A Residual LSTM based Multi-Label Classification Framework for Proactive SLA Management in a Latency Critical NFV Application Use-Case

Nikita Jalodia*†, Mohit Taneja*, Alan Davy*†, Behnam Dezfouli‡
*Walton Institute for Information and Communication Systems Science,
Department of Computing and Mathematics, Waterford Institute Of Technology, Waterford, Ireland
†SFI CONNECT Research Centre for Future Networks and Communications, Ireland
‡Department of Computer Science and Engineering, Santa Clara University, Santa Clara, CA, USA
{nikita.jalodia, mohit.taneja}@waltoninstitute.ie, adavy@wit.ie, bdezfouli@scu.edu

2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC) | 978-1-6654-3161-3/22/$31.00 ©2022 IEEE | DOI: 10.1109/CCNC49033.2022.9700502

***Abstract*—We are witnessing an emergence of a new era of applications delivered via a paradigm of flexible and softwarized communication networks. This has opened the market to a wider movement towards virtualized applications and services in key verticals such as automated vehicles, smart grid, virtual reality (VR), Internet of Things (IoT), industry 4.0, telecommunications, etc. With an increasing emergence of verticals driven by the vision of low latency and high reliability, there is a wide gap to efficiently bridge the Quality of Service (QoS) constraints for the end-user experience. Most latency-critical services are over-provisioned on all fronts to offer reliability, which is inefficient in the long run. In this work, we present a Residual Long Short-Term Memory (LSTM) based multi-label classification framework for proactive SLA management in a latency-critical Network Function Virtualization (NFV) application use case. We compose a multivariate time-series forecasting model with multiple time-step predictions in a multi-output scenario, and associate a multi-label classifier for a granular prediction of individual Service Level Objective (SLO) violations for each step in the forecast horizon. The Residual LSTM approach achieves an improvement of 31.1% over the baseline on the forecast classification accuracy, and a 2.65% improvement on the interpolated average precision over the standard LSTM methodology.**

***Index Terms*—Network function virtualization, machine learning, deep learning, LSTM, multi-label classification, residual LSTM, prediction methods, quality of service, service level agreements, supervised learning, artificial neural networks, SLA**

I. Introduction

5G's usage scenario of ultra-reliable low-latency communications (URLLC) is further expected to extend in scope to a high-throughput ubiquitous global connectivity at scale, driving all major verticals towards a change [1]. Further, the next generation of communication networks continue to be driven by a fundamental restructuring in the way that the networks and services are deployed and delivered. Network programmability and softwarization are the key drivers of this change, and are delivered via the concepts of Software Defined Networking (SDN) and Network Function Virtualization (NFV) [2]. These continue to play a pivotal role in the vision of 6G, forming the backbone of flexible and intelligent networks [1]. SDN abstracts the underlying network while NFV introduces softwarization and decouples network functions from the underlying hardware, overall creating a hardware agnostic virtualized environment for network applications [3].

As a result of such a shift, the Cloud infrastructure is no longer host to just web based application services, but is also being extended for the next-generation of requirements that fuel futuristic application verticals [2]. This shift also includes verticals that previously relied solely on specialised hardware. A key example of such a sector is the telecommunications industry, which is driven by one of the oldest and most complex operational and business support systems to date [1]. Traditionally, with its specialised infrastructure, the telecoms realm has evolved towards a highly reliant service, with carrier-grade offerings guaranteeing a five-nines standard of availability [4], set to improve to a seven-nines standard in 6G [1]. To match this in softwarized environments, 5G deployments include the proposition of network slicing, clustering applications with similar demands in an appropriate Cloud environment [3]. This ensures the placement of latency-critical URLLC applications in a high availability slice, where resources are suitably provisioned to ensure reliability. While service operators come up with new scaling policies to match the demand facing the current generation of Cloud based application services, these are still a long way to go towards supporting latency-critical applications with high availability values that match the legacy systems with specialized infrastructure.

Further, efficiency and reliability are competing elements within an SLA, marking a trade-off between system usage and requirements [5]. While network operators and Cloud operators may resort to over-provisioning to match the high requirements for these latency-critical applications in a high availability network slice, such practices are inefficient in the

This work has emanated from research conducted with the financial support of (i) Science Foundation Ireland (SFI), co-funded under the European Regional Development Fund under Grant Number 13/RC/2077, and (ii) NGI Explorers Fellowship Grant (grant agreement no. 825183).

