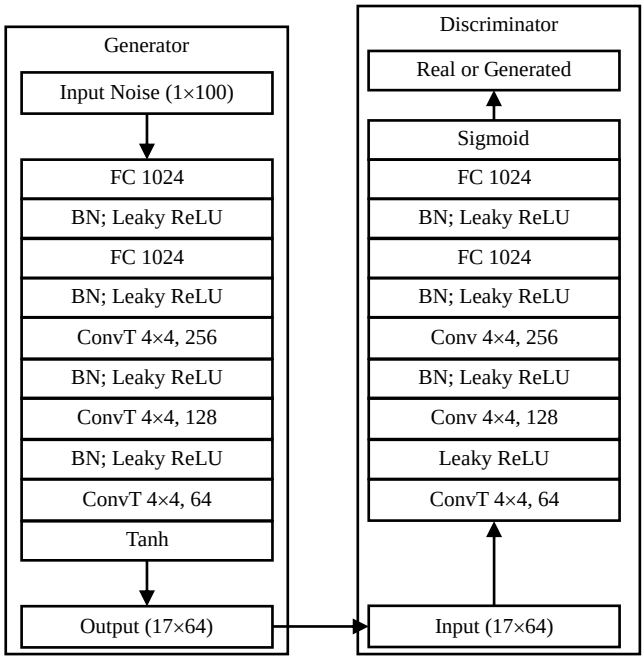


FIGURE 2.The architecture of the GAN; Conv – CL,ConvTrans - transposed CL,BN - batch normalization, andFC – FCL.

The entire model of GAN with generator and discriminator is configured as [4×4, 64] and this represents that the layer functions with a filter of size 4×4 with a 64 feature maps output.

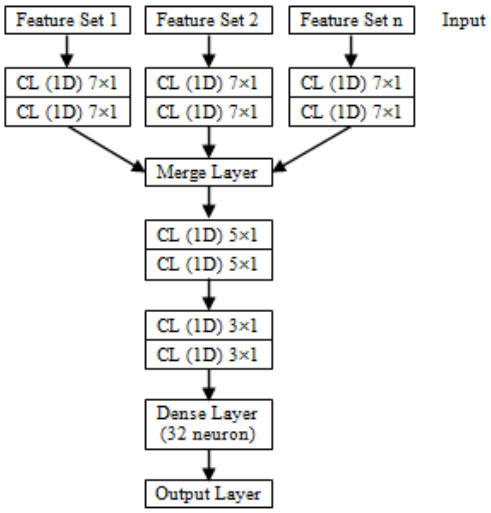


FIGURE 3. Traffic Feature Classification using CNN classifier

The inputs into the neural network are the features that have been extracted from the flow-based network traffic. The input vocabulary is the set of all IP addresses, all destination ports, and all transport protocols that are present in the flow-based data set. The flow-based data set serves as the de facto standard for this terminology. To avoid neural network inability to process categorical data, we can instead feed them continuous data.A lexical value has been specified for each input and output neuron. To simplify, let say the training data set has 100,000 distinct IP addresses, 20,000 distinct destination ports, and 3 distinct transport protocols. Therefore, there is just one hot component in the one-hot vector, giving it a size of 120,003. The neural network output layer employs a softmax classifier to display the likelihood of each vocabulary value being in the same flow

(context) as the input value. The similarity between the two numbers is an indication of the likelihood of a match. To guarantee that the aggregate of all neuronal outputs adds up to 1, the softmax classifier normalizes each individual output. Input neurons outnumber output neurons by a significant margin.

