Liquid Neural Network-based Adaptive Learning vs. Incremental Learning for Link Load Prediction amid Concept Drift due to Network Failures

Omran Ayoub[1], Davide Andreoletti[1], Aleksandra Knapińska[2], Róża Goścień[2], Piotr Lechowicz[2,3], Tiziano Leidi[1], Silvia Giordano[1], Cristina Rottondi[4], Krzysztof Walkowiak[2]

[1]*University of Applied Sciences of Southern Switzerland, Switzerland* [2]*Wrocław University of Science and Technology, Poland*
[3]*Chalmers University of Technology, Sweden* [4]*Politecnico di Torino, Italy*

***Abstract*—Adapting to concept drift is a challenging task in machine learning, which is usually tackled using incremental learning techniques that periodically re-fit a learning model leveraging newly available data. A primary limitation of these techniques is their reliance on substantial amounts of data for retraining. The necessity of acquiring fresh data introduces temporal delays prior to retraining, potentially rendering the models inaccurate if a sudden concept drift occurs in-between two consecutive retrainings. In communication networks, such issue emerges when performing traffic forecasting following a failure event: post-failure re-routing may induce a drastic shift in distribution and pattern of traffic data, thus requiring a timely model adaptation. In this work, we address this challenge for the problem of traffic forecasting and propose an approach that exploits adaptive learning algorithms, namely, liquid neural networks, which are capable of self-adaptation to abrupt changes in data patterns without requiring any retraining. Through extensive simulations of failure scenarios, we compare the predictive performance of our proposed approach to that of a reference method based on incremental learning. Experimental results show that our proposed approach outperforms incremental learning-based methods in situations where the shifts in traffic patterns are drastic.**

***Index Terms*—Traffic Prediction; Adaptive Learning; Incremental Learning; Concept Drift; Network Failure.**

I. Introduction

Network traffic prediction represents a foundational problem in optical network design due to its significant implications for the overall performance and efficiency of the network [1], [2]. Currently, network operators perform traffic prediction and then feed forecast values as inputs to optimization algorithms responsible for fine-tuning the resource allocation and the service provisioning in the network [3].

To accurately predict network traffic, operators typically rely on machine learning (ML) algorithms. More specifically, operators train ML models to extrapolate future trends in traffic based on historical data reflecting past traffic patterns [1], [2]. However, the dynamic nature of traffic in networks, coupled with the continuous deployment of emerging services, introduces a plethora of challenges that have garnered widespread research attention. Among these challenges, continuous adaptation of ML models to align with evolving traffic patterns is of particular concern.

A potential solution to address this challenge is adopting a continual retraining process of ML models, as the most recent traffic measurements become available. While such an approach is effective under typical network (and thus, traffic) conditions, it fails to adapt in a timely manner under unforeseen network circumstances (and hence, under previously unseen network traffic conditions) for which the ML model was not originally trained. One such scenario arises during network faults, such as link failures, which can have cascading effects on traffic throughout the network, generating unprecedented patterns not encountered during the ML training process. In the context of ML, this shift in data patterns represent a phenomenon known as *concept drift*, in which trained ML models fail when faced with new and unexpected data distributions [4]. Fig. 1 shows the shifts in traffic patterns on a given link and the consequent distribution of traffic data (concept drift) resulting from a sudden decrease in traffic load due to a network failure (at time = 100) affecting another link. Note that an ML model developed to predict traffic is trained on data that shares a similar distribution to that seen prior to the failure.

The consequence of this shift in data distributions, or, in simpler terms, this *mismatch*, between ML model training data and data seen in failure scenarios may cause, depending on the magnitude of this mismatch, a drastic deterioration in prediction accuracy, rendering the employed ML model ineffective. The inability to accurately predict traffic under these circumstances poses a severe obstacle to network operators, as timely and precise predictions are paramount for expediting the restoration of network elements and associated traffic flows. This issue underscores the critical need for a specialized focus on traffic prediction under unforeseen scenarios, such as in the case of a failure of a network element, where conventional models and approaches fall short. An advanced approach to tackle this issue is by relying on incremental learning strategies, which involves updating or expanding a

O. Ayoub and D. Andreoletti are co-first authors of this paper. This work was supported by National Science Center, Poland under Grants 2018/31/D/ST6/03041 and 2019/35/B/ST7/04272, the NAWA STER Programme Internationalisation of Wroclaw University of Science and Technology Doctoral School and by the European Union under the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, partnership on "Telecommunications of the Future" (PE00000001 - program "RESTART").

