Network Assurance in Intent-Based Networking Data Centers with Machine Learning Techniques

Xiaoang Zheng
Department of Software Engineering and IT
École de Technologie Supérieure Montreal
Montreal, Canada
seanzheng21@gmail.com

Aris Leivadeas
Department of Software Engineering and IT
École de Technologie Supérieure Montreal
Montreal, Canada
aris.leivadeas@etsmtl.ca

***Abstract*—Intent-Based Networking (IBN) is a recently proposed networking solution that allows networks to be configured and adapted autonomously according to the users' or operators' high-level intentions. However, a significant component of IBN is to assure that the network accurately and automatically deploys the intent throughout its lifecycle. To this end, in this study, we propose a network assurance solution for data center IBN networks. For the assurance model, we propose some specific data preparation procedures and Machine Learning (ML) models for the problem of time series forecasting. Specifically, we construct three main ML models that are based on the architecture of Convolutional Neural Networks (CNN) and Recurrent Neural Network (RNN). Our evaluation experiments, based on real data center Virtual Machine (VM) data traces, reveal the effectiveness of our methods in terms of CPU percentage usage prediction accuracy and speed. At the same time, our best-performing model can predict sufficiently far into the future with good accuracy.**

***Index Terms*—IBN, Network assurance, Machine Learning, Neural Networks, time series forecasting**

I. Introduction

With the latest advancements in networking technologies and the fast development of modern networks, we expect the networking systems to host numerous applications of various types and requirements. Modern networking technologies are able to provide technological solutions to diverse industrial, commercial, and social application domains. The fast growth of the network needs and various application requirements bring many challenges to network solutions. As a result, we expect the latest networking services to adapt faster and faster to diverse features and constantly evolving requirements, in order to align themselves with the latest business objectives.

In recent years, academic and industrial researchers have proposed a new network model called Intent-Based Networking also known as IBN. Networks using IBN solutions are able to configure and adapt autonomously to the user or operator intentions [1]. The technical network details and specific operations are encapsulated in user intents and hidden from users. Within IBN-based services, users can state their high-level and declarative intents and IBN directly translates them into device-level configurations required to achieve the desired functionalities. For network operators, IBN also offers some policy-based abstractions to just state their desired outcome in the form of network intents and then validate the network's automatic actions [2].

IBN proposes many benefits for businesses, network providers, and other stakeholders. From the network provider perspective, IBN enforces a network-wide and system-oriented management upon programmable infrastructures [2]. Allowing users to express their intents in an abstract way enables business teams and network engineers to simplify and accelerate the way network services are offered. Security teams can also benefit from IBN by proposing their own security-related intents, which are then fulfilled and monitored in IBN's activation-assurance loop. With these benefits, IBN has become one of the most promising candidates in the entire networking industry.

There are three main functional components in IBN [2]. The first component is the intent expression module with which users express their high-level intent. The second component is the activation component, which consists of many controllers and orchestration functions that deliver the established policies across all of the relevant network domains. The final IBN component, and the focus of this study, is the assurance component. By analyzing the contextual data of the network, this component constantly checks if the intent has been applied as per users' requests. The assurance modules monitor the network's exhibited behavior from telemetry data and report the compliance status.

Accordingly, the main objective of this study is to develop an assurance model for IBN using Artificial Intelligence (AI) techniques and ML algorithms. Among the three main types of networks for IBN to consider (carrier networks, data center networks, and enterprise networks) [3], in this study we focus on the data center networks. Trained with some historical network data, ML models like LSTM (Long Short-Term Memory) and CNN (Convolutional Neural Network) are used in this study to predict the CPU usage of VMs in the near future, based on the current networking metrics and the latest trends. The prediction of CPU usage will be used by data center network administrators to take preventive actions (migrate VMs, perform scaling, modify traffic routes, etc) to assure the level of Quality of Service (QoS) and reliability expressed in the intent.

The rest of this paper is organized as follows: Section II presents some related works in the domain of IBN and network assurance. In Section III, we present our proposed architecture

978-3-903176-36-2 ©2021 IFIP

of the IBN assurance system model. Then Section IV discusses the experimental results and performance evaluations of this study. Lastly, we will present our conclusions and propose some future work directions.

II. Related Work

In this section, we divide the pertinent literature into two main categories. The first consists of some studies on IBN-related topics and network assurance in general, while the second focuses distinctively on network assurance with AI and ML techniques.

