Applications of Artificial Neural Networks in Network Planning and Management

Reggie Collette

***Abstract*—Network planning and management today is, and always will be, governed by hard limits in capacity that cannot be exceeded. These limits are clearly defined in the standards that define the technology. These limits have been well covered in previous papers. Operators today, however, are becoming increasingly aware of the need to additionally manage the network based on a user perceived experience. The experience is often defined by the download speed that the user can achieve, known as throughput. The challenge that operators have today is that traditional methods of modeling throughput do not work because of the large number of drivers that influence what throughput is. This creates a challenge when planning capacity strategies into the future based on customer experience. In this paper we provide examples that describe why traditional methods of modeling do not work. Furthermore, even if they did work at some point, we discuss why they are not practical for future use.**

To break this barrier, we discuss the use and practicality of using artificial neural networks to accurately assess what the performance of a site will be in the future. The strong points of this approach are analyzed against future needs and the weak points are evaluated for awareness. Finally, a use case is provided for operationalizing this type of model into network planning strategies. This use case also discusses how this model can still be used if differentiation of services such as QCI (quality class indicator) is in use. In addition to the use case, a summary of the accuracy level that can be expected from such an approach is provided. In this case, the forecasted throughput error on a 6 month holdout set has a median error of 5% and a median absolute error of 27%. Work presented in this paper is primarily based on a feed forward network (FFN). The R language is used to access the toolsets offered by TensorFlow to prepare and model the data that is presented here.

***Index Terms*—capacity planning, quality of experience (QoE), analytics, artificial neural networks.**

I. INTRODUCTION

Network operators today are faced with ever growing data usage that is increasing at exponential rates. This growth means that forecasting and planning strategies must be under constant improvement to achieve the most effective spend rates for network capacity deployments.

A common planning metric that is often used in creating these strategies is data volume. It has some clear advantages. It is technology agnostic for the most part and allows for forecasting across all times and platforms. As it is technology agnostic, it ties all the business units together with common ground. Marketing, sales, and engineering can all talk the same language. Each unit understands volume. Last but not least, technology standards can be used to calculate what the volume limits for a network resource should be [7][8]. These limits are also stable over time as they are hard limits that are not changed by user behavior changes. By extension, if volume over some time can be ascertained from the standards, then throughput capabilities can also be ascertained. However, this will define the throughput capability of the cell, not the throughput per user which defines the user experience. Combining these two concepts can define the operating boundaries, the box if you will, with which a cell can operate in. Figure 1 below can help illustrate this point. The volume and cell throughput are laws that cannot be exceeded while the throughput per user can take on an infinite number of possibilities and combinations. Throughput per user can also be influenced by the types of management techniques an operator chooses to employ.

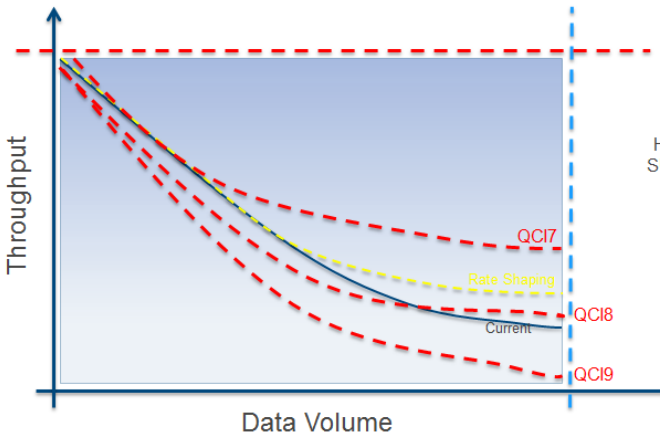

Fig. 1. Example of performance boundaries. X-axis is volume while y-axis is throughput. The throughput per user can fall anywhere inside the box based on various conditions that occur.