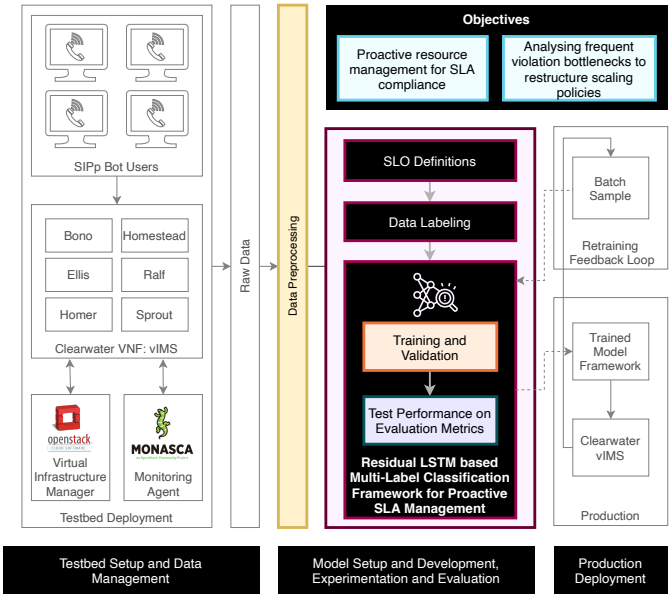


Fig. 1: Overview of system architecture, and objectives. The highlighted elements define the scope of our contribution.

long run [6]. A transition towards complete softwarization of networks and services brings in the requirement to adopt more complex models to guarantee QoS and reliability [7]. This is because of an impending evolution in not just the way networks are composed and managed, but also renewed application architectures, corresponding QoS and SLA management techniques, and optimization and automation to cope with the added complexity [8]. A key aspect to driving such a change is in how the Cloud reacts to such a latency-critical demand, and in being precisely proactive over time [6].

In this work, we present a machine learning based framework for proactive SLA management in the use case of a latency-critical NFV application. The key contributions are summarised as follows:

- We work with a real-world deployment of a latency critical NFV application with two months' worth of raw network telemetry data sampled every 30 seconds, and use that as the basis for all our policy formation and learning models. An overview of the system and scope is provided in Figure 1.
- We compose a multivariate time-series forecasting model with multiple time-step predictions in a multi-output scenario, i.e. the model forecasts a sequential range of future values for multiple features in one go, thereby enabling us to track a realistic deployment setup. Further, we propose the suitability of a residual connections based Long Short-Term Memory (LSTM) architecture when used for such a task, and compare the performance of multiple forecasting methodologies on such a use-case.
- While drafting the SLA, we decompose it into a set of realistic Service Level Objective (SLO) definitions for such a latency critical use-case in an operational setting. We categorise the SLOs into four broad characteristics that are critical towards the deliverance of required performance, and enhance these for a fine-grained monitoring of a latency-sensitive application that needs high availability and reliability. Further, we associate and model a multi-label classifier to effectively predict each of the multiple SLO violation categories that an application state can concurrently be associated with at an instance, i.e. as a multi-output prediction target. This helps in proactively predicting a more granular state of impact within an SLA violation projection.

To the best of our knowledge, this is the first approach in the area that proposes and applies a Residual LSTM based framework for proactive SLA management, and applies multi-label classification towards such predictive objectives.

The rest of the paper has been structured as follows: §II describes the background and related work, §III describes the Clearwater NFV application, and defines the SLA and SLOs drafted for the purpose of violation prediction. §IV provides an overview of the proposed framework. Thereafter, §V expands on the details of the experimental setup, §VI evaluates the results obtained through the various models, and §VII presents the conclusion and future work.

II. Related Work

Much of the work done so far addresses QoS with characterizing and anticipating traffic patterns, and a combination of reactive and proactive scaling policies. Significant progress has been made in the context of forecasting and clustering anticipated network traffic [9], [10], using machine learning to classify network traffic in NFV [11], [12], and related resource allocation [5], [6]. Authors in [13] present an overview of linear and non-linear forecasting methods, and discuss their use to improve multi-slice resource management in 5G networks. LSTM based approaches [13]–[15] have been successfully applied in the area of communication networks for resource forecasting. However, there have been very limited applications to the use of Residual LSTM in NFV [16], and that too have been in the domain of network slice reconfiguration. An in-depth survey [17] on the autonomic provisioning and QoS management for SDN-based networks highlights the need for more in-depth machine learning models that target and improve policy-based QoS management, and remark that assuring end-user QoE continues to be an open research area.