A. Network Assurance in IBN

Even though IBN is a new technology it has already drawn the interest of major standardization and industrial organizations. For example, the Internet Engineering Task Force (IETF) [1] clarifies the concept of ”intent” and formally defines the terms, concepts, and functionality of Intent-Based Networking. Particular emphasis is also given on the closed loop automation of the network by properly defining the functions, roles, and steps for intent assurance and intent fulfillment. Additionally, IETF made an effort to classify intents according to the type of users (e.g. customers, operators, developers, etc.), the type of the intent (e.g. customer service intent, network intent, operational task intent), and types of networks (e.g. carrier, data center, and enterprise networks) [3].

In terms of industry initiatives, Cisco is one of the pioneers in the IBN domain. For instance, Cisco provides an architectural overview of Cisco’s Digital Network Architecture (DNA), one of the first IBN products currently on the market [4]. It provides detailed descriptions of network intents and policies in the context of Cisco’s services. Regarding network assurance, a DNA analytics engine is presented which is based on three main components: telemetry interfaces, analytics engine and controller, and Northbound APIs.

IBN is also often associated with Software Define Networking (SDN) and relies on SDN monitoring capabilites for providing the necessary assurance. For example, the authors in [5] describe the development of an IBN-enable SDN structure for monitoring the quality of voice and video streams in real time, which allows to keep the Quality of Experience (QoE) at an acceptable level. The system periodically monitors the required QoE parameters in the network, and when an unacceptable quality is detected in the pre-selected ”best” path, a reserved (backup) path is used to improve the quality of service. Another way to restore the quality of communication in multi-layer SDN transport networks is proposed in [6]. Specifically, the authors proposed two dynamic restoration algorithms for network assurance which allow to individually restore each failure-affected service while meeting bandwidth, latency, availability and service disruption tolerance.

Network assurance can be also established as an automation method during the network-level testing phase of system integration. One such approach is proposed for IBN-based Information and Communication Technology (ICT) systems [7]. More precisely, authors proposed and experimentally validated that a flexible generation of evaluation programs can bring significant time reduction in the system integration testing phase to assure the network performance. Finally, network assurance can be established semi-automatically, by allowing users to provide feedback on their satisfaction level when consuming a network service [8].

B. Network Assurance with AI and ML techniques

In the previous subsection, we briefly provided some IBN directions and how network assurance is considered. Nonetheless, most of them focus in monitoring tools to perform a reactive-oriented assurance. In this part, we present studies that can assure network intents by proactively guessing the network’s behavior. For instance, an intent-based closed control loop (CCL) system platform for service assurance through ML-based QoE estimation from network-level monitoring is presented in [9]. The QoE estimation uses a Random Forest classifier with multiple variables for the prediction of network QoS metrics. The model is trained with per-segment streaming traffic for specific users, network conditions, and network topology. In the experiment, the CCL workflow triggers the Orchestrator to take appropriate network-level actions to avoid network QoS degradations for the video service.

The authors in [10] also resorted in ML techniques to build a self-assurance IBN system on top of SDN and Network Function Virtualization (NFV) platforms. In the proposed assurance engine, alarms are activated when resources reach max utilization and a ML model periodically receives the resource status and updates the system in accordance with future state requirements. In fact IBN is expected to play a major role in automating the management and orchestration process in an NFV and SDN context [11]. Another ML approach for an SDN-based IBN architecture is proposed in [12], with the goal to improve QoE oriented cognitive networks. This framework explains the ML model flow including metrics selection, function training, and model evaluation.

Reinforcement Learning is another category of ML models that can be used for network assurance. Accordingly, the authors in [13] proposed a Reinforcement Learning method to automate the link capacity management in a Digital Subcarrier Multiplexing (DSCM) IBN system. In this framework, the intent is a request to create a virtual link between two end nodes with the best Quality of Transmission (QoT). Thus, the authors modeled their link capacity management framework with a reinforcement learning technique called Q-learning, which uses rewards and penalties to guide the state of the links to make proper adjustments.

In the context of network assurance in a data center environment, the study of [14] aims to enable easy configuration through high-level instructions and the use of ML for automatic control over the network infrastructure. The GWA-T-13 Materna dataset [15] was used, which contains the resource utilization history of three traces of distributed data centers. The authors proposed the use of a Neural Network Multilayer Perceptron (NN MLP) classifier. The MLP is feeding forward an Artificial Neural Network (ANN) that consists of an input layer, a hidden layer and output layer, while it uses back propagation for weight training. The model's training and test accuracies are between 89 and 95%.