On the x axis is the total volume delivered by a cell in a time period. The y axis is the total cell throughput with boundaries defined. Defining these boundaries is a fairly straight forward calculation. However, defining what the user experience (or per user throughput) is at any point within the boundaries is much more difficult and vary based on many conditions. For example, general trends may develop that show throughputs per user will generally decline as usage increases, see the dashed lines labeled with QCI and rate shaping. That is only generalized however. It may also be that a point exists where there are few users with high data usage. In this case data

usage is high but since there are few users, the per user throughput is still acceptable. It could also be that another point exists where there are many, many users with relatively small usage per user. In this case, volume limits may not be reached but the throughput per user is low.

In this area of planning, volume gives us our hard limits and sets concrete expectations. It ties engineering to the rest of the business units and is a stable limit over time. However, per user throughput brings an additional element that makes it possible to make the best decisions possible for capacity planning. Both reference points are needed to make quality decisions, but how do you forecast throughput into the future? In order to forecast throughput you must first be able to model it from historical data and metrics from the network. It turns out, that is much easier said than done. In the next section we will discuss the challenges of modeling throughput with traditional methods followed by a machine learning solution which is capable of handling this task. Lastly we will discuss how the machine learning model and current forecasting techniques can be used to forecast the per user throughput into the future.

II. CURRENT CHALLENGES

Traditional, first step approaches to modeling these types of problems is often simple linear regression. As volume is a business requirement for forecasting input (this is what ties the network to the business units) it can be helpful to first understand the relationship between volume and the throughput per user. For this purpose we offer figure 2 below for a 5 MHz channel on a single vendor. Both volume and throughput are being measured at the PDCP (packet data convergence protocol) layer and are measured over a one hour reporting period.

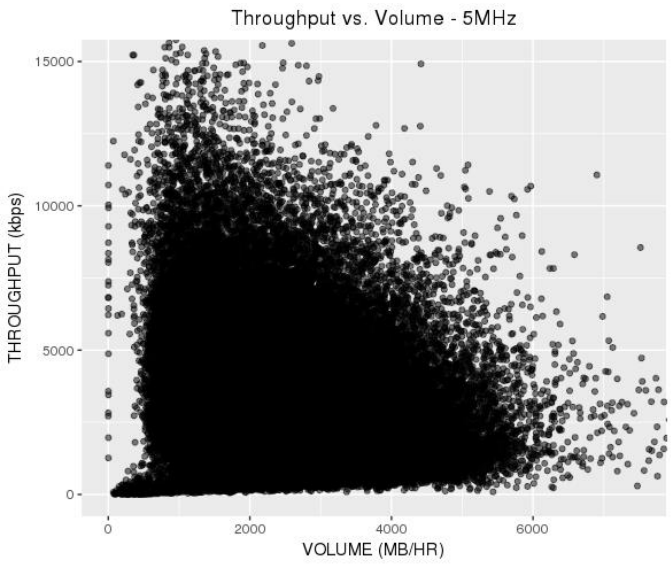

Fig. 2. Throughput given volume example

This illustration reveals that volume alone, while general trends can be seen, is not going to be a good predictor of what per user throughput is going to be. There are clearly other factors at play that are heavily influencing throughput per user.

The next natural step is to bring in more metrics or features that can add clarity and value. In LTE this could include many things such as RF quality, hardware type, or resource utilization levels. For our purposes here we will discuss physical resource block (PRB) utilization, physical downlink control channel (PDCCH) utilization, and channel quality indicators (CQI). With these indicators a multi-variate model approach can be built using methods from [4]. By developing the model with a training data set and a test data set, we can understand how well this model would perform. The data set used to train a multivariate approach had 25% of the data points withheld as a test set. The other 75% of points were used to train the model. By applying the resulting model to the 25% of data points withheld from training, we can understand how well the model performs on unknown data sets. By comparing the predicted value from the model to the actual measured throughput value in the test set, we can understand the magnitude of the error and where the errors occur. Figure 3 below illustrates the difference in the actual throughput versus the predicted throughput observed in a test data set.

