2024 15th International Conference on Information, Intelligence, Systems &amp; Applications (IISA) | 979-8-3503-6883-3/24/$31.00 ©2024 IEEE | DOI: 10.1109/IISA62523.2024.10786660

An Incremental Learning Assurance Approach for Intent Based Networking enabled Data Centers

Steve Lévesque [*], Xiaoang Zheng [*], John Violos [*], Aris Leivadeas [*], Matthias Falkner [†]
[*] Department of Software and IT Engineering, École de technologie supérieure, Montreal, Canada
Email: steve.levesque.1@ens.etsmtl.ca, seanzheng21@gmail.com, ioannis.violos.1@ens.etsmtl.ca, aris.leivadeas@etsmtl.ca
[‡] Cisco Systems, Ottawa, Canada, Email: mfalkner@cisco.com

Abstract—**Intent Based Networking (IBN) is a novel paradigm that aims to create a closed-loop automation towards building a self-governed network. An inherent part of IBN is the network assurance, which tries to automatically detect any performance drift and propose a remedy solution. Following this trend, in this paper we propose a novel IBN-enabled network assurance approach in a data center setting, in order to predict key resource utilization metrics and to help network administrators to proactively take corrective actions when needed. Specifically, we propose an incremental learning approach for data center assurance, designed to accommodate the dynamic nature of infrastructures. Experimental findings obtained from real-world datasets substantiate the effectiveness of our approach, demonstrating its ability to accurately forecast resource utilization even amidst highly dynamic environments.**

Index Terms—**Intent Based Networking, Network Assurance, Machine Learning, Network Automation**

I. Introduction

As the network infrastructure grows and the versatility of new applications with strict Quality of Service (QoS) guarantees is amplified, there is a great need for autonomous tools that will allow the self-configuration of the network. This need can be fulfilled by the introduction of closed-loop automation systems, such as Intent Based Networking (IBN), that aim to create a self-governed network [1]. IBN aims to reduce the gap between user knowledge and network capabilities, by allowing users to express their high-level network requirements in a declarative way, called intent, without using any technical jargon [2]. Following, an IBN system can autonomously translate the user utterances into low-level network policies and deploy the intent into the network fabric.

An inherent part of IBN is network assurance, where through the use of telemetry tools the QoS of an already deployed intent is constantly monitored in order to reveal any performance drifts [3]. Accordingly, in this paper, we focus on a proactive assurance approach within a data center network, where telemetry tools closely monitor the resource utilisation of Virtual Machines (VMs) that were provisioned during the translation of the intent. With this information an IBN system can autonomously take a corrective action (i.e., VM migration, load balancing, resource scaling) or generate an alert to the network administrator in order to decide the best preventive action to be taken, when a performance drift is detected.

Current data-center assurance approaches usually leverage Machine Learning (ML) techniques aiming to estimate the resource utilisation of a single data center environment. For example, the authors in [4], treat an intent as an e-commerce service and monitor the CPU, memory, and bandwidth utilisation of a web server. Subsequently, they train a Neural Network (NN) and predict the available bandwidth that should be allocated to assure the availability of the service. Long Short-Term Memory (LSTM) models have also been proposed as a possible NN to predict the resource utilization of a VM, in order to trigger the scaling of the VM's resources [5]. LSTM can be also combined with Convolutional Neural Networks (CNN) to enchance the temporal prediction of VM resources over longer time windows [6].

Even though these approaches can yield significant accuracy and an efficient assurance, they are considered static and cannot be easily operationalized across multiple data centers. This can refrain an IBN-enabled assurance product to be directly applied in another data center or to be effortlessly extended in a multi-cloud deployment. Thus, the main goal of this study and the driving idea behind its novelty is to offer a way of limiting the accuracy drift that may occur when a domain shift happens by moving a model from an old environment to a new one with only the previous environment's knowledge. This event can happen if there are no historical data on the new environment because it has never been explored yet. Hence, the contributions of this paper are threefold: i) we propose a proactive IBN assurance ML-based technique in a data center network, ii) we experimentally show that an environment switch can inversely affect the prediction accuracy, by using two real-world datasets from two heterogeneous data centers, and iii) we propose for the very first time and to the best of our knowledge in the context of IBN assurance, a dynamic Incremental Learning (IL) technique allowing the assurance ML models to be quickly adapted in the new environment/data center leading to significant performance gains.