978-3-903176-62-1

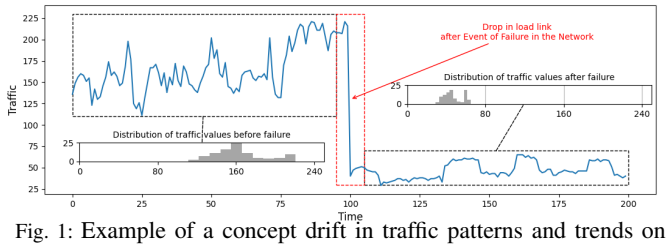

Fig. 1: Example of a concept drift in traffic patterns and trends on a link due to network failure (at time step 100).

model's knowledge over time without completely retraining the entire model. A primary limitation of this approach is its dependency on acquiring data that reflect the newly seen patterns for retraining. In a scenario of network failure, an operator must first collect an adequate amount of data prior to retraining and redeploying the model. The temporal gap between the occurrence of the failure and the reintroduction of the freshly trained model imposes a period of operational uncertainty, during which the operator cannot rely on an updated and reliable model. Note that frequent retraining, or retraining at shorter intervals, may not be a feasible solution for operators to adequately address the task at hand, considering that the quantity (and quality) of data used may be insufficient. In fact, identifying the *just-enough* quantity of data to collect (and hence, amount of waiting time) is crucial when adopting incremental learning approaches and merits considerable attention. However, even when this is determined, it does not eradicate the period of uncertainty that operators inevitably endure. Hence, employing methodologies that adapt seamlessly to shifts in data patterns under such failure scenarios, and consequently, allow operators to extract meaningful traffic predictions under a failure scenario until newly trained models are deployed, is essential.

In this work, we address the aforementioned challenge with a primary focus on the rapid adaptation of ML models for traffic prediction to drastic changes in traffic conditions arising from failure scenarios. Specifically, we propose a novel approach based on liquid neural networks (LNNs) [5], which can adapt to changes in data patterns without need for retraining. We compare the performance of our proposed approach to a reference method based on incremental learning, which performs retraining periodically. To conduct our experiments, we propose and employ a traffic model and a restoration mechanism and simulate dynamic network operations in the event of a network failure impacting network traffic patterns. We present a comparative analysis of the two approaches in terms of their predictive performance and the time required to provide predictions within a predetermined error threshold. Experimental results show that LNN-based approaches provide great utility in scenarios characterized by abrupt shifts in traffic patterns whilst incremental learning-based approaches undergo retraining. The results also highlight the preference for extended intervals of periodic retraining of incremental learning approaches when faced with moderate changes in traffic patterns.

The rest of the paper is organized as follows. Section II discusses related work. Section III formulates the problem of traffic prediction under failure scenarios and describes our proposed methodology to tackle it. Section IV describes traffic and network models adopted in our study. Section V reports the experimental settings and discusses numerical results. Finally, Section VI concludes the paper.

II. Background and Related Work

A. Addressing Concept Drift

Various techniques can be employed to address concept drift in ML. Retraining is a traditional technique that involves periodically updating ML models with the most recent data, allowing them to dynamically adjust to changing patterns. For instance, assuming that the data is a time series, *windowing* or *sliding window*, which is an approach that focuses on considering only a recent window of data for training, excluding older observations, can be employed. This technique enables models to swiftly adapt to the most recent patterns and minimizes the impact of outdated data, however, it still requires the availability of enough data for retraining. Another approach is *incremental learning*, which supports updating models with new data without the need for retraining on the entire dataset, thus facilitating a more immediate response to changing patterns, yet still requiring the availability of new data. On the contrary, *adaptive models*, such as online learning algorithms, are designed to inherently adjust their parameters as new data arrives, promoting continuous adaptation without the necessity for explicit retraining. In our work, we employ a novel online learning algorithm, i.e., LNNs, that does not require periodic updates to perform real-time adaptation.