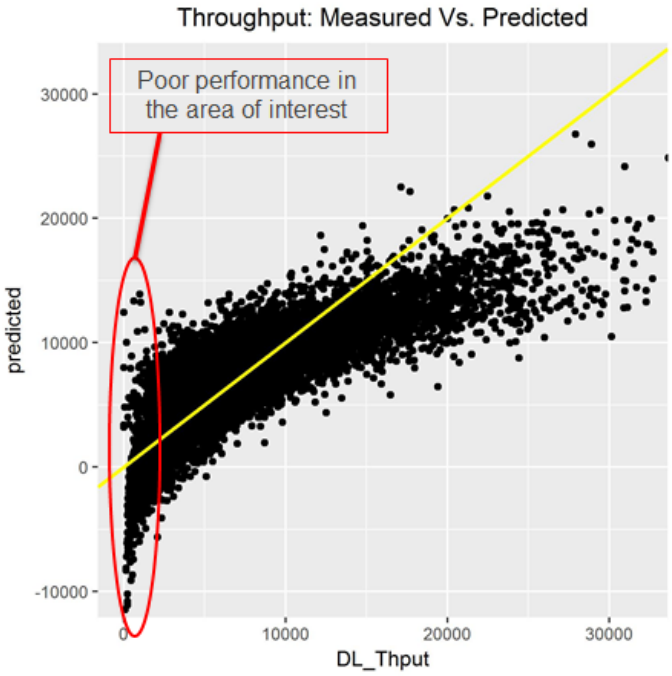

Fig 3. A comparison of the actual vs predited throughput using a multivariate regression approach.

In this case the correlation factor was actually measured at 0.84. This indicates that the model may have some level of explanatory power. This however, simply means that a change in volume can explain a significant percentage of the change in throughput. However, we are more interested in the predictive power of the model, particularly in a certain operating region. Let's consider the application and use case for a moment. An operator would begin to be concerned when throughput per user starts to get low. Let's assume for a

moment that threshold of concern is 1 Mbps. Does this model predict well when the 1 Mbps threshold is crossed? It does not. In fact, at lower throughputs, the predicted throughputs are deep into the negative range which is not a valid result.

These results begin to illustrate the complexities of what makes up the throughput per user performance. Other steps are also applied such as transforms and scaling, only to arrive at the same result.

Let's also assume that we could better classify the data. For example we could classify the data by vendor type, bandwidth, CQI, and loading level to create separate training sets. Assuming that gave acceptable results, an operator would have to maintain an enormous library of models in which each model would need to be maintained and tuned on an individual basis. Additionally, that approach would not be well suited to handle networks of the future which we will discuss in a later section and will touch on measures taken to manage the user experience.

The simple lesson we can learn from this short example is that there are many inputs required to model throughput. More than what traditional modeling methods are able to handle effectively. But how does an operator make progress toward this goal? Artificial neural networks are proposed as a solution. We will start with a brief introduction to artificial neural networks in the following section which will allow operators to feed a large number of inputs into the model to create a model that has much more predictive power to give expected throughputs.

III. INTRO TO ARTIFICIAL NEURAL NETOWORKS

An artificial neural network (ANN) is a machine learning algorithm that can solve some of the most complex problems known to date. ANN is at the core of revolutions like artificial intelligence (AI), deep learning, self-driving cars, and more. The word "neural" is used to describe this method because the method is often thought of as "learning" from data sets much in the same way as our own human brains learn. Figure 4 is a visual representation of a neural network in which you can see nodes (i.e. neurons). A series of nodes makes up a layer. Together, these nodes and layers make up a neural network[1].