Similar to [14] in this work we use the same dataset with the goal to increase the CPU prediction accuracy as part of the network assurance process. However, we explore different ML techniques and we manage to achieve even higher accuracy with low prediction times. Furthermore, instead of providing predictions for the next 5min period as in [14], we accomplish to predict the VM's CPU utilisation for the next 30 min without loosing significant accuracy. The latter can be proved crucial by giving the necessary time to the network and its operator to react appropriately according to the level of the increase or decrease of the CPU utilization (e.g. to perform VM migrations, resource scaling, shut down servers, etc.).

III. System Model

In this section, we present our high-level network assurance architecture solution for an IBN enabled data center. Firstly, we discuss the general architectural components of our proposed framework and secondly, we provide more in-depth details about the ML models that are used in our solution.

A. *IBN Assurance Engine Architecture for Data Centers*

The architectural designs for different solutions and use cases of IBN can be very different from each other, because of the differences in networking requirements, expectations, and priorities [3]. Different use cases of IBN need to be studied separately and may require different architectures.

According to IETF's IBN classification proposal, there are three main types of networks for IBN to consider: carrier networks, data center networks, and enterprise networks [3]. Each of these types has its typical types of intent users. In this part of our study, we focus on network assurance for data centers that provide IBN-based services. For example, a data center's administrator may have the following load balancing intent: for traffic that require NFV service chaining, limit the max load of nodes below 50% and the max load of links below 70%. To address examples like this, in this part of the study we propose an architecture to support assurance in the scenario of data center networks. Our architectural design strictly follows the concepts and principles of IBN defined in [1] and [3]. Our design is also guided by some existing industrial solutions described in [2]. Figure 1 describes the main components and their interactions of the proposed assurance architecture.

In our envisioned architecture, telemetry data are gathered from services that are deployed in the data center's network. Once the data are gathered, they need to be normalized before any further exploration and storage. After normalization, the data goes through the deduplication process, where the amount of received duplicate information is reduced. After deduplication, some optional cleaning steps like compression, aggregation, and filtering may take place. The type and amount of these cleaning steps mainly depend on the telemetry data's quality and the expected quality of downstream components.

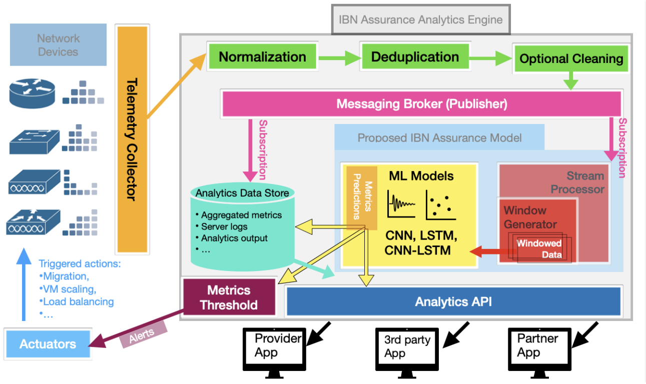

Fig. 1. Datacenter IBN assurance architecture

Next, a messaging broker publishes the data produced in the previous steps. It uses a publisher/subscriber pattern to interact with downstream pipelines like the analytics data store and the stream processor [4]. The advantage of this publisher/subscriber pattern is that the subscriber can retain all the emitted information from the publisher, even under heavy traffic load. Architectures with publisher/subscriber patterns are ideal for building real-time data pipelines and streaming applications thanks to the benefits of horizontal scalability, fault tolerance, and speed clearance.

The stream processor is a subscriber of the messaging broker, which transforms the incoming data into a data stream. The data from the messaging broker are not directly ready to be consumed from the ML forecasting models, because the previous components are aiming for general network telemetry data. Some additional preparations are required in order to provide ML models with the most suitable input data. Firstly, we need to detect and correct missing or incorrect values. When a defective entry is detected, we can discard the entry or try to infer the correct values. When the data are collected by the telemetry collectors, they contain many features. However, many of the features are not very helpful to the ML models to make predictions. Keeping all these fields will result in the input data's high dimension, which affects forecasting accuracy and significantly slows down the ML models' training and learning speed. So in the stream processor, we need to reduce the dimension of the input before passing it to the ML models. Subsequently, the data are normalized and passed to the window generator.

In the stream processor, we use a customized window generator where sequential data from the stream processor's data source are transformed to windows containing consecutive samples. At a given time, the window generator produces a window that contains the latest data it receives. The time difference between two consecutive entries is called the time step. The time step depends on the sampling frequency of the telemetry collectors.