2.2 CNN Classification: Each network serves as a classification framework, with weights determined using the back propagation learning method. The hyper-parameter settings for the CNN classifiers are used for training the classifier using the KERAS software. There are a number of techniques that can be used during the training phase of classification to limit the likelihood of over-fitting. When it comes to preventing over-fitting, the L2 regularize was the first to the punch penalize. It does this by incorporating the squared magnitude of exceptionally large weights as a penalty in the objective function. The approach prevents over-fitting by imposing a penalty on extremely high weights. As an alternative approach, dropout can be used to deal with over fitting. Dropout is a learning mechanism that limits a neuron continued activity during training to a threshold probability, *p*, below which it is deactivated. Within a larger neural network, dropout can be thought of as a type of sampling the neural network, with weight updates occurring only in the portion of the network that was actually sampled. Within a bigger neural network, dropout can be viewed as a form of pruning.The proposed models use a different strategy for representing input data than the current CNN designs. Due to the fact that CNN takes into account the spatial link between surrounding features, the features are reordered in the input data to find the associations between them. We calculate model-level and feature-level feature correlations across all experiments. Following this, the characteristics are clustered using hierarchical agglomerative clustering, which divides them into subgroups based on the correlation values between features. The input features have been rearranged in a way that corresponds to the clustered feature order in the dendrograms. The information we take in is rearranged, not in a random fashion but rather so that the most highly correlated features are closer together.

2.3 Traffic Classification: The entire neural network development and training procedure was conducted in Tensor Flow 2. For optimal performance, we used the VGG19 pre-trained neural network in both our preliminary pre-train transfer learning and our subsequent model training. The VGG19 is one of the most well-known CNNs as of late due to its powerful architecture and widespread use. The CNN output layer was removed and two additional dense layers with 64×32 completely connected units were added for the purposes of transfer and retraining. This layering structure also included batch normalization at each level. A sigmoid activation and a single output unit are used to build the new output layer. To the input of the pertained VGG19, a layer of a 2D CNN was appended. This company uses a 128×32×1 input layer in its transfer learning model. To make the pre-trained CNN more appropriate for the new speech dataset, we disabled training on all except the last five layers of the modified VGG19. The effectiveness of the trained neural network was evaluated by three rounds of cross validation. Each fold of the training set contains either 78 or 38 samples. We selected a loss function based on the cross entropy of the two classes because we were dealing with a problem that required binary classification. To get the most out of the gradient descent, we employed ADAM with a learning rate of 0.01. Training parameters included a maximum of 600 training epochs, a batch size of 32, randomly selecting 20% of training samples for use in validation, and tracking validation set area under the curve performance metric.

2.4 Training: Data from network traffic flows are used to train the neural network. IP2Vec only uses the flow source IP address, destination IP address, destination port, and transport protocol. The training sample generation procedure is shown in Figure 4. [Insert citation here] IP2Vec creates five training samples, one for each flow, to feed into its neural network for the purpose of learning. Each sample used for training includes both the input values and the predicted output values. Before doing anything else, IP2Vec picks an input value to use as a training sample. Projected output values are displayed in black boxes on a white background. This set of training examples was constructed using the same input value throughout.In training, the input value is used to determine how the neural network will predict the outcomes of the remaining values in the lexicon. Out of ten training samples, you're guaranteed to get one with a certain output value, while the other nine have a zero chance of occurring. The output layer usually gives some indicator of the likelihood that one input value would appear in the same flow as another input value. Back-propagation is used to train the network. Training in this manner, while effective, might take a lot of time. Let pretend that there are 32 neurons in the hidden layer and that the training data set has a million unique IP addresses and ports for the sake of this example. Thus, 32 million weights are produced for each level of the network. That why it'll take a long time to train such a massive neural network. In addition, it takes a lot of practice to keep good form while adjusting many weights at once. Because of this, we'll have to tweak the values of millions of weights that are connected to millions of training-set observations. IP2Vec employs a method called Negative Sampling to learn faster, which is quite similar to what Word2Vec does. The weights are adjusted gradually rather than all at once after each training sample when utilizing the negative sampling method. To learn more about the technique of negative sampling, please visit the following link.