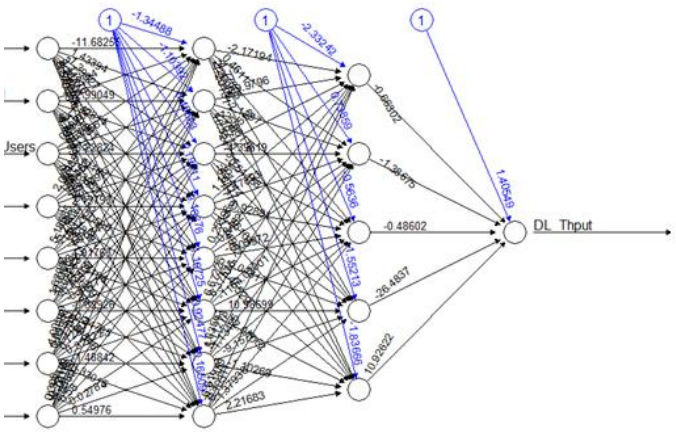

Fig. 4. Example of the architecture within an artificial neural network model.

Under the hood, each node takes in a set of data and uses linear algebra and matrix math to assign the combination of data a weighting. The weighting of the nodes in each layer is then provided to the nodes in the next layers [1]. Through a series of iterations and activation functions, the neural network "learns" the patterns in the data. Go through enough iterations and the machine learning model will literally memorize the data [2][3][6]. This is called over fitting and is not discussed in this paper but proper steps should be taken to strictly avoid overfitting of models.

The illustration shown in figure 4 only shows 8 inputs to the neural network that can then be used to predict a single output. Neural networks, however, are able to accept many inputs and still provide highly accurate models, even thousands of inputs in some cases. Just as it can take in many inputs, neural networks can also provide one or many outputs. The ability to provide many outputs is of tremendous value for our use case and will discuss why later.

While neural network models can provide spectacular results given complex data sets, they are not without weakness. In current modern tool sets, neural network models are a bit of a "black box" approach. Because of the highly complex math involving linear algebra, matrix math, and the sheer number of calculations under the hood, it is difficult to reverse engineer a neural network model. Another way to say this is, it is difficult to understand why a machine learning algorithm made a particular decision. There is much work in the academia and research space to overcome this but most of this work is in the development phase at the time of this writing.

IV. APPLYING ANN

In this section we will apply an ANN to a data set and illustrate its performance. For this writing, the Keras package is being used in R, to access the TensorFlow backend. The methods used are nearly identical to what is provided in [3]. TensorFlow is a language specifically designed to do machine learning. It was originally developed by Google and made available freely as open source software in 2016.

The following examples will apply a regression approach to machine learning utilizing a (64,64,64,32,16,1) network with a rectified linear unit activation function. This implies that the ANN input layer is 64 nodes wide, it has six layers total, and the last layer only has one output node. Inputs and features to this model will include CQI, PRB, Volume, PDCCH, and users.

The data set used for training in this example contains 64,000 points from a 5MHz network that is stratified across loading levels to obtain equal distribution of points across all loading levels. It is the same data set shown in the multivariate regression example previously.

Furthermore, the data set is broken into a training set and a test set. 25% of the points are withheld to make up the test set and 75% of the points are retained training, just as was done previously.

For this paper, the training is allowed to run for 10 epochs. More can be learned on this and the dangers of over fitting from Allaire in [3].

To prepare the data for the neural network, standard scaling practices are followed per the methods in [3]. This transforms the data of each metric to form a distribution of points normally distributed around a mean of zero so that scales of each metric are similar.

Upon completion of the training period, the model is applied to the test set. This is an important step in the process as the test set is a completely unknown set of data. This will provide guidance on how well the model will generalize to unknown data sets. Since the actual measured values are known in the test data set, we can compare the measured values to the predicted values to understand the deltas. To understand this it is helpful to look at two metrics, mean error and mean absolute error. Mean error will indicate whether the model is generally under predicting or over predicting. Mean absolute error (MAE) will indicate the magnitude of the errors (without direction) to reveal the overall accuracy.

Upon completing the training period and applying the model to the test set, we are able to see that the mean error of the model on the test set is -31 kbps and the MAE is 482 kbps. Figure 5 helps visualize this by plotting the predicted values against the measured values. If the performance of the model was perfect, all the points would fall on a line with a slope of 1 as shown by the yellow line.

