Mobile Traffic Forecasting Techniques

Haydar Qarawlus
Paderborn University
qarawlus@mail.upb.de

***Abstract*—The rise of traffic loads in networks is making the tasks of providing good quality of service more and more difficult. It is also limiting the ability to improve network management. That is why researchers are continuously working on providing new methods to predict network traffic. In this paper, several models will be discussed that are utilizing deep learning methods to predict network traffic. Other models that do not utilize machine learning will also be discussed and a performance evaluation comparison between the two models is conducted. The paper is concluded with selecting a model that outperforms the other models in the published metrics.**

I. Introduction

Networking demands are growing every day. With the introduction and rolling out of 5G mobile networks, that demands is expected to increase even more due to the rapid increase in Internet of Things (IoT) devices [1]. With this increase in traffic load the task of managing the network and its resources becomes increasingly difficult, this includes tasks such as traffic engineering, allocation of resources and quality of service provisioning [1]. Due to these reasons, predicting upcoming loads and anticipating traffic has become a crucial factor for network operators to guarantee optimum network resource optimizations [2]. With 5G and Network Function Virtualization (NFV), instantiating and scaling network services also becomes a relevant issue. In NFV, network services are created and placed on demand according to traffic loads. Therefore, predicting said traffic will allow for efficient decision making when it comes to where and when to instantiate virtual network functions (VNFs) and how much resources to allocate to the VNFs to ensure optimum performance [3].

The current availability of big data enables vast new methods to analyse network usage, and from that adding the ability to predict future traffic [4]. In earlier and some current attempts to predict mobile traffic, linear and non linear mathematical models were used such as AutoRegressive Integrated Moving Average (ARIMA) [5] and Support Vector Regression (SVR) [6]. These models use mathematical models to analyse traffic patterns and predict future traffic with a certain level of accuracy. Nowadays, with so many users using online services, the amount of data that is being generated is increasing daily [2]. This consumption of data by the users is also generating incredibly large amount of data at the network infrastructure level (e.g. system statistics, used resources, etc). Researchers such as [2] [3] [4] are now using this information to create and train deep learning models to try and analyse the data and to present an accurate traffic forecast for the network to ensure that better resource management and quality of service takes place. The authors of those models used traffic traces from known telecommunication companies presenting traffic in a considerably large metropolitan area. In this paper, the three main models will be discussed that use deep learning methods. Additionally, two models that use mathematical models. The main deep learning models use neural networks to analyse the past traffic and generate predictions. These models mainly use either Recurrent Neural Networks (RNN) or Convolutional Neural Networks (CNN) or a mix of both. [2] [4] Along with the use of those neural networks, Auto Encoders are also used in [4] in a stacked manner to provide extra accuracy to the predictions. In Section II, the models based on deep learning will be described extensively. In section III, models used to analyse time-series traffic information such as ARIMA and SVR that use mathematical models without the use of neural networks will be examined. The performance of the models will be examined and compared in section IV, and a conclusion on the presented topics will be presented in section V.

II. Deep Learning Models

Due to the availability of big data and today's computing power, using machine learning techniques such as deep learning in the network traffic prediction is now more accessible [7]. In this section, I will examine multiple models that utilize deep learning to predict mobile traffic with high levels of accuracy.

A. Hybrid Auto Encoder Model

In their paper [4], Wang et. al emphasize on the use of big system data (i.e. traffic load, resource usage, etc.) that are recorded at the mobile network's base stations to show spatial and temporal dependencies in the traffic that is being examined. Wang et al. argue that there is a temporal autocorrelation and spatial correlation among the generated traffic. This means that the traffic generated at a specific base node is related to the traffic generated at the same node, but at different times (temporal autocorrelation), and the traffic that is generated at a single base station is also related to traffic at other neighbouring base stations (spatial correlation). Due to this dependency in the data, the authors propose a new novel model that utilizes deep learning based on stacked multi level autoencoders and long short term memory units (LSTMs) for spatial and temporal modelling [4]. They believe that only using historical data of a single base station is not sufficient to accurately predict the future traffic, and that it is necessary to obtain the data of multiple nodes that are close to each other as well as the time of the data generation to generate accurate prediction of traffic for that node [4]. Additionally, the authors chose deep learning due to the fact that it can handle large amounts of input data with little manual configuration by the user [4]. To show the said correlation between neighbouring cells, they selected a big dataset from a Chinese wireless carrier called China Mobile. The authors' model to predict the data and show non-zero autocorrelation and non-zero spatial correlation consisted of the following: [4]

- A novel autoencoder based model for spatial modelling: This model consists of a Global Stacked AutoEncoder (GSAE) and multiple stacked auto encoders (LSAEs).
- Long Short Term Memory units for temporal modelling.