B. ML-based Traffic Prediction

Existing works on traffic prediction propose various statistical and ML-based methodologies for enhancing the predictive performance of their algorithms on short- and long-term traffic evolution [6]–[8]. Since network traffic patterns and trends gradually change over extended periods, a specific focus should be given to dynamic approaches to predict previously unseen traffic conditions, as opposed to offline-learned models which fail to adapt in such cases [9]. In particular, online traffic forecasting algorithms using data stream mining techniques were proposed as an effective solution to enable a gradual model adaptation over long periods [9]–[12]. Despite their effectiveness, these approaches are not seen fit to cope with rapid and drastic shifts in traffic patterns (i.e., with concept drift). To address this issue, incremental learning-based approaches have been recently proposed [3], [13], with a specific focus on forecasting traffic after a failure event in the network. More specifically, [13] proposes an algorithm based on *moving windows*, demonstrating the trade-off between prediction quality and speed of adaptation. In [3] authors employ *partial fitting*, an incremental technique facilitating swift convergence after sudden traffic pattern changes due to network failure. This is achieved through model retraining with each batch of new data. While effective, the performance of this approach relies heavily on the batch size and retraining frequency, introducing

uncertainty when significant changes in input data patterns occur in short periods. In this work, we tackle the problem from a different angle and propose a novel approach based on adaptive learning algorithm to adapt, in real time, to new data patterns without the need for retraining. We quantify the achievable advantages and identify the scenarios where such an approach can benefit the network operator with respect to the above-mentioned benchmark methods.

III. Adaptive Learning and Incremental Learning for Link Load Prediction

The link load prediction task can be modeled as a regression problem which consists of forecasting the amount of traffic to be provisioned along a link in a future time step t, considering as input a set of p observations of historical traffic measurements on the link. The success of the regressor (i.e., the link load predictor) is assessed by quantifying the deviation of traffic predictions with respect to the actual values. In the event of a failure in the network, which causes a shift in data distribution, the efficacy of the regression method is then quantified by its ability to promptly adapt to changes in traffic patterns (we introduce a metric explained in detail in Sec. V-A, referred to as *Time to Convergence*, to quantify this adaptability). We consider two distinct approaches to address the problem at hand: i) incremental learning techniques (reference scenarios) and ii) adaptive learning algorithms (proposed method).

A. Reference Approach: Incremental Learning

The reference approach considered in this paper is based on the algorithm proposed in our previous work [3]. To create a model capable of adapting to changing traffic after failures, we developed an *incremental learning* approach based on data stream mining techniques. To this end, we employ a MultiLayer Perceptron (MLP) regressor that is periodically *partially fitted*. In more detail, the model is first trained on a number of traffic samples, and after enough new data arrives, it is updated to match the current traffic conditions. The size of the retraining window is a parameter that steers the frequency of the model partial fitting. As input features for the MLP learning algorithm we directly use raw data, i.e., the *p* previous traffic samples. For example, model *p3* implies that the prediction is made using three previous traffic samples as input features. Typically, streaming models utilize the *test-then-train* protocol, i.e., for a specified batch size, the model outputs its forecast for the entire new batch of data, and when the real data is available – it undergoes retraining. However, such a methodology does not allow for the direct use of previous samples as features, as they are not yet available to the model when making batch predictions. Therefore, the closest samples outside the prediction window act as the model inputs. In this work, we modify this approach to consent the best possible model adaptation after concept drifts. Specifically, the model predicts the traffic for the upcoming sample using *p* previous ones. After the batch of new data is available, it undergoes retraining.