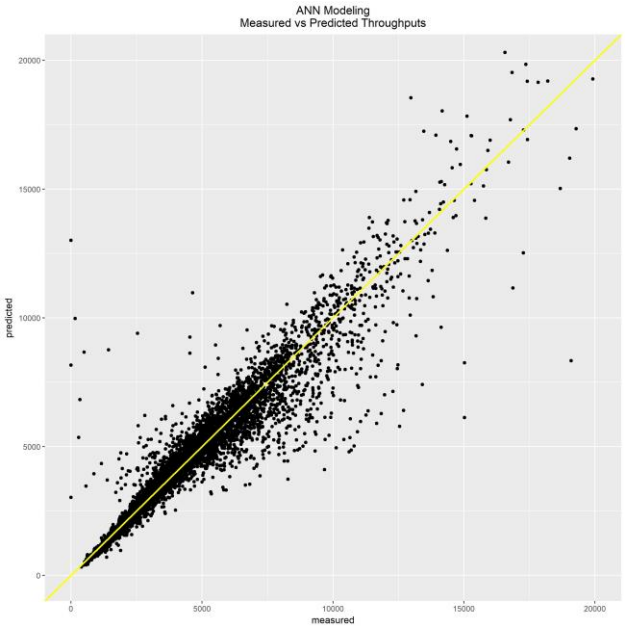

Fig 5. Comparison of measured values (x-axis) to values predicted by the ANN model (y-axis)

Comparing these results to the traditional methods of the previous discussion, the model already looks much better. How well does it perform in the area of interest though as discussed previously? Let's assume a 1 Mbps threshold again. To inspect this closer we use figure 6 which provides a narrowed view of the 0 to 2 Mbps region and provides it as a boxplot for ease of interpretation. At a 1 Mbps measured value, the error is +/- 150 kbps, much better than achieved previously.

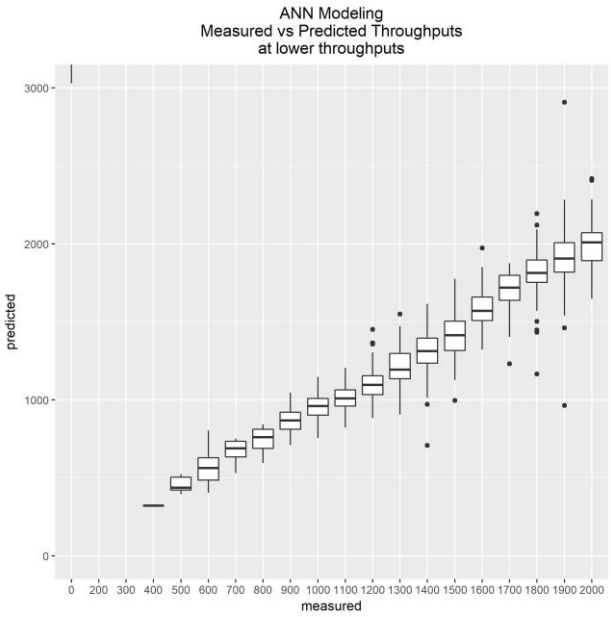

Fig. 6. Comparison of measured throughput values to ANN model predicted values. In units of kbps.

Assuming this level of performance is acceptable, the model could be deployed to any platform or language utilizing the TensorFlow backend. This includes R, Python, and of course TensorFlow itself [3]. If these results are not deemed to be acceptable, further work may be needed. This could include modifying the neural network architecture to add more layers or nodes, although improvements from this will depend highly on the data set and type of problem. It may mean adding more or different inputs to the model. Other methods prior to applying machine learning could also be employed to ensure the model is using the metrics which are the largest contributors to predicting throughput so that you may choose the top N predictors for neural network inputs. In general using the least number of predictors required to achieve usable results is recommended, particularly when applying these results to forecasting which is discussed in the next section.