The data the authors collected was collected from 2,844 base stations that covered 6,500 square kilometres. The dataset includes an hourly average traffic load for a period of five months. The data was divided into square grids to facilitate measurement [4]. In this model, each cell is identified as a tuple (m, n). The downlink/uplink data of the cells is denoted as $D = \{d_{m,n,t}\} \forall m, n, t$ for all timeslots t. The data is then normalized into a [0, 1] range using the $tanh\ estimator$ method [4].

Wang et. al conducted a preliminary mathematical analysis of the data to discover correlation in the data. In their analysis, they used the sample AutoCorrelation Function (sample ACF) to detect temporal autocorrelation. They also used a widely used metric [4] to calculate for spatial correlation. Their analysis confirmed that there is a non-zero autocorrelation in the temporal domain, and a non-zero spatial correlation [4]. These findings motivated the authors [4] to design a novel hybrid deep learning model to take into account the spatiotemporal correlation and provide a prediction for each cell by using historical data from the cell itself as well as the cell neighbouring it. The proposed model is illustrated in Figure 1.

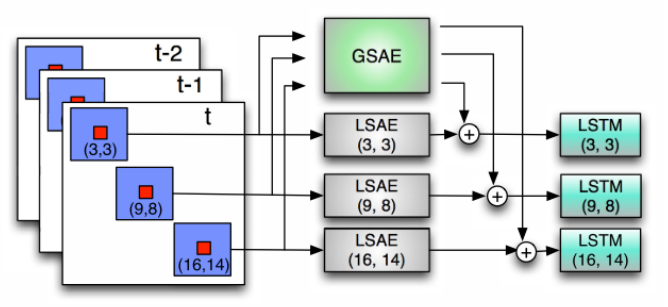

Fig. 1. The proposed deep learning model using stacked auto encoders. [4]

The model proposed by Wang et. al works by taking the data of the cell(s) of interest (shown as the red boxes) and the data from the neighbouring cells. The data collected from these cells include metrics such as downlink/uplink traffic load [4]. The data from all data patches is passed as input to a single global stacked autoencoder (GSAE), which then produces an encoded representation of this data that is called the global representation [4]. The data patches are then passed on to individual Local Stacked Auto Encoders (LSAEs) to generate another encoded representation of that data that is local to the LSAE. After both encodings are generated, the data are then concatenated to represent each patch, and then passed on to LSTMs for predictions [4]. The choice of RNN LSTM because of the availability of gates that control how much past information to pass through [4]. Since traditional training models are not sufficient to train the model proposed by Wang et. al, they came up with a new hybrid structure to train their model [4] The details of the Spatial and Temporal modelling techniques are discussed in the following sections.

Spatial Modelling:

The spatial modelling proposed by Wang et. al uses stacked autoencoders to effectively extract information that may not be extracted with a single auto encoder [4]. They use a trained GSAE to create a global representation of the given data patch, and due to the fact that spatial correlation exists for the data, the data is also passed to LSAE to obtain a better representation with less reconstruction loss [4] The data are passed through different layers of both autoencoders, each with different encoding and decoding weights. The first layer of the LSAE is trained to reduce the construction loss of the first layer in GSAE, and the higher layers of the LSAE are trained to give better representation of the data [4]. Overall, combining global and local autoencoders provide the following benefits [4]:

- Overall better representation of data: Since different data patches have common properties, the GSAE is used to capture that data. The different properties that have spatial dependency (i.e. dependant on the location of the data) are captured by the LSAE. This will give a thorough representation of the data that is better than a global representation only.
- Reduction in model size: Since autoencoders provide a representation of the data with a much lower size [4], stacking and using multiple autoencoders will divide the size of the data, thus the variables that need to be calculated over the autoencoders, which will make training for those models much faster than if using a single stacked auto encoder.
- Parallel Training Ability: Provided that the GSAE is trained, the training process of each independent LSAE can be in parallel to other LSAEs. This will reduce training time and improve performance.
- Application Specific Training: Since LSAEs handle different data patches, we can train the LSAEs that are only specific to the data patches that are of interest to the specific case that is being studied. This will also reduce training time and provide more specific results.