The remainder of this paper is structured as follows. Section II presents the selected static ML techniques, while Section III delves into the details of the new proposed and dynamic approach that is based on the IL methodology. Section IV discusses the obtained results. Lastly, Section V concludes the paper and proposes some future work directions.

This work was supported in part by the Cisco University Research Program of Silicon Valley Community Foundation 2021-234759 (3696)

II. Static Approach for Network Assurance

In this study, we use the two most efficient ML models found in the literature for our problem at hand; an LSTM which is an improved version of the regular Recurrent Neural Network (RNN) [5] and a hybrid model of CNN and LSTM, known as CNN-LSTM [6]. The models are trained with historical VM utilisation data to make predictions on the CPU usage that can be later used to detect any QoS violations.

A. Long Short-Term Memory

RNN models are suitable to explore temporal behaviors from temporal input. An RNN uses a memorization technique to help with processing inputs with variable lengths. This memorized internal state helps RNN models to produce better results than models with simpler feedforward NN structures [7]. Commonly used in time series prediction problems, LSTM is a type of RNN architecture that is suitable to process sequences of data in the form of videos, texts, and time series [5]. In addition to a simpler RNN structure, LSTM cells apply controls to information flow by tracking information across multiple time steps. In this study, our LSTM models have larger parameter counts than our CNN models, that will be subsequently presented. However, the LSTM models' additional complexity brings better results than our CNN models. In this work, the best results come from a configuration that consists of two LSTM layers and two dropout layers.

B. CNN-LSTM

In some recent time series forecasting studies, best-performing models have structures that mix some CNN and LSTM layers [6]. To explore and interpret the spatial patterns in time series data, this hybrid CNN-LSTM model leverages not only a convolution kernel that explores spatial patterns but also an internal memorization state. In this combined architecture, the input layer is followed by CNN layers, which aim to generate feature maps from input windows. The LSTM layers are placed after the CNN layers, which enables them to process the feature maps from the previous CNN layers. For the dataset in this study, the best CNN-LSTM model accuracy comes from a structure of two convolutional layers, each followed by a max pooling layer, followed by an LSTM layer with a dropout layer.

III. Incremental Learning Approach for Network Assurance in a Data Center

A. Static Initial Model

As stated before, ML models trained in one environment/data-center will most likely fail in a different new environment. At this time, there are no methods that can help mitigate the problem in a resource utilization prediction setting for assurance purposes, without having a large dataset from the new environment. For multiple reasons such as privacy, security, lack of technological maturity, or any area of use that is completely new (no historical data), it is generally a standard case to have no data to train the new model. Nonetheless, when applying a model trained on a previous environment into the new one, monitoring data typically flow at a high rate. However, the static approaches fail to leverage this influx of data to acquire updated insights, as they disregard all incoming information and make predictions with the static models. Thus, this paper assess this problem with the help of a dynamic model which is based on IL.