V. FORECASTING THROUGHPUTS

Forecasting at its simplest is using a time series to predict the future. This often involves simple to complex algorithms. A common approach in many industries, telecom included, is to use approaches such as seasonally adjusted ARIMA models or exponential smoothing (ETS) [9][10]. If this is generally accepted in practice, why then, can we not forecast throughput itself as a simple time series if we have historical data?

While it is not the focus of this paper, throughput as a function of load has an exponential decay to it based on many factors. This by itself could probably be overcome. However, the rate of this decay depends on many factors such as, but not limited to: hardware, bandwidth, channel quality, and vendor type just to name a few. There would be an infinite number of "curves" that would have to be indexed and stored for later

retrieval. Traditional methods of ARIMA and exponential methods will tend towards linear forecasts (seasonal components aside). Furthermore, they are heavily affected by any trends that develop late in the time series. They are therefore not well suited by themselves to predict throughput well into long term time horizons.

ARIMA and smoothing methods, however, do work well on predicting individual metrics which do tend to have linear growth with some seasonal component (excluding outside influences) such as volume, PDCCH, PRB, users, etc. With this in mind, the individual metrics can be forecasted into the future. Using the neural network model constructed previously, forecasted metric values can be fed to the model to get future predicted throughputs. A visualization of this flow can be seen in figure 7.

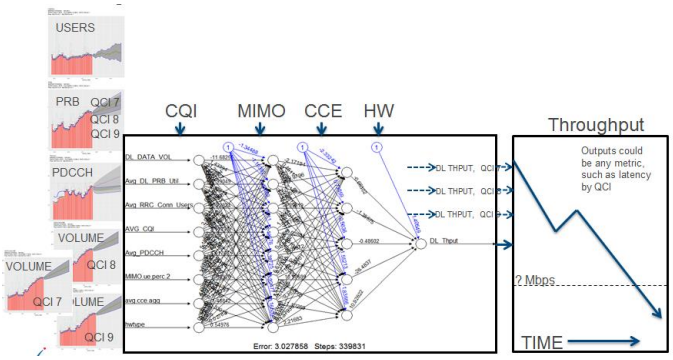

Fig. 7. Example of forecasted metrics being fed into a neural network model to produce future forecasted throughput values.

VII. PERFORMANCE

To quantify the accuracy performance of using ANN models in forecasting as previously described, an analysis was completed on a set of 10,000 random cells. Two years of monthly data was available as historical data for metrics discussed previously (including throughput). A time series forecast was created for each metric for each cell using 18 months of data while the most recent six months of data was used as a holdout set.

Using models tuned with the methods previously discussed, the ANN models are applied to the forecasted metrics for each cell. This provides a forecasted throughput value alongside a known throughput value. Comparing these two values will quantify the amount of error. For this discussion, a positive value of error will indicate over-forecasting while a negative value will indicate under-forecasting. Table 1 below summarizes these results as a percentage of error. The "Time Series" method utilized a multivariate ARIMA method that forecasted the individual metrics as well as throughput. The "ANN on TS" method applied a neural network model tuned as shown in section four to the individual metrics that were forecasted using the same multivariate approach.

Method	Mean Error %	Median Error %	Mean Abs Error %	Median Abs Error %
Time Series	-5.0%	4.8%	43.0%	31.0%
ANN on TS	20.0%	5.0%	37.0%	27.0%

Table 1. Summary of results comparing time series forecasting and application of ANN models to the time series.

The results from table 1 imply that the median errors are nearly the same when comparing time series only forecasting to use of ANN models. However the mean error also illustrates that the bias of the time series only forecasting is to under forecast while the bias of the ANN models is to over forecast. In this case it is important to be clear that the time series only forecast would trigger more capacity triggers as it is driving throughputs down more aggressively. The application of the ANN model would trigger less capacity needs because it is biased towards maintaining higher throughput due to slight over forecasting. Somewhere in between is where the reality will be.