The authors use a well known greedy algorithm to train the GSAE [4]. The trained GSAE is used to train the LSAE. The authors wrote an algorithm [4] to train the model. The algorithm trains the first layer of the LSAEs differently than the other layers, since the first layer take the trained first layer of the GSAE as input [4] Also as discussed earlier, since training LSAEs is independent, and can be application specific, this step of training LSAEs with the algorithm can be done in parallel. This also means that the internal structure of each

LSAE can be created differently to tailor that specific application (i.e have different number of internal layers, number of hidden units, etc.). [4]

Temporal Modelling:

For Temporal Modelling, Wang et. al propose the use of ecurrent Neural Networks (RNN), this proposed network will take the representations that were learned from the previously discussed hybrid spatial model as its input. They propose to use Long Short Term memory as a solution to the problem of modelling long term dependencies with standard RNN [4]. The benefits of using LSTM lies with the fact that it has gates that make it possible to better analyse and adapt to previous runs and the current model with that data [4].

This model takes the data for a specific data patch for the past timeslots as input. This data is then passed to the global and local autoencoders for encoding. Afterwards, the original data patch $d_{m,n,t}$ and the encodings from the GSAE and LSAEs will be concatenated as a vector of all those three values and then passed to the LSTM to predict $d_{m,n,t+1}$ [4]. The model and its prediction process can be seen in Figure 2.

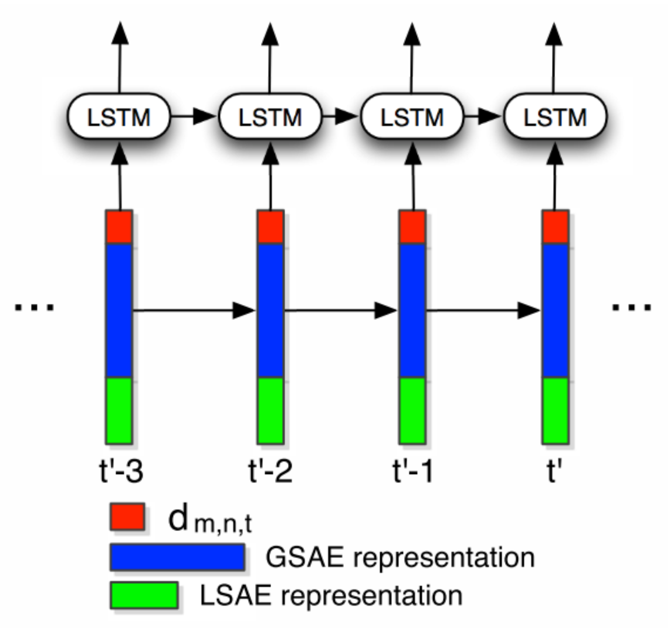

Fig. 2. The temporal model and prediction1. [4]

B. Combination of CNN and RNN Deep Learning

In this model, Huang et. al [2] provide a model to predict future mobile network traffic. Like the previous discussed model, this model relies on deep learning to achieve this prediction. It also operates on a basis of geographical and temporal features of the data [2]. However, unlike the previous model, this model does not use Auto Encoders to facilitate the prediction of future data, rather it uses multiple types of neural networks such as RNN, Convolutional neural networks (CNN) and a combination of both CNN-RNN [2]. In addition to the mix of neural networks, this model also presents the concept of using multi-task learning (MTL)architecture, which allows the model to be trained for multiple tasks as opposed to a single task [2]. This allows the model to produce more accurate results due to being trained on more scenarios and having a better level of generalization by sharing information among the tasks that are being learned [2].

To produce and tests their model, Huang et. al used a dataset from Telecom Italia [2]. The data is for the months of November and December and includes various data points such weather, social networks, electricity, and mobile telecommunications. The authors focus on the telecommunications record in the city of Milan [2]. The data are divided into geographical grids of 235 x 235 meters and includes the Square ID for each grid, the start and end time intervals, and the internet activity data [2].

Huang et. al utilize the stated neural networks (RNN, CNN, CNN-RNN) to observe the traffic data for the last hour. The models then produce the predicted traffic load for the next hour. Additionally, alongside the multiple neural networks multi-task learning is also being used to share the outputs of each task that is being learned among other tasks for better accuracy [2]. All runs of the models have the same input which is a four dimensional array that is defined as $T \times R \times C \times F$. The components of this array are [2]:

- T: Number of the time intervals of the call detail records (CDRs).
- R: Number of rows in the data grid.
- C: Number of columns in the data grid.
- F: Number of types of records to analysed.

The main concepts behind the model are the following:

Multitask Learning:

Multi-task learning (MTL) allows the model to learn several tasks at once [2]. It also allows for information to be shared among tasks. This allows for the training process to be quicker, as well as for the results to be more accurate. In [2], the MTL architecture consists of two main stages. The first and upper stage is the multi-task regression stage, and the lower stage is called the feature extraction stage. The bottom stage allows for the operation of different deep learning extracts features from the inputs and provide it to the regression layer. The upper stage then processes that input and provides a multiple task output that is used for joint training [2]. For their model, Huang et. al chose three tasks to be learned by their model: Maximum, minimum, and average network traffic. These tasks will be calculated for the next hour, and the information sharing among the tasks will occur in the features extraction layer [2]. The two main stages are illustrated in figure 3.