There are three main parameters of the window generator: input width, label width, and offset. At a fixed input width, the window generator gradually fills up its input window with sequential data. Providing a window of consecutive entries instead of one single entry to the ML models yields great results. This is because methods like LSTM or CNN can explore variations in two or more dimensions, as we will see later on. Window generator adds a dimension of time, allowing ML models to explore the latest trends of all input features. The width of the input window is a hyperparameter, that will be adjusted in the hyperparameter tuning process. In this architecture for data centers assurance, the ML models predict the state of the future at one time step. So the window generator's label contains one time step. The window generator's offset is the difference (in time steps) between the last element in the input sequence and the first element in the label sequence. This parameter controls how far into the future the model predicts and shall be adjusted according to the requirements of the prediction. The stream processor outputs windowed data to the ML models for time series forecasting.

The core of this IBN assurance architecture is the ML models for time series forecasting. For this particular work, the models are trained with historical network data and CPU usage percentage as the label. The reason for selecting this particular metric is because many actions in data center networks (VM migrations, upscaling, downscaling, energy optimization etc.) are often triggered by the change of CPU usages of the VMs. Therefore, the CPU usage is a good representation of the state of the VMs. The objective of these ML models is to forecast the surges in CPU usage in the near future, based on the latest network status in the windowed data and to satisfy the intent imposed by the users and data center providers. More details about the ML models will be provided later in this section.

In the next step, the forecasted results and the information of the data analysis are made available to analytics applications via the APIs of the analytics engine. When a surge is predicted, specific application(s) are notified by alerts from the analytics API. The produced analytics information can also be stored in the analytics data store, which also contains other information that is pertinent to the analytics engine or the analytics APIs (system logs, contextual cache, etc). The information in the data store is available to the ML models for training and prediction. Authorized analytics applications are also able to consult the data store's content via the analytics API.

Actuators in this IBN structure are elements that can change the network configurations and the network structure of the hosted services. According to the predictions from the ML models, network controllers can be instructed to take some actions. For example, when the ML model predicts that some VMs' CPU usages will have a surge up to 80% in fifteen minutes, OpenStack, a typical Cloud controller, can start the VM migrations and an SDN controller can modify the traffic routes accordingly. Actuators play an important role in the feedback because they take proactive actions to prevent utilization surges and packet drops.

B. ML Models for Time Series Forecasting

In this section, we present the ML forecasting models for the proposed network assurance model. The models take batches of two-dimensional input in the form of windowed data. In the case of IBN assurance for data center networks, the CPU usage percentage is used as the output label. The ML models predict the CPU usage percentage at a future time-stamp, in order to forecast utilization surges of the VMs. In the rest of this section, firstly, some baseline models will be presented, following we will describe the structure of the three ML models under consideration: a CNN model, an RNN model using LSTM, and a model using both CNN and LSTM.

1) Baseline models: Before building some trainable models, we need a performance baseline that will serve as a point for comparison. In this study, there are two baseline models: a naive model and a linear model. Our naive model simply returns the current label value as the prediction, predicting "no change". This is a reasonable baseline when there are not many fluctuations in the CPU usage percentage. The linear model inserts linear transformation between the input features and the output label. It assumes that the output label y (CPU usage percentage in a future time step) is linearly correlated to the current features X and a constant offset b. The linear model is described as $y = MX + b$, where M denotes the trainable weights of the features. Weights M and offset b are adjusted during training via gradient descent to reduce the Mean Square Error (MSE).

2) Convolutional Neural Network: CNN is a type of deep neural network that is mostly applied to image processing and computer vision. CNNs are based on convolution kernels (called filters) in a shared-weight architecture [16]. The CNN kernels slide along the input features, in order to generate feature maps that are translation equivalent. Compared to RNN models, convolution implementations are often faster and less resource-demanding. Because the windowed data of the time series forecasting problem is in a grid-like topology, CNN is ideal for handling this type of input. During the convolution and the pooling of the CNN models, the feature's spatial relations are explored. Because CNNs take advantage of the data's hierarchical pattern for regularization, they are less likely to overfit the data. In CNNs, convolution operations are performed in the hidden layers, where a layer calculates a dot product between the CNN's convolution kernel and a section of the layer's multi-dimensional input. The activation function in our CNN model is the ReLU function. The convolution operation generates a feature map by sliding the convolution kernel in the input matrix. Using convolutional layers reduces significantly the number of free parameters so that multiple convolutional layers can be stacked in the learning model. After each convolution layer, our proposed CNN model uses a max pooling layer. The goal of using pooling layers is to reduce the data's dimension. The max pooling in the proposed CNN model outputs the maximum value of each local cluster in the feature map. For the problem of forecasting CPU usage percentage in data center networks, we found out that the best configuration for the hidden layers is two convolutional layers, each followed by a dropout layer.