3. RESULTS AND DISCUSSIONS

The publicly accessible CIDDS-001 data set is used for this. Network topology, IP addresses, and one-way traffic patterns are all included in this dataset. The CIDDS-001 data set includes four internal subnets: a development subnet (dev) with only Linux clients, an office subnet (off) with only Windows clients, a management subnet (mgt) with mixed clients, and a server subnet (srv) with only other servers. IP address ranges serve as identifiers for each of these subnets (srv). It easier to figure out what to do with the obtained data when additional specialized knowledge is added to the mix. In our study, we construct a generative model that can generate new flows whose characteristics are determined by the empirical probability distribution of the input data. The baseline uses simple counting methods applied to the incoming data to make inferences about the probability distribution for each attribute. New flows are generated by inferring from the empirical probability distributions and then responding to those flows. The time taken to complete the transfer, the protocol used, the IP addresses of both the sender and receiver, the total number of bytes transferred, the number of packets sent, and the TCP flags are all obtained independently. At the end of the process, we calculate the algorithm F1-score, accuracy, precision, and recall to see how well it performed (Figure 4-7). we employ k-fold cross-validation to evaluate the efficacy of our classifiers.

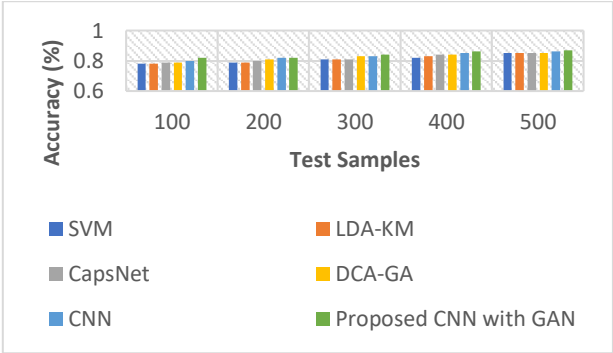


FIGURE 4. Accuracy

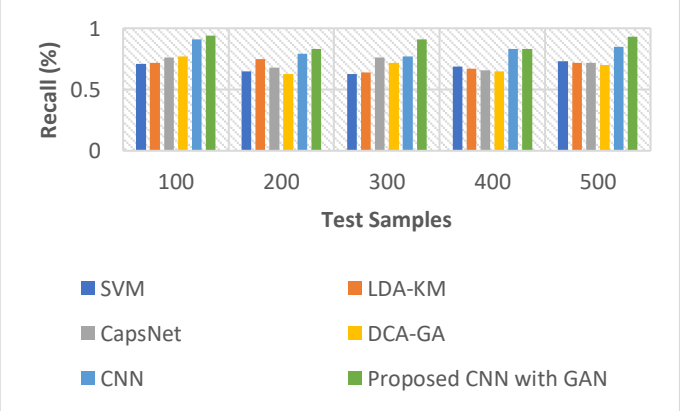


FIGURE 5. Recall

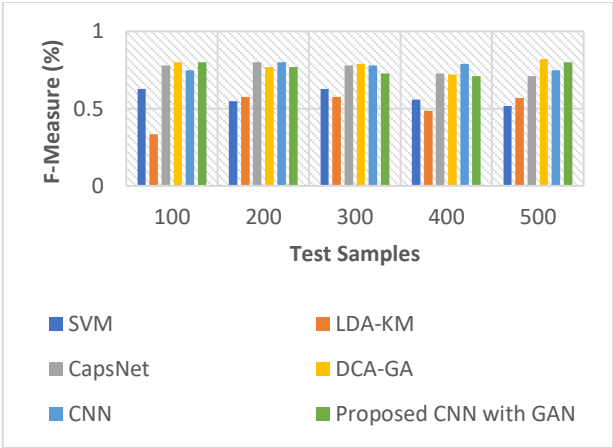


FIGURE 6. F-Measure

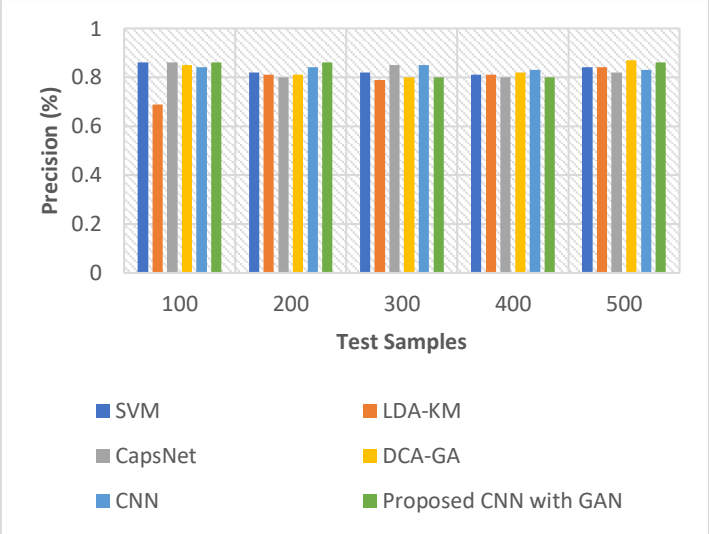


FIGURE 7. Precision

4. CONCLUSIONS

Similarly, it is difficult to leverage actual network traffic because of a lack of ground truth. Manually identifying actual network traffic can be challenging even for skilled people due to the sheer volume of data involved (millions to billions of flows). While this study does utilize flow-based data to examine the ITS traffic condition, the amount of data really required is fairly little. Therefore, in this study, a GAN is used to produce synthetic data for ITS, with a focus on flow-based traffic data. CNN automatically classifies both the synthetic and natural samples. Simulating the system in question allows researchers to see if it is feasible to put the concept into practice.

REFERENCES