Recurrent Neural Network (RNN):

Due to the fact that RNN utilizes memory blocks, it is suitable to perform sequence processing on the time series data [2]. Additionally, when utilizing LTSM as part of RNN, it enables it to learn long-term dependencies of data [2]. The RNN architecture is shown in figure 2 in the feature extraction layer. It is shown as a multi-to-one architecture in which all LSTMs share the tasks with each other, and then pass it to the upper stage via only one cell [2].

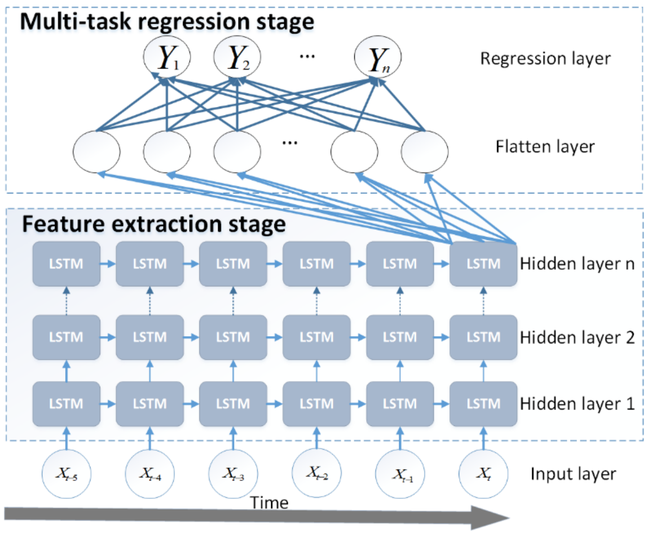

Fig. 3. The MTL architecture with multi-task regression on top, and RNN as a feature extraction at the bottom. [2]

Convolutional Neural Network:

Convolutional Neural Networks allow to capture geographic feature from the input data [2]. Huang et. al use 3D CNN to capture the spatial-temporal features of the data, as they pass time as the third dimension to the neural network [2]. The 3D CNN model is slightly customized by the authors and is shown as part of the mixed CNN-RNN architecture in figure 4. It consists of three convolutional layers and three pooling layers [2].

CNN-RNN Combination:

Huang et. al propose to combine both CNN and RNN neural networks to improve accuracy [2]. The proposed architecture combines CNN and RNN in a hierarchical order in the feature extraction stage [2]. In this architecture, CNN is responsible to extract spatial information, while RNN is responsible for capturing and extracting temporal information from the data [2]. The proposed architecture is shown in figure 4. The CDR inputs are passed through CNN to extract spatial information, then to RNN stage in that sequence [2].

C. *Densely Connected Convolutional Neural Networks*

In this model [8], Zhang et. al develop a model that also utilizes deep learning technology to predict future mobile traffic. Their model utilizes the CNN model in a new manner by using a densely connected version of CNN [8]. The model is similar to [2] and [4] in that it considers spatiotemporal correlation [8]. It is particularly similar to [2] in that it uses the same dataset that was released by Telecom Italia [8]. It does present a number of new concepts, such as modelling two temporal dependences instead of just one [8].

The model uses the densely connected CNN to capture both spatial and temporal dependences. The spatial dependence is captured by the convolution operation of the CNN, and the temporal dependences are captured by two CNNs, and the