B. Proposed Approach: LNN-based Online Learning

Artificial Neural Networks (ANNs) are ML models inspired by the structure of the mammalian brain, which is composed of neurons organized in interconnected layers. In traditional ANNs, neuron states are determined by linear combinations of inputs from other neurons, enhanced with specific non-linear activation functions (e.g., the sigmoid) to increase model expressiveness. A significant subtype of ANNs, specifically designed to model time-series data, is the recurrent neural network (RNN). Such a network is characterized by recurrent mechanisms, that enable neuron activation to be influenced by linear combinations of inputs from other neurons and their own previous states. However, traditional RNNs often face challenges in adapting to complex time-series dynamics. LNNs [5] represent a fundamental shift from traditional RNNs. Indeed, while still making use of recurrent mechanisms, LNNs explicitly model time-series dynamics through differential equations that determine neuron states. Specifically, the neurons' state is the solution to the differential equation presented in Eq. 1:

$$\frac{d\mathbf{x}(t)}{dt} = -\left[\frac{1}{\tau} + f(\mathbf{x}(t), \mathbf{I}(t), \boldsymbol{\theta})\right]\mathbf{x}(t) + f(\mathbf{x}(t), \mathbf{I}(t), \boldsymbol{\theta})\mathbf{A} \tag{1}$$

where $\mathbf{x}(t)$ is the vector representing the states of network neurons at time t, τ is a constant that ensures numerical stability, $\mathbf{I}(t)$ represent the inputs to the neurons at time t, f is a neural network parametrized by $\boldsymbol{\theta}$ and $\mathbf{A}$ is a bias term. This design offers more effective modeling of dynamic systems and possesses the remarkable capability to adapt LNN's behavior post-training, i.e., they can adjust to the dynamics of unseen inputs without the need for further training.

For this reason, LNNs are particularly useful in scenarios where sudden shifts in data distribution occur due to unforeseen events, such as a significant change in network traffic following a network failure. In such situations, network operators are required to respond quickly, basing their actions on the traffic load estimated by their models. However, models trained on prior data distributions may not provide reliable estimations after data distribution has changed. Typically, these models would necessitate re-training through incremental learning methods, which can only take place once an adequate volume of new data has been accumulated, thus further delaying network recovery. In contrast, models utilizing LNNs can adjust to novel data without the need for re-training, thereby facilitating a quicker response from network operators.

IV. Traffic and Network Model

A. Traffic Model

We use Euro28 topology (28 nodes, 82 directed links), which models the European core network [14]. *R* nodes selected based on real data[1] host a data center (DC) and provide anycast services/contents. We assume a continuous inter-DCs synchronization, which guarantees that each DC offers exactly the same content and can serve any of the interested clients.

[1] Available at http://www.datacentermap.com

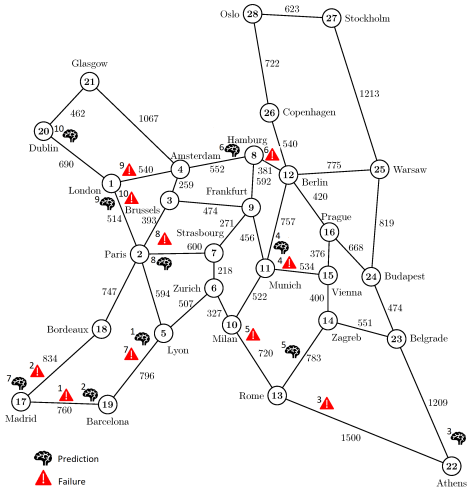

Fig. 2: Topology of the network and failure scenarios.

However, we always assign a client with the closest (according to the distance in kilometers) working DC.

We work under the assumption that the network traffic is a result of four transmission types, *i*) city to city: a basic unicast transmission between each pair of network nodes (representing cities), *ii*) city to DC: a service/content request and control data send from each city node to the closest working DC, *iii*) DC to city: a service/content provision and control data send from the selected DC to each city node, and *iv*) DC to DC: a unicast transmission observed between each pair of DCs; it realizes the inter-DCs synchronization.

To mathematically describe the network traffic, we use the model proposed in [15]. It models each transmission type (its time process) as a sine (trigonometric) function with parameters (amplitude, pulsation, initial phase) determined by the network economical, demographic and topological parameters. The entire traffic between a pair of nodes is then a sum of the sine functions related to the transmission types observed between that pair. Considering a simulation consisting in T-iterations (time steps), the model brings information about the traffic volume observed between each pair of network nodes for each time step $t \in T$. Within each time step of that time perspective, the average network load (the sum of all offered bitrate) is always equal to B [Tbps].