[1]. Anusha, M., & Kiruthika, P. (2022). A Comparative Study on Augmented Analytics Using Deep Learning Techniques. In Ubiquitous Intelligent Systems (pp. 135-142). Springer, Singapore.

[2]. Zhang, Y., Wang, S., Chen, B., Cao, J., & Huang, Z. (2019). Trafficgan: Network-scale deep traffic prediction with generative adversarial nets. IEEE Transactions on Intelligent Transportation Systems, 22(1), 219-230.

[3]. Sarker, I. H. (2021). Deep learning: a comprehensive overview on techniques, taxonomy, applications and research directions. SN Computer Science, 2(6), 1-20.

[4]. Ravindran, U., & Gunavathi, C. (2022). A survey on gene expression data analysis using deep learning methods for cancer diagnosis. Progress in Biophysics and Molecular Biology.
[5]. Piccialli, F., Di Somma, V., Giampaolo, F., Cuomo, S., & Fortino, G. (2021). A survey on deep learning in medicine: Why, how and when?. Information Fusion, 66, 111-137.
[6]. Soother, D. K., Daudpoto, J., Harris, N. R., Hussain, M., Mehran, S., Kalwar, I. H., ... & Memon, T. D. (2021). The importance of feature processing in deep-learning-based condition monitoring of motors. Mathematical Problems in Engineering, 2021.
[7]. Kumar, I. P., Mahaveerakannan, R., Kumar, K. P., Basu, I., Kumar, T. C. A., &Choche, M. (2022, March). A Design of Disease Diagnosis based Smart Healthcare Model using Deep Learning Technique. In 2022 International Conference on Electronics and Renewable Systems (ICEARS) (pp. 1444-1449). IEEE.
[8]. Tsagkatakis, G., Aidini, A., Fotiadou, K., Giannopoulos, M., Pentari, A., &Tsakalides, P. (2019). Survey of deep-learning approaches for remote sensing observation enhancement. Sensors, 19(18), 3929.
[9]. Zhongsheng, W., Jianguo, W., Sen, Y., &Jiaqiong, G. (2020). RETRACTED: Traffic identification and traffic analysis based on support vector machine. Concurrency and Computation: Practice and Experience, 32(2), e5292.
[10]. Aureli, D., Cianfrani, A., Diamanti, A., Vilchez, J. M. S., &Secci, S. (2020, April). Going beyond diffserv in ip traffic classification. In NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium (pp. 1-6). IEEE.
[11]. Yao, H., Gao, P., Wang, J., Zhang, P., Jiang, C., & Han, Z. (2019). Capsule network assisted IoT traffic classification mechanism for smart cities. IEEE Internet of Things Journal, 6(5), 7515-7525.
[12]. Rahman, A., Jin, J., Cricenti, A., Rahman, A., & Yuan, D. (2016, December). A cloud robotics framework of optimal task offloading for smart city applications. In 2016 IEEE Global Communications Conference (GLOBECOM) (pp. 1-7). IEEE..