3) Recurrent Neural Network: Among many ML models, RNN is a top candidate to explore temporal behaviors from temporal sequences. An addition of using RNN to simpler feedforward neural network structures is that RNN keeps an internal state in the model [17]. The internal state assists the model to process input sequences of different lengths. LSTM is a type of RNN architecture, which has a great ability of handling sequences of data, like texts, videos, and time series data [19]. LSTM is therefore widely used in the applications of time series predictions. Compared to simpler RNN models, LSTM uses cells that are able to track information throughout many time steps with a controlled information flow. Compared to the previously mentioned CNN models, LSTM models often have more parameters to train due to their additional complexity. However, for our problem of forecasting CPU usage percentage in data center networks, LSTM yields much better results than CNN models. After extensive fine-tuning, we found out that the best configuration consists of two LSTM layers, each followed by a dropout layer.

4) CNN and LSTM: In some recent studies in the domain of time series forecasting, there are some promising results with a model structure that mixes some CNN and LSTM layers [18]. The idea of combining CNN and LSTM layers is to explore spatial patterns of the time series, with the help of the model's internal memory states. For the architecture that combines CNN and LSTM layers, the CNN layers are placed after the input layer. Their job is to produce feature maps from the input windowed data. After the CNN layers are the LSTM layers. So for LSTM layers, this configuration allows them to work on the feature map produced by CNN layers. Intuitively, the CNN section and the LSTM section split the forecasting task by exploring data from their own perspectives. Some studies are indicating that for some time series forecasting problems on certain datasets, the CNN-LSTM model produces better results than both the existing state-of-the-art LSTM models and the best-performing CNN models [19]. After careful examination, for our problem at hand, the best CNN-LSTM configuration we observed has two convolutional layers, each followed by a max pooling layer, then one LSTM layer with dropout.

IV. Performance Evaluation

A. Experiment Description

To test out our IBN assurance solution, we have conducted some experiments to verify our architecture design and evaluate the ML models. Our experiments were running on a MacBook Pro with a 2.3 GHz 8-Core Intel Core i9 and an AMD Radeon Pro 5500M graphics card. Our ML models are implemented with the TensorFlow framework.

In our experiments, we selected the GWA-T-13 MATERNA dataset, which is a high-quality and open-source dataset used in many ML research projects [15]. This dataset contains some performance metrics and resource utilization history of VMs in three Materna data centers. The traces in this dataset contain data from 520, 527, and 547 VMs over a period of three months. The traces for each VM are stored in a separate CSV files, and each CSV file contains about 8350 entries. The trace entries are organized in chronological order, from the oldest to the newest. Each entry in this dataset contains one timestamp column and the following 12 parameters: CPU cores, CPU capacity, CPU usage in MHz, CPU usage percentage, memory provisioned in KB, memory usage in KB, memory usage percentage, disk read throughput in KB/s, disk write throughput in KB/s, disk size in GB, network received throughput in KB/s, and network transmitted throughput in KB/s.

For labels, we use the column CPU usage percentage. For feature columns, we keep these eight columns: CPU capacity, memory provisioned, memory usage percentage, disk read throughput, disk write throughput, disk size, network received throughput, and network transmitted throughput. The column of CPU cores is not used because it is constant for each VM. The column memory usage in KB is removed because it is directly proportional to the memory usage percentage.

In our experiments, the Keras tuner API is used for hyperparameter tuning with the hyper band tuner. It tunes the number of filters in CNN layers, the number of units in LSTM layers, and the dropout rate in dropout layers. Apart from the hyperparameters in the model layers, there are two additional hyperparameters: the data window's input width and the learning rate of the Adam optimizer. The Adam optimizer in our implementations uses stochastic gradient descent. The tuning process can be extremely resource intensive and time consuming when the dataset is large because the Keras tuner builds multiple models with different hyperparameter combinations. Due to the hardware limitations, hyperparameter tuning is carried out on a subset of the entire dataset. 20 CSV trace files were chosen, with a total of 167,015 entries.