B. Network Model

Formally, a network is modeled as a directed graph $G = (V, E)$, where V denotes a set of nodes and E indicates a set of directed fiber links. Each fiber link offers the same spectrum range divided into S frequency slices. Adjacent slices can be grouped then into spectral channels, each one characterized by the first slice index and the channel width.

We consider a dynamic network operation, which refers to the allocation of new arriving demands and resource release after expired demands at each time step. A demand is given by a tuple $d = (s, t, b, h)$, where $s(d)$ and $t(d)$ are demand's source and destination nodes, $b(d)$ is a demand's volume (bitrate) in Gbps and $h(d)$ is a demand holding time. Note that all demands are unicast, since the DC selection task is solved off-line and known in advance. The applied traffic model (see Sec IV-A) provides data regarding the total traffic bitrate observed between each pair of communicating nodes at each time step $t \in T$. To translate these series into sets of demands arriving at each time step, we make use a special method introduced in [16]. At each time step $t \in T$, it iterates through all pairs of communicating nodes and divides the offered bitrate into a set of arriving demands. To this end, it takes into account all previously offered and still existing demands. The method also assumes that the maximum demand's bitrate can be 250 Gbps while their duration is randomly selected from the range $(0, 30]$ time steps.

To deploy a traffic demand, it is necessary to assign it with a lightpath (a routing path connecting the demand source and destination nodes and a channel tailored to the demand's bitrate and path's length). A demand can be allocated only at the time step of its arrival. If the available spectrum is not sufficient to serve it, the demand is rejected.

We assume that optical channels are multiplexed in a flexible grid with a standard slice width of 12.5 GHz. An elastic transceiver operates at 28 Gbaud with an optical channel bandwidth of 37.5 GHz (i.e., 3 frequency slices) and can use one of four modulation formats: BPSK, QPSK, 8-QAM and 16-QAM. The supported bitrates and transmission reaches are 50 Gbps, 6300 km for BPSK; 100 Gbps, 3500 km for QPSK; 150 Gbps, 1200 km for 8-QAM; 200 Gbps, 600 km for 16-QAM. We allow for the usage of signal regenerators only if necessary, i.e., when a path length exceeds the modulation reach. To select a modulation and a routing path for a particular demand d, we use the distance-adaptive transmission rule [17]. It chooses the most spectrally efficient format that simultaneously minimizes the number of used regenerators.

The network operation is simulated within T subsequent time steps. In each of them, a set of demands arrives (and needs to be served) and a subset of already allocated demands expires (and releases resources). When a link failure occurs, the demands using that link (on their lightpaths) are affected and have to be restored. They are handled in the same way as new demands, however, their duration is shortened according to the time already spent in the network. The objective function is to serve as much bitrate as possible. However, the network load in this research is chosen to mitigate any demand blocking and to allow for full traffic restoration after a link failure.

V. Experimental Results

A. Experimental Settings

1) Failure Scenarios: We simulate 10 failure scenarios considering the topology depicted in Fig. 2 (see legend to identify failed link and inspected link pairs for each failure scenario). Upon a link failure event, the traffic restoration process takes place as described in Sec. IV-B. Based on shift in data distribution of the traffic load on an inspected link, we divide the above 10 cases into two categories: 1) *highly impacted* and 2) *moderately impacted*. Specifically, highly impacted (resp., moderately impacted) category consists of cases in which the inspected link suffers from a drastic