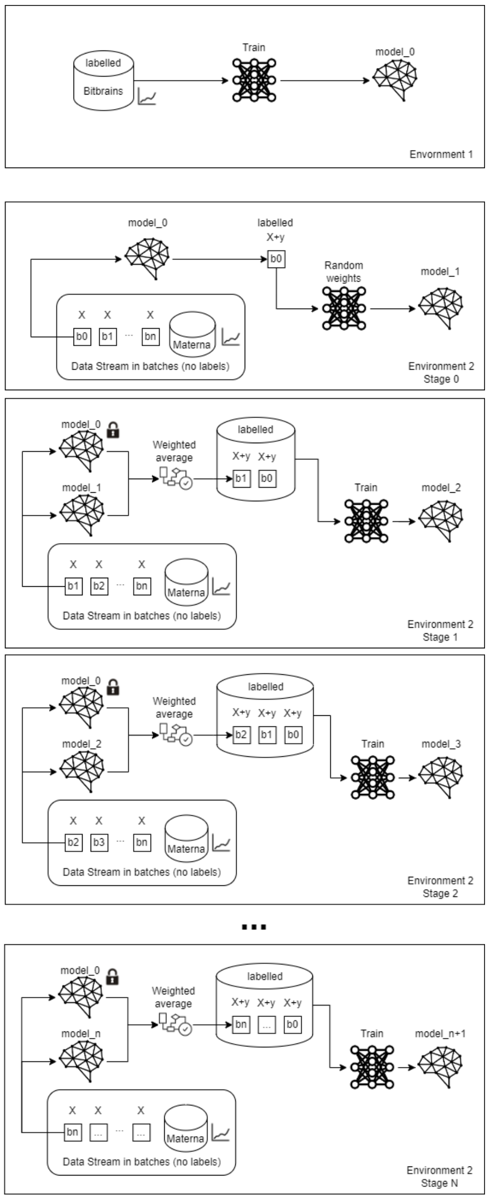


Fig. 1: Incremental Learning (IL) methodology

B. Incremental Learning Methodology

The proposed IL methodology, which we also mention it as dynamic, consists of retaining previously acquired models in order to learn progressively on new incoming data and enhance their performance results over time.

Fig. 1 shows the topology of the Dynamic approach with a transfer of a deep learning model from the Environment 1 to Environment 2. In the beginning, we see an initial model_0 that is trained with the historical data of the Environment 1. Next, we transfer the model_0 in the Environment 2. This model_0 predicts the first batch of data flows ("batch0") to allow model_1 to be retrained for the first time. As a sequence of monitoring data flows in the Environment 2, there is a model_n that incrementally learns in batches. In this case we use both models simultaneously (i.e., model_0 and model_n) and apply a weighted average on their predictions. For example, when the batch1 of monitoring data comes in the Environment 2, we can have a weight of 0.9 for model_0 and a weight of 0.1 for model_1. These weights gradually changes with the weight of model_0 to decrease and the weight of model_n to increase over time. This means that it decreases the importance of the model that comes from the previous environment and increases the importance of the model that incrementally learns in the environment. In this work, the reduction and increase takes place with a step of 0.1.

In the dynamic approach, every next model is trained from the old and new batches of labeled data streams. For the prediction of the new batch of data ("batch-n"), model_n-1 is replaced by model_n and combined with model 0 by making a weighted average prediction. Finally, the next re-training iterations are performed with the same logic until the weighting of model 0 becomes 0 and that of the dynamic 1.

IV. Experimental Evaluation

The programming language used for the experiments is Python (version 3.10). The framework used for implementing the ML algorithms and conducting experiments is Tensorflow with the ADAM optimizer. Also, Pandas is the main Python module used for most data manipulations like data cleaning. Specific hyper-parameters were used for the dynamic model in relation to the tested environment. For instance, we used 10 iterations, 1000 epochs, while the data throughput for a singular batch was set to 550 and the weight increase/decrease constant to 0.1. The iterations represent the batches count (b0, b1, ..., bn) in Fig. 1, meaning for the current experiment we have 10 batches in total that each have 550 data entries (time steps). With the same logic, the 1000 epochs are for each iteration separately for the training process on the new resulting model $n + 1$. Finally, the weight's increase/decrease constant of 0.1 is a multiple without remainder (clean division) that fazes out the static model at the last iteration (10th). Finally, the evaluation metric used to assess the attained performance is the Mean Absolute Error (MAE), which is calculated by the sum of the absolute differences between the predicted and ground truth values, divided by the size of the samples. The lower the MAE the better is the prediction.