After the hyperparameter tuning, we trained the tuned model with more data from the Materna dataset. The maximum number of epochs is set to 100 with an early stopping callback, which stops the training if the validation accuracy does not improve during several iterations (patience, set to 5). The training also uses a checkpoint callback that saves the model's weights at the end of every epoch. At the end of all epochs, the final model uses the weights of the best epoch (the epoch that has the least validation loss). The models use the Adam optimizer with the optimal learning rates found during hyperparameter training.

B. Results and Discussions

The diagram in Figure 2 shows the trends of normalized validation loss during training for the three ML models under consideration. As it can be seen, the three proposed models do not have significant overfitting. The LSTM and CNN-LSTM model stabilize at about the same level, while CNN model stays in a range of higher loss. For the CNN and the LSTM models, there is a significant drop in validation loss during the first ten epochs. The LSTM model's loss, however, stays in a relatively small range compared to others. We can also see that the CNN-LSTM model has a smoother validation loss than the CNN and the LSTM model. According to the checkpoint callback, the best LSTM model corresponds to the model after 11 epochs, the best CNN model is the model after 14 epochs, and the best CNN-LSTM model is selected after 17 epochs.

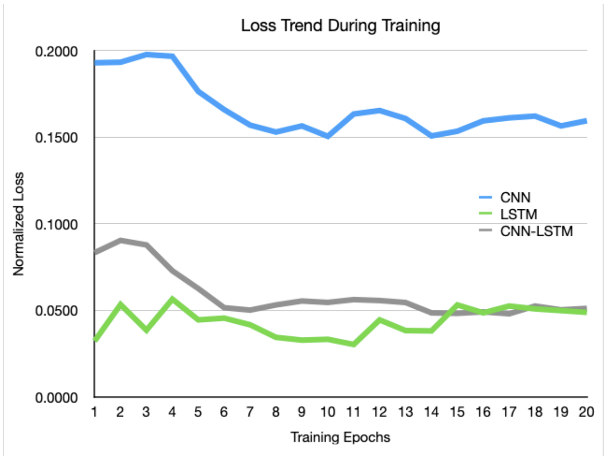

Fig. 2. Loss Trend During Training

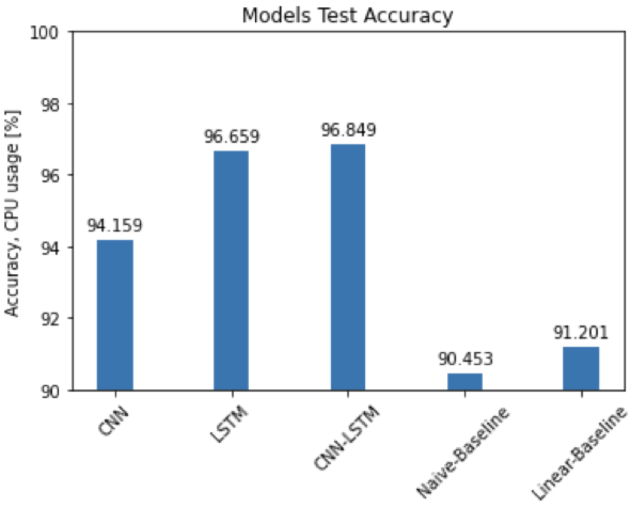

Fig. 3. Models Test Accuracy

Because different ML models have very different complexity (number of parameters to train), the training time can vary a lot. For large data center networks, there are often enough resources to carry out the model training. However, large models that are expensive to train can be problematic for small networks with a pool of limited resources. In our experiments, the two more complex models, LSTM and CNN-LSTM, took four to five times longer to train than the CNN model. To address this issue, we applied an early stopping callback with smaller patience.

After the model training, a test dataset is created using a different section of data from the training and validation dataset. All the proposed models (CNN, LSTM, and CNN-LSTM) and the baselines (naive baseline and linear baseline) are evaluated using the same test dataset. During our tests, the models predict one time step (five minutes) into the future. Accuracy is calculated by comparing the predicted values from the models and the expected label values from the test dataset. Figure 3 shows the test accuracy of the proposed models and baselines.

Both of the baseline models' accuracies are around 91%. The CNN model and the CNN-LSTM model are the best-performing models, with an accuracy of about 96%. The CNN model clearly beats the baselines, but its accuracy lies much below the level of the other two models.