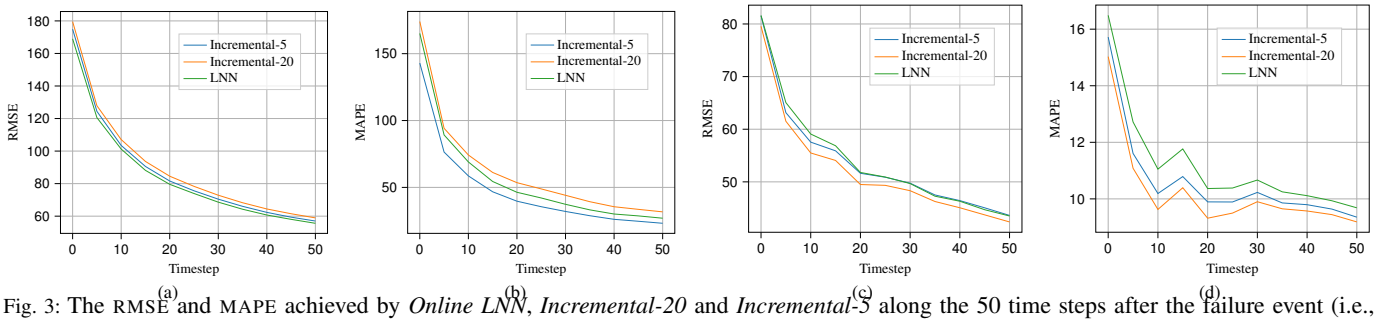

Fig. 3: The RMSE and MAPE achieved by *Online LNN*, *Incremental-20* and *Incremental-5* along the 50 time steps after the failure event (i.e., after concept drift) for the highly impacted (subfigures (a) and (b)) and moderately impacted (subfigures (c) and (d)) cases.

(moderate) variation of more (less) than 80% between the mean value of the 50 observations just prior to failure and that of the 50 observations just after the failure.

2) Model Parameters and Training: We consider a neural network composed of a first LNN layer with 30 neurons and hyperbolic tangent activation function, followed by 3 dense layers with a linear activation function and number of neurons of 10, 5 and 1 (i.e., for the output layer), respectively.

We use the same LNN architecture for all links. In all cases, we train the LNN using first 6000 time steps of the available data. The LNN is designed to take, at time step t, the past 3 observations from the previous three time steps ($t-3$, $t-2$ and $t-1$) to predict the traffic at time step $t+1$. Note that these 6000 observations (time steps) correspond to normal traffic conditions (prior to failure, with no concept drift). The LNN does not undergo any further training. We refer to this approach as $Online-LNN$.

For the Incremental Learning approaches, we use an MLP with one hidden layer of twenty-five neurons, with the *ReLU* activation function and *adam* optimizer. Similarly to the case of LNN, we design the model to take as input, at time step t, the past 3 observations from previous three time steps ($t-3$, $t-2$ and $t-1$) to predict the traffic at time step $t+1$. We consider two variations of this model. The first undergoes batch retraining every 5 time steps while the second undergoes batch retraining every 20 time steps. We refer to them as $Incremental-5$ and $Incremental-20$, respectively.

3) Evaluation Metrics: We consider three metrics to evaluate the performance of the proposed approaches, namely, the *Root Mean Square Error* (RMSE), the *Mean Absolute Percentage Error* (MAPE) and the *Time to Convergence* (TConv).

The RMSE evaluates how well the predicted values of the various approaches align with the actual observed traffic values in terms of their absolute values, penalizing larger errors more heavily, while the MAPE evaluates the percentage deviation between predicted and actual observed traffic values, to quantify the accuracy in a relative sense. In our work, we compute the RMSE and the MAPE of the various approaches considering different time intervals, in terms of time steps, after the failure. This allows us to measure how the deviation between the predicted values of an approach and the actual values differ as we move farther from the failure. In other words, it allows us to identify when incremental learning methods, which undergo training, start to provide more accurate predictions than the LNN-based online learning approach.

The *TConv* is defined as the amount of time needed by an approach to provide x number of predictions that are all within a predefined *th* percentage error. We assume that *th* represents a percentage error in the prediction that is tolerated by the operator, or in other words, a percentage error that minimally impacts the network operations that leverage the predictions. This allows us to identify when the incremental learning approaches provide reliable predictions and hence, the operator can again rely on them instead of LNN-based online learning approach. In our analysis we consider a fixed value of $x=5$ and two distinct th: $th \in \{10, 15\}$.