A. Experimental Environments (Datasets)

For the experiments, two environments are necessary. The first environment acts as the old data center that trained the static models. In contrast, the second environment is new to the model. For this reason, two real datasets have been used. Specifically, the Bitbrains dataset [8] is used for the Environment 1 and the Materna dataset [9] for the Environment 2. Materna hosts clients from 6 Business Lines including IT Factories, Digital enterprises, Government and Communications. Bitbrains hosts clients related to financial industries like Banks, Credit Card Operators (ICS), Insurers, etc. Both datasets have the same features, monitored every 5 minutes, including the VMs CPU utilisation that we try to predict. It is important to note that the two data centers are very heterogeneous in terms of the applications they host. This fact allows us to better show the accuracy drift that will be noticed when shifting from environment 1 to environment 2.

TABLE I: MAE for VM 1

	Last Batch		All Batches	
	CNN-LSTM	**LSTM**	**CNN-LSTM**	**LSTM**
Static	0.1021	0.0872	0.0730	0.0484
Dynamic	**0.0281**	**0.0221**	**0.0532**	**0.0213**

TABLE II: MAE for VM 2

	Last Batch		All Batches	
	CNN-LSTM	**LSTM**	**CNN-LSTM**	**LSTM**
Static	0.0586	0.0470	0.0839	0.0679
Dynamic	**0.0126**	**0.0135**	**0.0609**	**0.0240**

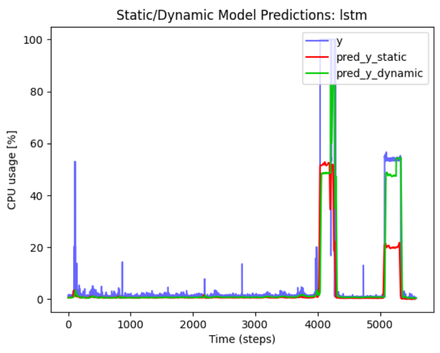
Materna and Bitbrains contain traces from 520 and 500 VMs respectively. However, in this work, the results of two VMs are shown. It is to be noted, that similar results were obtained in a large set of random VMs that we have selected to test. However, due to space limitations, the discussion is focused on these two VMs, but can be generalized for the rest of the infrastructure.

B. Outcomes

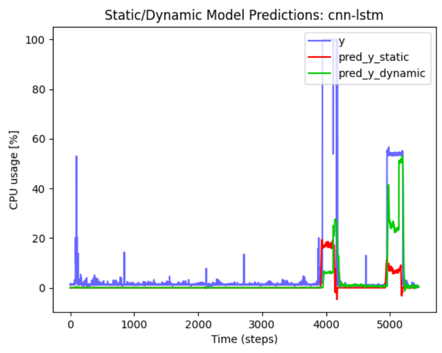
Tables I and II refer to the two VMs under consideration and show the performance of the models described in the previous section. The column "All Batch" summarizes the prediction errors during an experimentation of six hours. In contrast, the "Last Batch" column presents the errors during the last period when the dynamic model has ingested more than five hours monitoring data from the second environment. Additionally, for a better visual representation, Figs. 2 and 3 illustrate the static and dynamic predictions for the same VMs.

From the attained results, we can assess that both CNN-LSTM and LSTM models show a successful gain in performance with the dynamic IL approach in comparison to the static one. For example, all MAE results are lower for the dynamic approach than the static model, and for all VMs evaluated. At the same time, it appears that the LSTM is the best-performing model. A particularly interesting behavior is that CNN-LSTM sometimes outperforms (only by a few decimals) LSTM in specific use cases. For instance, we can see that for VM 2 (Fig. 3b and Table II), the trends at the far right (for the last batches and around the 5000th step) are predicted more efficiently with CNN-LSTM compared to LSTM. Nonetheless, LSTM predicts the high spikes better (around the 4000th step).