Automated SLA management for use-cases deployed on softwarized networks has been highlighted to be a critical requirement for next generation networks [18], [19]. A theoretical SLA management framework that maps high-level requirements to low-level resource attributes is presented in [20], where the authors highlight the additional challenges that 5G and future architectures present. Authors in [21], [22] present a cognitive management architecture for these softwarized networks, and discuss the importance of machine learning techniques in such complete end-to-end management control loops. Existing work on SLA and SLO violation prediction approaches it as a single label output classification

[23]— either identifying an overall SLA violation with a binary classification, or identifying a defined SLO breach with multi-class classification [24]. A proof of concept for SLA enforcement in programmable networks in a Cloud-based environment is presented in [25], where the authors work towards identifying an SLO breach with a multi-class decision tree classification methodology. However, in a realistic scenario, there is a pressing need for the incorporation of multi-output models as we move towards more complex decision-making [26]. As future networks as well as deployed services gain complexity, it is impractical to define and consider an SLO as a mutually exclusive single-output target. To fill this gap, we propose the use of multi-label classification methodology for a multi-output SLO violation prediction in NFV environments. Associating structured data with multiple semantic information at once holds tremendous potential in the future as we advance towards solving more complex decision making problems [26]. Our previous work in the area involves an in-depth analysis and benchmarking of multiple machine learning and deep learning methods employed towards multi-label SLA violation prediction in such a use-case, and also addresses the challenges of imbalanced classification often associated with real-world data in a multi-output target [27].

To the best of our knowledge, this is the first approach in the area that proposes and applies a Residual LSTM based framework for proactive SLA management in rapid forecasting based resource monitoring of latency sensitive NFV applications, and applies multi-label classification towards such target objectives.

III. Defining Service Level Agreements

While an SLA is a qualitative measure that binds the service provider and facilitator into a formally agreed contract ensuring QoS for the end user, this is realised on a set of low level metrics delivered through SLOs and Service Level Indicators (SLIs). The SLIs can be defined as quantitative measures that build upon raw system metrics, which further feed into the SLOs as a quantitatively definitive target range or threshold towards the deliverance of an SLA.

$$SLI \leq target\ threshold \tag{1}$$

$$lowerbound \leq SLI \leq upperbound \tag{2}$$

The breach of an SLA implies an explicit consequence, often financial; while the SLOs and SLIs are typically measurable indicators that define the policy of tolerance with measurable service characteristics [28].

A. Project Clearwater Cloud IMS

The IP Multimedia Subsystem (IMS) is a reference architecture first defined by the 3GPP for delivering fixed-line and mobile communications applications built on the Internet Protocol (IP) [29]. Project Clearwater[1] is an open-source implementation of IMS in the Cloud, following IMS architectural principles and supporting all of the key standardized interfaces

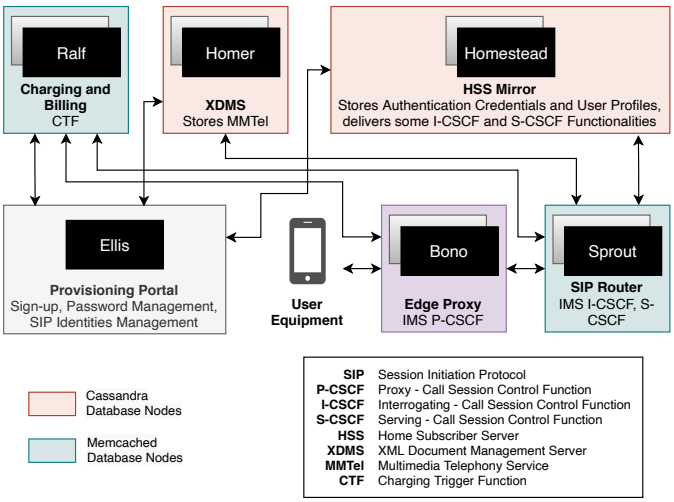


Fig. 2: Clearwater vIMS architecture, depicting the various VNFCs and their high-level functionalities.

expected of an IMS core network. The web services-oriented design inherent to Clearwater makes it ideal for instantiation within NFV environments as a virtualized VNF. The new Service-Based Architecture adopted by the 5G standards is very closely related to the inherent Clearwater model, and it has been widely used in research as a standard test-bed setup for NFV related work [4]–[7], [25].

In our work, we use Clearwater as the use-case for a Cloud based virtualized NFV application. It consists of 6 main components, namely Bono, Ellis, Homer, Homestead, Ralf, and Sprout. A high level view of these VNFCs and their functionalities replicating a standard IMS architecture is as shown in Figure 2.