B. Numerical Results and Discussion

We start our discussion by comparing the performance of the three approaches in terms of RMSE and MAPE considering the period after failure, in particular from the moment of failure (time step 0) to 50 time steps after failure (time step 50), as we aim to investigate how the various approaches react after the occurrence of concept drift. Figs. 3(a) and 3(b) show the RMSE and the MAPE of the three approaches in the highly impacted cases. In terms of RMSE, LNN outperforms the incremental-based approaches directly after failure (at time step 0), showing an RMSE substantially lower (better) than *Incremental 20* (168 instead of 180) and slightly lower than *Incremental-5* (168 instead of 172). LNN consistently outperforms incremental-based approaches in the subsequent 50 time steps, even though the incremental-based methods underwent multiple partial refitting processes up to that point. Specifically, *Incremental-5* underwent 10 partial refitting processes, and *Incremental-20* underwent two such processes. These results unveil two main findings in case of drastic change in traffic patterns: i) LNN exhibits a consistent edge, albeit marginal, over incremental-based approaches. This advantage persists even when the incremental learning-based approaches undergo an intensive retraining process (every 5 time steps), for a substantial duration (e.g., up to 50 time steps), and ii) frequently performing partial refitting (considering a batch of 5) offers an improvement in the predictive quality of the models. In terms of MAPE (Fig. 3(b)), results show that *Incremental-5* outperforms LNN and *Incremental-20*. While the results confirm the second finding observed when analyzing the performance in terms of RMSE, they oppose the first one. This can be explained considering that the MAPE measures a relative error (i.e., to the actual traffic value), while the RMSE measures an absolute error. Hence, large absolute errors may

be absorbed by high reference traffic values, and are therefore more evident from the RMSE than from the MAPE. We observe that, in the context of traffic allocation, absolute errors are more relevant than relative ones. For instance, the same MAPE value is more impactful on a large traffic flow than on a small flow, as the actual traffic difference (e.g., the RMSE) is much greater in the former case. In turn, underestimating or overestimating high traffic volumes (i.e., having a large RMSE but possibly a small MAPE) could lead to significant problems such as congestion, under-utilization of resources, or even service outages. The choice of the metric to serve as foundation to select the approach to be adopted should be determined by the network managers, considering their specific objectives and network conditions.

Figures 3(c) and 3(d) report the RMSE and MAPE of the three approaches for the moderately impacted cases. Results show that, in terms of both metrics, *Incremental-20* outperforms the LNN and *Incremental-5* directly after the failure (i.e., before *Incremental-20* has undergone a partial refitting process) and throughout the considered period (up to time step 50). The LNN, although very comparable to *Incremental-5* in some cases, fails to edge any of the incremental learning-based approaches. We observe that in case of a relatively moderate change in data patterns, incremental learning approaches maintain their reliability over LNN. This is attributed to their ability to leverage the knowledge accumulated through partial refitting, allowing them to adapt more effectively to the changing dynamics when the latter are not very drastic. It is important to note that in such scenarios, achieving better performance is observed when partial refitting is conducted less frequently, using a larger batch size of 20, as opposed to a more frequent approach with a smaller batch size of 5. This is expected, as larger batch sizes in partial refitting allow the model to accumulate knowledge over a more extended period before updating, thus potentially capturing more stable patterns in the data and reducing overfitting to recent (and most likely, short-term) fluctuations (which are in fact very likely to occur just after the network has experienced a link failure).

We now compare the approaches in terms of the *TConv*. Table I reports the *TConv* for each approach across the 10 cases, considering values of x and th as reported in Sec. V-A. We first note how *TConv* varies drastically among failure scenarios (e.g., for $th = 10$, in failure scenario #5 the various approaches require 32 time steps to converge while in failure scenario 6 they converge from the first time step). This highlights the complexity and variability of the problem. Comparing the various approaches, for $th = 10$, LNN shows the best *TConv* in 6 out of 10 scenarios (in some scenarios exhibiting the same *TConv* of incremental learning approaches). For $th = 15$, LNN shows the best *TConv* in 50% of the cases, achieving convergence in almost half the time with respect to the other approaches (e.g., failure scenario #5). Despite LNNs efficacy in some scenarios, they struggle to outperform incremental learning in the rest of the scenarios. Therefore, a hybrid approach exploiting the strengths of each individual approaches could represent a promising solution.