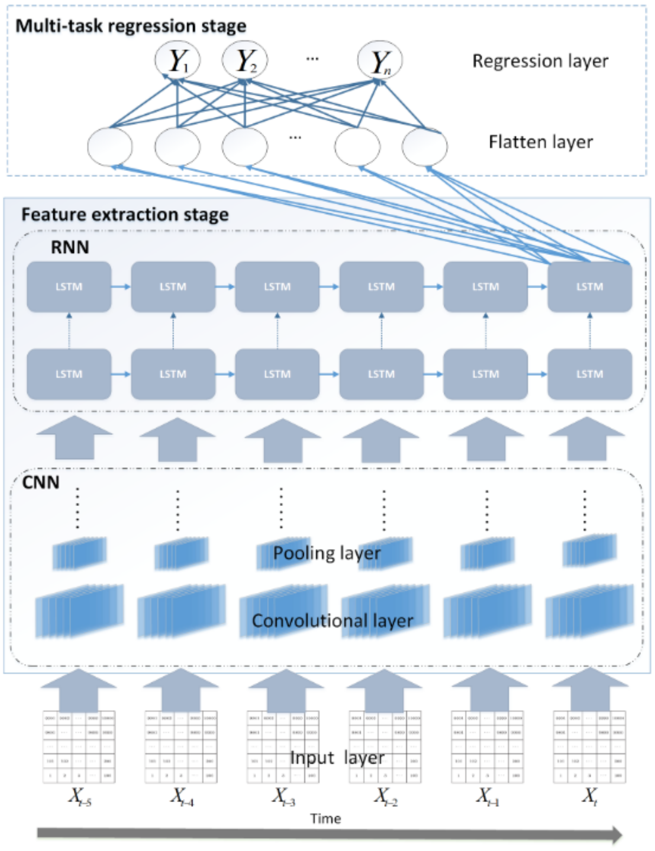

Fig. 4. The combined CNN-RNN deep learning architecture. [2]

results are then merged using a custom matrix-based scheme [8]

The data that Zhang et. al chose to use is the same dataset as [2]. They used an analysis that is similar to the analysis used in [4] to show spatial and temporal correlations. This discovery led them to produce a new model based on CNNs to predict traffic that mainly consists of the following main concepts, the structure of which is illustrated in figure 5. [8]

Training Data Construction:

During this stage, the input network traffic that is intended to train the model are represented as two-channel matrices that are similar to images [8]. The data will then be analysed using a sliding window scheme to calculate the period and closeness temporal dependencies [8]. Temporal closeness means how the time of the traffic is close to each other in units of time, and temporal period is basically is the period (e.g. days) that the traffic was recorded in [8]. Once that is done, the traffic is then concatenated as a vector (tensor) using the vector's first axis [8]. The concatenated traffic is then passed to the next stage [8].

Densely Connected CNN:

CNNs are great at capturing geographical (spatial) information [8] [2], and because of this, they are also used in this model to capture that information. Using multiple layers of convolutions in CNNs in this model allows to capture local and city global spatial dependencies, this is essential to correctly calculate the spatial dependencies of the data from each cell [8]. In this model, Zhang et. al use two CNNs that are similar in structure to separately model period and closeness dependences [8] . The networks then produce two separate representations for closeness and period, which are then fed to the next stage [8].

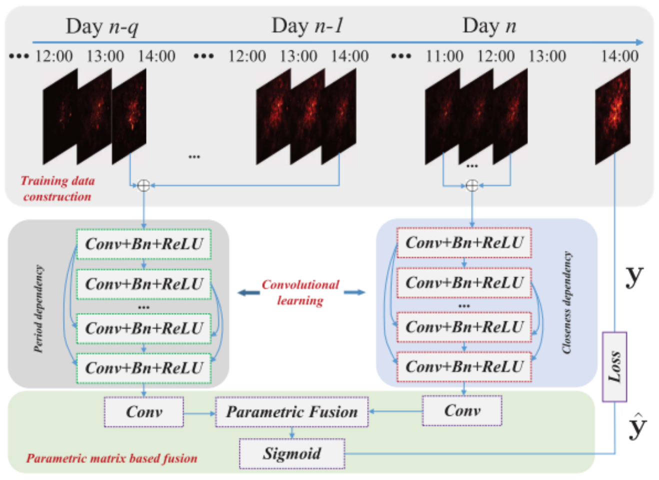

Fig. 5. The structure of the densely connected CNN model. [8]

Parametric Matrix Fusion:

After the representations were generated by the CNNs, Zhang et. al use a parametric matrix to fuse the two representations as they vary from each other [8]. A parametric matrix is proposed and used to capture and fuse the properties of the representations based on parameters that capture the relationship between period and closeness [8] The proposed model can then be trained a minimization of the Frobenius norm of the value that the model produces and the actual data that was recorded in the dataset, allowing the model to produce more accurate results. [8]

III. Non Deep Learning Algorithms

Machine learning and deep learning papers have been increasing in numbers in recent years [2] [4] [8] [1]. This can be related to the availability of big data that are being made available to researchers recently [9]. However, there are other models that exist that do not utilize machine learning to provide mobile traffic predictions [10] [6]. The models include Auto Regressive Integrated Moving Average (ARIMA) and Support Vector Regression (SVR). The use of the ARIMA model for traffic prediction is not new and can be found in papers dating in the late 1990s [11]. In the following sections, the concepts behind the ARIMA and SVR models will be discussed.

A. Auto Regressive Integrated Moving Average (ARIMA)

There are multiple implementations of the ARIMA model to predict network traffic [11] [5] [10] [12]. In this section, the model and implementation by Zhou et. al [12] will be discussed.