When evaluating the performance of the ML models, another very important factor to consider is the prediction time. For instance, some ML models with high prediction accuracy may take much longer to produce prediction results. We evaluated the prediction time of the three proposed models making 1000 predictions. The CNN model is more than three times faster than the LSTM model and more than six times faster than the CNN-LSTM model. Despite the accuracy of the CNN model being two to three percent worse than the LSTM and CNN-LSTM model, it makes its predictions way faster than the rest. At the same time, although the LSTM model and the CNN-LSTM model have very similar accuracies, the LSTM model is twice as fast as the CNN-LSTM model. The additional complexity of the CNN layers in the CNN-LSTM model makes it slower to make predictions, without any significant accuracy gains.

Based on our previous results, we have the following main observations. For networks that need the ML model to make predictions as fast as possible, the CNN model is the best choice. These types of networks may host some low latency applications that value reactivity over accuracy. Networks with less strict time constraints may also benefit from using CNN models as backup models when they are under heavy traffic load and the resource pressure is very high. When high accuracy is the priority of the IBN assurance engine, the LSTM model is the best choice. Its prediction accuracy is as good as the CNN-LSTM models' but it is about twice as fast. However, when considering other factors like the training time and tuning difficulty, the best overall model is the LSTM model. Specifically, the LSTM model is neither very costly to train nor extremely hard to tune. It has the performance advantage over the CNN model and the simplicity advantage over the CNN-LSTM model.

As a last experiment, we want to evaluate various prediction shifts, which correspond to how far the model predicts into the future. In this experiment, we increase the "shift" parameter of the data windows, so the label of a window is further ahead of the features in the time series. The diagram in Figure 4 illustrates the trends of test accuracy of the three proposed models as the prediction shift increases.

Among the proposed models, the CNN model and LSTM model do not have much drop in accuracy as the prediction shift increases, compared to the CNN-LSTM model. When we push the CNN model and LSTM model from predicting 5 minutes into the future to 35 minutes into the future, their accuracy drop is for both less than one percent. The CNN-LSTM model is one of the models that has the best one-step prediction accuracy, but this experiment shows that its accuracy drops much faster as it predicts farther into

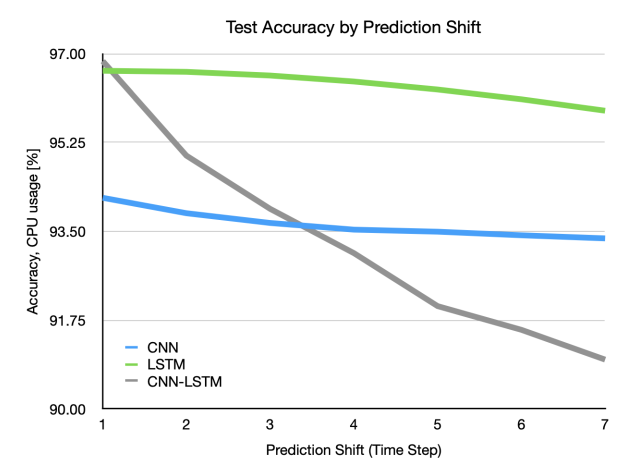

Fig. 4. Models Prediction Shift

the future. For one-step prediction, the CNN-LSTM model outperforms the CNN model, but its accuracy drops below the CNN model's accuracy after four time steps. The CNN-LSTM model even cannot beat the baseline models after eight time steps. This performance drop may be caused by the fact that two types of learning layers are used.

Thus, our recommendation is that for short-term predictions of one or two time steps, the LSTM and the CNN-LSTM model are the best candidates because of their high initial accuracy. But the CNN-LSTM model should be avoided when predicting multi-time-step CPU usage, because of its accuracy drop. In conclusion, the best-performing model from the prediction shift results is the LSTM model.

V. Conclusion

In this study, we investigated different aspects of IBN as a networking solution, and various network assurance approaches using ML techniques. The main contribution of this study is that we proposed a network assurance solution for data center IBN networks. Specifically, we proposed a high-level architectural design of the IBN assurance engine architecture, where at the very center of this solution is the network assurance model. This last is composed of some specific data preparation procedures and ML models for time series forecasting. Accordingly, we proposed three types of deep learning models: CNN models, LSTM models, and hybrid CNN-LSTM models.

Our results, based on the MATERNA dataset, illustrate that the LSTM and CNN-LSTM models are much more complex than the CNN models. All of the three models have good prediction accuracies of above 90%. However, the best-performing models are LSTM and CNN-LSTM, with accuracies above 96%. Although the CNN model has lower accuracy, it can make predictions much faster than the other two models. Furthermore, the accuracies of the CNN and the LSTM models do not drop significantly when predicting farther into the future, compared to the CNN-LSTM model.