TABLE I: TConv achieved by the various approaches across the 10 failure cases (see Fig. 2) considering two distinct values of *th*: 10 and 15, and for x = 5.

th	Approach	Failure Scenario ID									
		1	2	3	4	5	6	7	8	9	10
10	*Incremental-5*	14	49	**2**	**10**	**32**	**1**	2	8	**4**	**41**
	Incremental-20	**6**	**21**	144	17	**32**	**1**	17	8	5	**41**
	LNN	9	**21**	35	18	**32**	**1**	**1**	**5**	**4**	74
15	*Incremental-5*	7	**2**	**2**	2	32	**1**	**1**	**4**	3	10
	Incremental-20	**6**	7	34	2	31	**1**	**1**	5	**1**	**9**
	LNN	**6**	7	14	**0**	**17**	**1**	**1**	5	3	41

VI. Conclusion

In this paper, we addressed the concept drift issue in traffic patterns arising due to network failure for machine learning-based link load prediction. To this end, we propose an adaptive learning approach based on the use of liquid neural networks to adapt to changes in traffic patterns without requiring any retraining. As reference scenario, we design incremental learning-based approaches that undergo partial refitting periodically. We compare our LNN-based proposed approach to incremental learning-based approaches in terms of quality of their predictions and time to adapt to newly-seen traffic patterns. Our results show that LNN-based approaches can come in handy in circumstances of drastic change in traffic patterns, whilst incremental learning-based approaches can be retrained and adapted. Results also reveal that a larger interval for periodic refitting is desirable when change in traffic patterns is relatively moderate. Overall, our results provide network managers with valuable insights for machine learning-based traffic prediction in case of network failure.

References

[1] R. Vinayakumar *et al.*, "Applying deep learning approaches for network traffic prediction," ICACCI 2017.

[2] N. Ramakrishnan *et al.*, "Network traffic prediction using recurrent neural networks," ICMLA 2018.

[3] A. Knapińska *et al.*, "Link load prediction in an optical network with restoration mechanisms," *JOCN*, 2023.

[4] J. Lu *et al.*, "Learning under concept drift: A review," *Trans. KDE*, 2018.

[5] R. Hasani *et al.*, "Liquid time-constant networks," AAAI 2021.

[6] I. Lohrasbinasab *et al.*, "From statistical-to machine learning-based network traffic prediction," *Trans. on Emerg. Telecomm. Technol.*, 2022.

[7] G. O. Ferreira *et al.*, "Forecasting network traffic: A survey and tutorial with open-source comparative evaluation," *IEEE Access*, 2023.

[8] D. Andreoletti *et al.*, "Network traffic prediction based on diffusion convolutional recurrent neural networks," INFOCOM WKSHPS 2019.

[9] A. Knapińska *et al.*, "Long-term prediction of multiple types of time-varying network traffic using chunk-based ensemble learning," *Applied Soft Computing*, 2022.

[10] A. Shahraki *et al.*, "A comparative study on online machine learning techniques for network traffic streams analysis," *Comp. Netw.*, 2022.

[11] W. Liu *et al.*, "Multiclass imbalanced and concept drift network traffic classification framework based on online active learning," *Engineering Applications of Artificial Intelligence*, 2023.

[12] M. Balanici *et al.*, "Classification and forecasting of real-time server traffic flows employing long short-term memory for hybrid e/o data center networks," *JOCN*, 2021.

[13] R. Goścień *et al.*, "Efficient network traffic prediction after a node failure," ONDM 2022.

[14] S. Orlowski *et al.*, "Survivable network design library," INOC 2007.

[15] R. Goścień, "Traffic-aware service relocation in software-defined and intent-based elastic optical networks," *Comp. Netw.*, 2023.

[16] R. Goścień, "Traffic-aware dedicated path protection in elastic optical networks," RNDM 2022.

[17] K. Walkowiak, *Modeling and optimization of cloud-ready and content-oriented networks*. Springer, 2016.