Given the above interpretation, an illustration of the general bias can be provided. In figure 8 below, a sample forecast is provided. "Historical" values are the actual historical values of the throughput. "Thput_ANN_Historical" is the predicted throughput from the ANN model using historical metrics. In this case, the throughput model performs very well on known historical metrics. Each of those metrics is forecasted as a time series. The throughput itself is also forecasted as a time series and provided in figure 8 as "forecasted". The ANN model is also applied to the forecasted metrics and provided as "Thput_ANN_Forecasted".

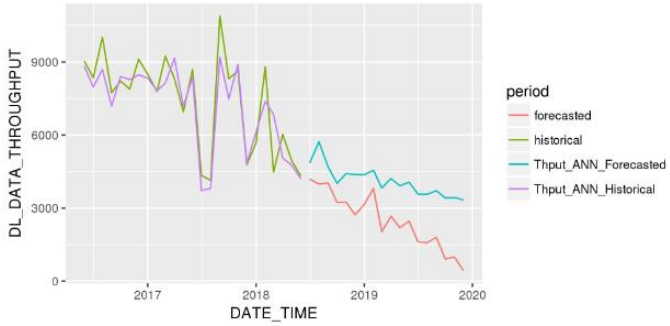

Fig. 8. Example scenario comparing response of a time series only forecast to that of a time series forecast with ANN applied.

Through this illustration in figure 8, it is shown that the time series only forecast picks up on a recent decline in the throughput time series and predicts a more aggressive decline than when the ANN is applied. The ANN model takes into account the rate of growth of all the other metrics and predicts that the decline in throughput will not be as aggressive and the rate of decline will actually slow down as time goes on. During testing, inspection of thousands of these cells revealed this same scenario across the board.

VIII. EXPECTATIONS

It is important to understand the expectations of ANN modeling and be aware of what it can and cannot be expected

to do. As discussed previously, an ANN model is somewhat of a black box approach. It is difficult to extract why a model made the decision that it did. For example, it would not be possible to generalize from the model what an increase in a single metric would do to throughput values. One can arrive at the answer by transforming the data and re-applying the model to obtain new outputs. However, it is not possible to come up with a static description of what happens under certain conditions. This may not be an issue for the use case we have discussed here, but what if your model must be clearly understood and clearly interpretable? For example, what if regulatory requirements required that a company must be able to explain why and how a model behaves for each specific piece of criteria? This is the case for industries such as credit ratings [3]. In this case, ANN models may be powerful tools, but they cannot be used due to the lack of interpretability required by regulation.

In terms of overall maintenance, these models are also not a deploy-and-walkaway scenario either, at least for the proposed use case. Behaviors may change over time. It is not necessarily expected that the model needs to be trained before each use, but it should be validated against a new test data set at a minimum. This validation process could even be automated into reports to speed the process but an engineer must still look at it and assess model performance. Appropriate thresholds could be applied to determine ahead of time that the model will need to be retrained whenever it falls outside a certain range of accuracy. This retraining process could be as simple as pulling more recent data to train from or could require adding new metrics to the training data.

VIII. OTHER APPLICATIONS

The power of neural network modeling lies not only in its ability to accurately model throughput, but also in its vast flexibility. If new metrics become necessary, they can quickly be added to the model and the model can be redeployed. Also recall that the model can have multiple outputs.

The flexibility of the model in this regard is important to operators. Operators are becoming more active in shaping the network experience with things like rate shaping, service differentiation, deep packet inspection to prioritize services, etc. Other concepts are also beginning to emerge that have not been implemented yet such as 5G, network slicing, etc. These tools, while very useful to the operators and beneficial to users, makes modeling throughputs all that more complex because active interventions are taking place that would otherwise be extremely difficult to account for in traditional methods. With a little creativity, most of these interventions can usually be inserted into data sets as features to be used in training a neural network model [3].