B. Defining SLOs for Clearwater

We use raw network telemetry data and system monitoring metrics obtained via a standard realization of the Clearwater test-bed setup to define the SLIs and SLOs governing an informal SLA. We utilise these metrics as the foundations for the SLIs, which when matched with a target threshold or range form SLOs. These metrics were collected on a 30 second sampling frequency through Monasca[2], an open-source Python based monitoring service running on each of the Clearwater VNFCs. Further details regarding the data is elaborated upon in Section V.

We define four SLOs for the Clearwater VNF, targeting the load, computation, disk, and input/output (IO) characteristics respectively. This is to highlight the varying reason behind the loss of QoS at any time, so that:

1) The scaling decisions can be dynamically adapted with a high degree of detail, considering the projected forecasts. This helps towards the objective of proactive resource management for SLA compliance.

[1]https://www.projectclearwater.org

[2]www.monasca.io

2) The scaling policies can be customised at a more granular level towards better efficiency, upon analysing long term trends of frequent SLO bottleneck categories as specific to the application.

Authors in [25] recognize the lack of realistic SLOs in consideration in research, and recommend that an SLO be composed of a combination of atleast two metrics. To set a fair ground for our analysis, we define the SLOs with this definition in mind, and use a combination of over two SLIs while drafting each SLO rule. The thresholds were largely defined based on the application's usage characteristics, reaction to stress tests, and use-case requirements.

1) *SLO_1: Load*: This SLO is a measure of the computational work ongoing, and captures the running processes— either using the CPU, or in a wait state.
2) *SLO_2: Computation*: This SLO is defined as a combination of certain CPU and RAM characteristics combined with the idle time profile, which overall characterizes an overload or malfunction.
3) *SLO_3: Disk*: This SLO captures prolonged periods of inefficient IO wait times when the CPU is otherwise idle, which indicates potential bottlenecks in the read/write operations accrued by the hard disk.
4) *SLO_4: IO*: This SLO captures the latency when interacting with IO devices, when there is a sudden and prolonged surge in incoming network traffic as compared to the moving average.

A detailed composition of the SLO definitions can be found in our previous work [27]. Formally, let $\mathcal{L}$ denote the set of SLOs thus defined:

$$\mathcal{L} = [SLO_1, SLO_2, SLO_3, SLO_4] \tag{3}$$

This equivalently denotes:

$$\mathcal{L} = [SLO_{load}, SLO_{computation}, SLO_{disk}, SLO_{io}] \tag{4}$$

The metrics captured by Monasca are at the granularity of the individual VNFCs as shown in Figure 2, and an SLO violation at any of the individual VNFCs triggers an SLO violation state for the Clearwater application service. Therefore, we ultimately define the SLOs at the application level, i.e. for the entire VNF as an application service. Thus, each data instance is associated with 4 SLOs as defined by $\mathcal{L}$ above, where $SLO_j, j \in |\mathcal{L}|$ assumes one of two states:

$$SLO_j = \begin{cases} 1, & \text{if } Violated \text{ (at any VNFC)} \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

IV. Proactive SLA Management Framework

Given that the aim is to be able to proactively predict the future SLA violations given the time-series of tracked system and application metrics at a set sampling frequency, we decompose the problem as that of continuous feature forecasting, followed by a classification methodology that predicts the associated SLO violations in the forecasts. The workflow and pseudo-code of the methodology adopted is as depicted in Algorithm 1.

Algorithm 1: Residual LSTM based Forecasting and Multi-Label Classification

Input: Data: $\mathcal{D} \in \mathbb{R}^d$

• **Pre-Processing** ($\mathcal{D}$)

procedure Data Splits
  Split the data in train, test, and validation sets
end
procedure Data Transformation
  Data standardization and normalization
end
procedure Data Windowing and Batching
  Split data into windows of features and associated labels)
  < input width, all input features >
  < label width, predicted features >
  Prepare tensor slices of windows as model inputs
  < batch, time, features >
end

• **The Model** (output from Data Windowing)

repeat
  Forecasting– Residual LSTM
  Take train and validation data windows
    if Stacked LSTM model(Return Sequence) **then**
      Model architecture based sequentially stacked LSTM layers
    else
      Model architecture based LSTM layer
    end if
    for each time step t **do**
      delta = MODEL$_t$(MODEL$_{t-1}$)
    end for
    return (MODEL$_{t-1}$ + delta)
  Model architecture based Dense layer
  Model architecture based Reshaping layer
  Output forecast values
  Multi-label Classification
  Dense layer
  Reshaping layer
  Output multi-label classification prediction values
until convergence;

A. Clearwater Feature Forecasting

Artificial neural networks (ANNs) are powerful non-linear function approximators that are flexible to be adapted to both regression and classification tasks, and have demonstrated tremendous potential within the machine learning space. Each neuron within a layer represents a mathematical function comprising of inputs, weights, bias, and threshold; and uses an activation function to transform the outputs to a non-linear space to learn and perform more complex tasks [30]. Thus, subject to the right choice of architecture and parameters for the task at hand, ANNs can be trained to address a wide variety

of complex tasks, including that of time-series forecasting.