However, looking at VM 1 we could assess that CNN-LSTM is not a suitable model for such a trend because

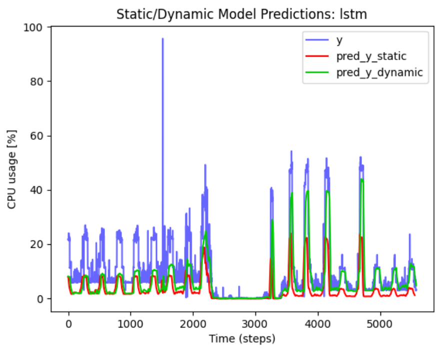


(a) LSTM predictions for VM 1

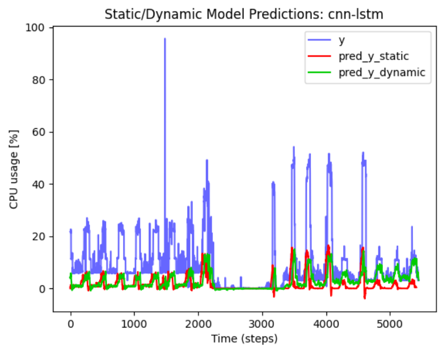


(b) CNN-LSTM predictions for VM 1

Fig. 2: CPU usage predictions for VM 1



(a) LSTM predictions for VM 2



(b) CNN-LSTM predictions for VM 2

Fig. 3: CPU usage predictions for VM 2

LSTM outperforms it completely. Furthermore, the IL approach shows better results at each new increment, as seen in Figs. 2a and 3a, where towards the end of the experimentation the predictions of the dynamic model are very close to the real values. Hence, persistent and sudden spikes of CPU usage that can affect the performance can be accurately predicted after few time instances, allowing the assurance component to promptly generate alerts or take proactively corrective actions by triggering resource scaling and VM migration techniques.

V. Conclusion

In this paper, we studied the problem of IBN assurance in a data center environment by introducing an IL approach to gain better predictions on a new environment that has no available historical data to be trained. In this case, a static model trained on a first environment is merged with new data generated by a second environment, allowing a smooth operationalization of the network assurance component. The experiment results, based on two realistic and heterogeneous datasets, showed that the proposed IL and dynamic model can offer a better prediction output than its static counterpart. Our future work involves enhancing the attained results in a larger scale experimentation, while also providing assurance of different intent scopes such as security and availability.

References

[1] A. Leivadeas and M. Falkner, "A Survey on Intent-Based Networking," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 1, pp. 625–655, 2023.

[2] A. Clemm, L. Ciavaglia, L. Z. Granville, and J. Tantsura, "Intent-Based Networking - Concepts and Definitions," Internet Engineering Task Force (IETF), Request for Comments RFC 9315, Oct. 2022. [Online]. Available: https://datatracker.ietf.org/doc/rfc9315

[3] A. Leivadeas and M. Falkner, "Autonomous network assurance in intent based networking: Vision and challenges," in *2023 32nd International Conference on Computer Communications and Networks (ICCCN)*, 2023, pp. 1–10.

[4] R. Hurtado, M. Torres, B. Pintado, and A. Muñoz, "Development of an intent-based network incorporating machine learning for service assurance of e-commerce online stores," in *Machine Learning for Networking*. Springer Nature, 2023, pp. 12–23.

[5] K. Abbas, T. A. Khan, M. Afaq, and W.-C. Song, "Network slice lifecycle management for 5g mobile networks: An intent-based networking approach," *IEEE Access*, vol. 9, pp. 80 128–80 146, 2021.

[6] X. Zheng and A. Leivadeas, "Network Assurance in Intent-Based Networking Data Centers with Machine Learning Techniques," in *2021 17th International Conference on Network and Service Management (CNSM)*, Oct. 2021, pp. 14–20.

[7] 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, Dec. 2018.

[8] "GWA-T-12 Bitbrains." [Online]. Available: http://gwa.ewi.tudelft.nl/datasets/gwa-t-12-bitbrains

[9] "GWA-T-13 Materna." [Online]. Available: http://gwa.ewi.tudelft.nl/datasets/gwa-t-13-materna