In their implementation, Zhou et. al introduce a combination of the ARIMA model and the Generalized Auto Regressive Conditional Heteroscedasticity (GARCH) model. The ARIMA model is a linear time series model, while the GARCH model is not, using this combination of linear and non linear models allows for a better overview and analysis of the data. [12]. The combination with GARCH also allow for the handling of bursts that may occur in the data, and also for its prediction [12] Zhou et. al provide a method to estimate the parameters that will be used for the prediction process, which are defined as 5 parameters (r, d, m, p, q), the first three of which are parameters of the ARIMA part (regression order parameter r, the differenced parameter d, and the moving average order m) [12]. The latter 2 parameters belong to the GARCH model, which are related to the back shift parameter (r, m) of the ARIMA model [12].

The prediction scheme in this model can be used to predict one time step or multiple time steps ahead [12]. Given the historical data and the estimated parameters, Zhou et. al use the minimum mean square error (MMSE) forecast method to measure the forecast performance of the predicted values [12]. This is used due to the fact that the forecast provides a range of values and not a single one [12]. Once the one step ahead traffic was generated, the model can be used again by aggregating the generated traffic to the historic traffic and running the model on it again [12]. This can be done recursively to generate traffic predictions for multiple steps ahead [12]. The predicted traffic is also compared with the actual traffic data to correct errors in the predictions [12]. The architecture of the model is illustrated in Figure 6.

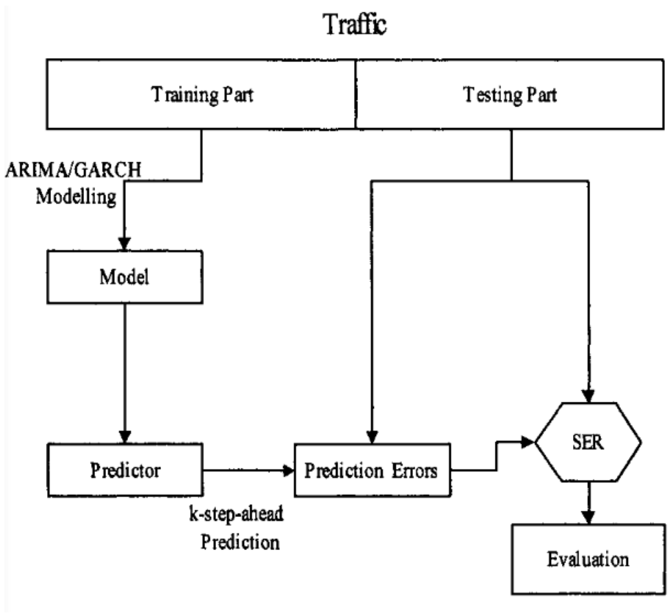

Fig. 6. The architecture of the ARIMA/GARCH prediction model. [12]

B. Support Vector Regression (SVR)

Support Vector Regression is a non linear model basically used to minimize Structural Risk (Structural Risk Minimization SRM) [6]. In their implementation of the model, Rizwan et. al use SVR to predict futrue mobile traffic for three main measures: minimum, maximum and mean granularities [6]. The authors use the same dataset as [2] and use similar techniques to [4] to analyse and prove spatiotemporal patterns of the data [6]. They calculate the Mean, Maximum, and Minimum levels of internet activity during the first stage of their analysis. Afterwards, they studied the spatiotemporal changes of the data [6]. Once those steps were finished, they use and train three independent SVR models to calculate each of the three different levels that were discussed earlier [6]. The SVRs are trained using only two weeks of data at a time and are the outputs are then validated using the third week's data [6]. The architecture of the model by Rizwan et. al is illustrated in Figure 7.

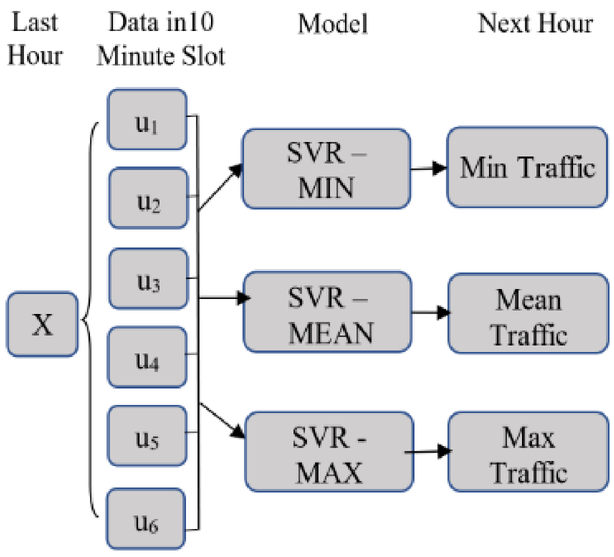

Fig. 7. The architecture of the SVR prediction model. [6]