In the case of prioritization, QCI (quality class indicator) management can be an effective tool. However, this means an operator must be able to predict throughput for each QCI for it to work. Implementation and configuration strategies for doing this are not in the scope of this paper and could be an entire topic in itself. In the context of this paper, this means an operator must produce multiple outputs from one model. That is, one throughput per user prediction for each QCI class. In current 4G implementations, this means a minimum of 9 outputs from a single model and possibly more depending on operator implementations. As we have already discussed, this is very possible for neural networks. Certainly, more inputs will be required to arrive at this as there may be separate inputs for each QCI class for metrics such as volume, users, etc. If we started with 8 inputs, and provide QCI metrics for 3 of them, the number of inputs increases to 32. This significantly increases the number of inputs to the model but neural networks are well suited to handle this. This begins to demonstrate that neural networks may be a nice to have tool for now, but will become an essential requirement for planning in the future.

In the case of concepts like network slicing, the network is virtualized and appears as if it's an entire network dedicated to a particular user group. Each group may have their own behaviors and growth rates. However, they still share the same physical resources and therefore must be considered as a whole when developing capacity planning strategies. It remains to be seen how many network "slices" might exist in the future, but let's consider only three slices with QCI implementations on each. We stated previously that QCI implementations by itself increase the number of inputs to 32 and outputs to 9. But now we have three network slices to contend with. We now have 96 inputs and 27 outputs for every single cell in the network. This is of course without even considering 5G which may also share radios, basebands, and other types of hardware.

IX. CONCLUSIONS

Neural networks provide an amazing opportunity to solve problems that would have been otherwise unsolvable using traditional methods. The only way to solve these in the past was to literally have an engineer review each case and apply their knowledge of the network and market space to make an educated, fact based decision. Looking into the future as networks become far more complex, this is not sustainable nor is it effective.

In this case the power of neural networks will not replace an engineer. Rather, it will augment the engineer's ability to make quicker, more educated decisions, and also do it more consistently. It allows the engineer to scoop the complexity into the model to keep it from becoming a distraction. The complexity is still there, and will continue to get more complex, it is just allowing the engineer to focus more on the outputs and results. It is still important for each engineer to understand these complex interactions. The model can be revisited and even trained in a supervised fashion in real time if needed. The important thing is that the model can be revisited and retrained on an as needed basis and the engineer does not need to consider or store every bit of the information for every single decision that is made for every cell during the capacity planning process.

Lastly, while forecasting accuracy itself is not within the scope of this paper, it can't be left unsaid that the effectiveness of this work is dependent on the accuracy of the forecasting

methodologies. Every operator should have a good understanding of what they are able to expect (and not expect) from their forecasting algorithms in terms of accuracy. While ARIMA and ETS methods are common solutions for many industries, neural networks can also do time series forecasting. Feed forward networks (FFN) were the type of neural network used in this paper. However, recurrent neural networks (RNN) have been very popular in many forecasting competitions and often take first place or top spots among such events. Study of forecasting accuracy and modern day RNN's could be an area of study that could extend this work to further enhance its effectiveness

REFERENCES

[1] Goodfellow, Bengio, Courville, *DEEP LEARNING*. Cambridge, Massachusetts.
[2] Brett Lantz, *Machine Learning with R.* United Kingdom.
[3] Chollet, Allaire, *Deep Learning with R.* Shelter Island, New York.
[4] Fox, Weisberg, *An R Companion to Applied Regression.* Thousand Oaks, California.
[5] Sesia, Toufik, Baker, *LTE – The UMTS Long Term Evolution.* United Kingdom.
[6] Geron, *Hands-On Machine Learning with Scikit-Learn & TensorFlow*. Sebastopol, California.
[7] 3GPP TS 36.213 Release 10, "Physical Layer Procedures". March, 2014.
[8] 3GPP TS 36.211 Release 10, "Physical Channels and Modulation". March, 2014.
[9] Mills, *The Foundations of Modern Time Series Analysis*. UK
[10] Derryberry, *Basic Data Analysis for Time Series with R.* Hoboken, New Jersey.