Finally, regarding our future directions, we aim to study additional IBN network types and their impact on the proposed architecture. We expect that the architecture design for the assurance engine and the ML model structure may be different from what we proposed for data center networks. Addressing these scenarios requires a different approach, with more suitable network and statistical features and possibly a different data preparation pipeline.

References

[1] A. Clemm et al., ”Intent-Based Networking - Concepts and Definitions,” IETF, Network Working Group, 2019.
[2] Cisco Public, ”Intent-Based Networking Building the bridge between business and IT,” 2019, https://securenetworkers.com/2018/08/31/intent-based-networking/.
[3] C. Li et al., ”Intent Classification,” IETF Network Working Group, 2019.
[4] T. Szigeti, D. Zacks, M. Falkner, S. Arena, ”Cisco Digital Network Architecture Intent-based Networking for the Enterprise,” Ed. 1, Cisco Press, 2019.
[5] M. Medvetskyi, M. Beshley, M. Klymash, ”A quality of experience management method for intent-based software-defined networks,” IEEE 16th International Conference on the Experience of Designing and Application of CAD Systems (CADSM), 2021, pp. 59–62.
[6] M. Savi, D. Siracusa, ”Application-aware service provisioning and restoration in SDN-based multi-layer transport networks,” Optical Switching and Networking, vol. 30, pp. 71-84, 2018.
[7] K. Tanabe, T. Fukuda, T. Kuroda, ”Automated Performance Evaluation of Intent-based Virtual Network Systems,” 16th International Conference on Network and Service Management (CNSM), 2020, pp. 1-7.
[8] A. Leivadeas and M. Falkner, ”VNF Placement Problem: A Multi-Tenant Intent-Based Networking Approach,” 24th Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN), 2021, pp. 1-8.
[9] C. E. Rothenberg et al., ”Intent-based Control Loop for DASH Video Service Assurance using ML-based Edge QoE Estimation,” 6th IEEE Conference on Network Softwarization (NetSoft), 2020, pp. 353-355.
[10] T. A. Khan, A. Mehmood, J. J. Diaz Ravera, A. Muhammad, K. Abbas, W. Song, ”Intent-Based Orchestration of Network Slices and Resource Assurance using Machine Learning,” IEEE/IFIP Network Operations and Management Symposium (NOMS), 2020, pp. 1-2.
[11] ETSI GR ZSM 005, ”Zero-touch network and Service Management (ZSM); Means of Automation”, Tech. Report, 2020, https://www.etsi.org/deliver/etsi_gr/ZSM/001_099/005/01.01.01_60/gr_zsm005v010101p.pdf
[12] L. Wang, D. T. Delaney, ”QoE Oriented Cognitive Network Based on Machine Learning and SDN,” IEEE 11th International Conference on Communication Software and Networks (ICCSN), 2019, pp. 678-681.
[13] L. Velasco et al., ”Autonomous and Energy Efficient Lightpath Operation based on Digital Subcarrier Multiplexing,” IEEE Journal on Selected Areas in Communications, pp. 1-1, 2021.
[14] T. A. Khan, A. Mehmood, J. J. Diaz Rivera, W.-C. Song, ”Machine Learning Approach for Automatic Configuration and Management of 5G Platforms,” 20th Asia-Pacific Network Operations and Management Symposium (APNOMS), 2019, pp. 1-6.
[15] Materna. (2020). GWA-T-13 Materna [Online dataset]. Consulted at http://gwa.ewi.tudelft.nl/datasets/gwa-t-13-materna.
[16] W. Zhang, K. Itoh, J. Tanida, Y. Ichioka, ”Parallel distributed processing model with local space-invariant interconnections and its optical architecture,” Applied Optics, vol. 29, no. 32, pp. 4790-4797, 1990.
[17] A. Tealab, ”Time series forecasting using artificial neural networks methodologies: A systematic review,” Future Computing and Informatics Journal, vol. 3, no. 2, pp. 334-340, 2018.
[18] C. Benzaid and T. Taleb, ”AI-Driven Zero Touch Network and Service Management in 5G and Beyond: Challenges and Research Directions,” in IEEE Network, vol. 34, no. 2, pp. 186-194, 2020.
[19] K. Abbas, M. Afaq, T. A. Khan, A. Mehmood and W. -C. Song, ”IBNSlicing: Intent-Based Network Slicing Framework for 5G Networks using Deep Learning,” 2020 21st Asia-Pacific Network Operations and Management Symposium (APNOMS), 2020, pp. 19-24.