The authors used a non-linear SVR as the result of their analysis of the data showed that the data is not linear. [6] The data from each hour is split into six values of 10 minutes each. The input is then mapped to a multidimensional and nonlinear feature space using a specific non-linear transformation function [6], the data is then an outcome is generated [6]. To measure the accuracy of the output, a loss function called the epsilon insensitive loss function [6] is used. This function ignores errors that are within epsilon (ϵ) from the correct values [6]. This aides in removing noise from the outcome and provides a more polished view of the outcome for the next hour [6].

IV. Performance Evaluation

The performance of the models is a crucial factor in determining their efficiency and whether to use them or not. The authors of each of the models discussed above perform multiple performance evaluations of their models and compare them against state-of-the-art models for evaluation purposes. There multiple metrics that are recorded when performing experiments on the models. Metrics such as Mean Accuracy (MA) [2], Mean Squared Error (MSE) and Mean Absolute Error (MAE) [4] are calculated or recorded from the experiments. The performance metrics of the models have only been conducted on cellular networks, since it is argued that regular models using big data is not very beneficial in backbone networks due to the complex infrastructure of such networks [1].

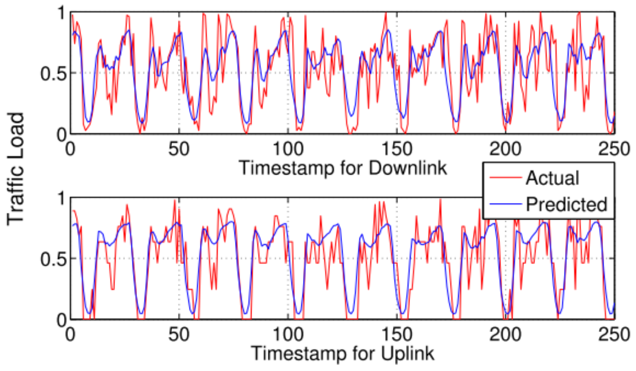

Fig. 8. Predicted traffic in [4] compared to real data. [4]

In [4], Wang et. al chose the output on random 15 cells for testing purposes. Figure 8 shows a comparison of the predicted traffic and the actual traffic trace [4]. The values show a MSE, MAE values of 0.042, 0.165 respectively. This shows that the data matches the trend of the actual data and showing that the model predicts the values with reasonable accuracy. [4]. Wang et. al then proceed to compare their prediction accuracy (prediction errors ratio) to other non machine learning models. Figure 9 shows the average prediction errors for the Auto Encoder based model (colored in purple) and both SVR and ARIMA models (colored black and green respectively) [4].

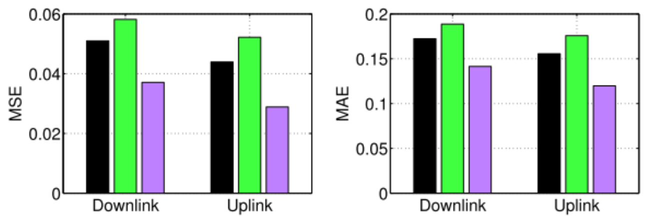

Fig. 9. Performance comparison of the AE based model (purple) compared to SVR (black) and ARIMA (green) [4]

From the graph we can see that the hybrid stacked autoencoder model outperforms both SVR and ARIMA models [4] From the diagrams, we can also see that SVR also outperforms ARIMA in that particular experiment.

In addition to the stacked autoencoder model in [4], Huang et. al [2] also use deep learning techniques. The also utilize multitask learning as explained in earlier. Utilizing the Mean Accuracy (MA) metric, they show the accuracy of the models they proposed using both single task and multi-task learning and compared them to the baseline ARIMA model [2]. The results are shown below in figure 10.

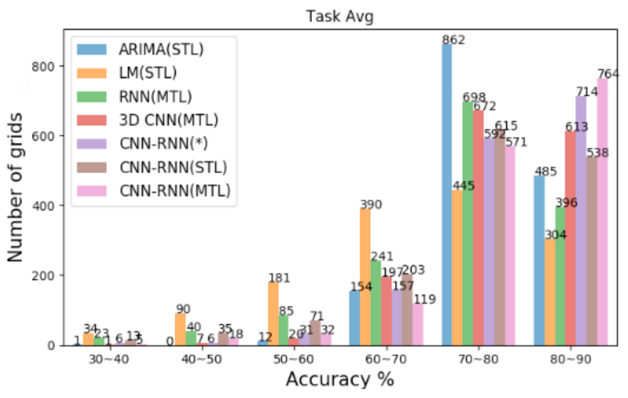

Fig. 10. Accuracy comparison of the models proposed by Huang et. al. [2]

From the figure it can be seen that ARIMA outperforms the other models for the most number of grids at an accuracy level of 70-80%. However, for higher accuracy, the multitask learning CNN-RNN hybrid model outperforms the other models significantly [2].

This claim however has been disagreed with in [6] for worst case scenarios. Rizwan et. al argue that in their experiments, their non deep learning model that is based on SVR achieves better results than the model by Huang et. al [6]. For the Minimum Accuracy (MA) metrics that was also used by Huang et. al, Rizwan et. al show that their particular implementation of the SVR model has the values 90.25%, 91%, 89.72% for min., mean, and max. values respectively. They also show that the model presented by Huang et. al produces much lower values for the same metrics (69%, 72%, and 67% for min., mean, max. MAs respectively) [6]. Rizwan et. al presented their model because they believe that deep learning might not be viable for every scenario, and that their model will perform better in scenarios where deep learning might not be viable [6]

For the time being, the SVR model presented by Rizwan et. al serves as a good model to predict traffic in situation in which using a deep learning algorithm might not yield the best results.

V. Conclusion

The rise of network demands has only increased the importance of predicting network loads to offer better quality of service and enable better network resource management. In this paper, several deep learning were discussed that are based traffic prediction models along with a couple of models that are not based on machine learning. The performance metrics of the models were also examined and compared.

References

[1] V. A. Le, P. L. Nguyen, and Y. Ji, "Deep Convolutional LSTM Network-based Traffic Matrix Prediction with Partial Information," *IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pp. 261–269, 2019.

[2] C. W. Huang, C. T. Chiang, and Q. Li, "A study of deep learning networks on mobile traffic forecasting," in *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC*, 2018.

[3] I. Alawe, A. Ksentini, Y. Hadjadj-Aoul, and P. Bertin, "Improving Traffic Forecasting for 5G Core Network Scalability: A Machine Learning Approach," *IEEE Network*, vol. 32, no. 6, pp. 42–49, 2018.

[4] J. Wang, J. Tang, Z. Xu, Y. Wang, G. Xue, X. Zhang, and D. Yang, "Spatiotemporal modeling and prediction in cellular networks: A big data enabled deep learning approach," *Proceedings - IEEE INFOCOM*, 2017.

[5] G. Jia, P. Yu, P. Xiyuan, C. Qiang, Y. Jiang, and D. Yufeng, "Traffic forecasting for mobile networks with multiplicative seasonal ARIMA models," *ICEMI 2009 - Proceedings of 9th International Conference on Electronic Measurement and Instruments*, pp. 3377–3380, 2009.

[6] A. Rizwan, K. Arshad, F. Fioranelli, A. Imran, and M. A. Imran, "Mobile Internet Activity Estimation and Analysis at High Granularity: SVR Model Approach," *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC*, vol. 2018-September, pp. 1–7, 2018.

[7] I. Shamshurin and J. Saltz, "Will deep learning change how teams execute big data projects?" in *2018 IEEE International Conference on Big Data (Big Data)*, Dec 2018, pp. 2813–2817.

[8] C. Zhang, H. Zhang, D. Yuan, and M. Zhang, "Citywide Cellular Traffic Prediction Based on Densely Connected Convolutional Neural Networks," *IEEE Communications Letters*, vol. 22, no. 8, pp. 1656–1659, 2018.

[9] F. Xu, Y. Lin, J. Huang, D. Wu, H. Shi, J. Song, and Y. Li, "Big Data Driven Mobile Traffic Understanding and Forecasting: A Time Series Approach," *IEEE Transactions on Services Computing*, vol. 9, no. 5, pp. 796–805, 2016.

[10] H. Z. Moayedi and M. A. Masnadi-Shirazi, "Arima model for network traffic prediction and anomaly detection," *Proceedings - International Symposium on Information Technology 2008, ITSim*, vol. 3, pp. 6–11, 2008.

[11] A. Adas, "Traffic models in broadband networks," *IEEE Communications Magazine*, vol. 35, no. 7, pp. 82–89, 1997.

[12] H. D. S. Z. Zhou, Bo, "Traffic modeling and prediction using arima/garch model," in *Modeling and Simulation Tools for Emerging Telecommunication Networks*, A. Nejat Ince and E. Topuz, Eds. Boston, MA: Springer US, 2006, pp. 101–121.