Recurrent neural networks (RNN) are a class of neural networks that are powerful for modeling sequential data such as time-series, and are especially crafted towards such use-cases [6]. An RNN layer maintains an internal state that encodes information about the time-steps it has seen so far. LSTM is a special enhancement on RNN, and overcomes the potential gradient vanishing and gradient exploding problems that RNNs tend to succumb to while training using back-propagation [31]. An LSTM layer consists of a set of recurrently connected memory blocks, known as LSTM units. Each LSTM unit is composed of a memory cell and three multiplicative units – the input, output and forget gates. These control the interaction of the cells with the network by regulating the flow of information in and out of the cell with continuous analogues of write, read and reset operations [30].

LSTM models are adept at capturing short-term and long-term dependencies in temporal sequential data, and have been shown to outperform linear models on a wide range of use-case scenarios [15]. Moreover, neural network based approaches like LSTM are well equipped to effectively model problems with multiple input variables, making them a good fit for multivariate time-series forecasting.

However, LSTMs are computationally expensive [13]. In use-cases such as ours with a high sampling frequency and rapid forecasting windows, the output is expected to be a small change as compared to the previous time-step. ResNets (Residual Networks) in deep learning refer to architectures where each layer adds to the model's accumulating result [32]. Adapting that structure into LSTM layer(s) [33], we can take advantage of the fact that the change at the next time-step is expected to be small. Thus, instead of predicting the next value of each feature at each time-step, a better approach to the model structure would be initialize the LSTM layer with the model's values from the previous time-step, and then to predict the subsequent change in these values over the next time-step. We reason that the LSTM layers in our use-case scenario can benefit from such a Residual LSTM model structure, and lead to better performance as opposed to traditional LSTM based models. Algorithm 1 presents the procedural workflow adopted towards achieving such a model. We evaluate this methodology in the following sections for both wide and stacked LSTM model structures.

B. SLA Violation Classification

Multi-label classification is defined as a classification task where each data sample instance can be assigned n labels from a set of $|\mathcal{L}|$ possible label classes as defined in 3 and 4, where $n \in [0, \mathcal{L}]$, and $|\mathcal{L}| > 1$. Each of the class labels in $\mathcal{L}$ is binary, i.e. either 0 or 1, where 0 denotes a negative occurrence and 1 denotes the positive occurrence.

Semantically, a multi-label target can be thought of as a set of labels for each sample. Multi-label classification differs from multi-class classification in that the latter applies mutually exclusive labels to a data sample, which is not the case for multi-label problems.

Formally, let $\mathcal{D}$ be a multi-label dataset where $\mathcal{X} = \mathbb{R}^d$ is a d-dimensional input instance space of numerical features, and $\mathcal{L} = \{\lambda_1, \lambda_2, \cdots, \lambda_q\}$ a finite output label space of $|\mathcal{L}| = q$ discrete class labels (with values 0 or 1), and $q > 1$.

The task of multi-label learning is to learn a function $f : \mathcal{X} \longrightarrow 2^{\mathcal{L}}$ from the multi-label training set $\mathcal{S}$ with u examples, $\mathcal{S} = \{(\mathbf{x}_i, Y_i) \mid 1 \leq i \leq u\}$. To compare, multi-class classification can be seen as a special case of multi-label classification where $f : \mathcal{X} \longrightarrow \mathcal{L}$, while in binary classification $f : \mathcal{X} \longrightarrow \{0, 1\}$.

For each multi-label example $(\mathbf{x}_i, Y_i)$, $\mathbf{x}_i \in \mathcal{X}$ is a d-dimensional feature vector $(x_{i1}, x_{i2}, \cdots, x_{id})^\top$, and $Y_i \subseteq \mathcal{L}$ is the set of labels associated with $\mathbf{x}_i$. Label associations can also be represented as a q dimensional binary vector $\mathbf{y}_i = (y_{i1}, y_{i2}, \cdots, y_{iq})^\top = \{0, 1\}^q$, where each element is 1 if the label is relevant, and 0 otherwise. By contrast, in single-label (binary or multi-class) learning, $|Y| = 1$.

Specific to the task at hand, we appropriately design the model such that the output layer consists of $|\mathcal{L}|$ neurons, each representing a label λ_j in $\mathcal{L}$, where $\mathcal{L} = \{\lambda_j \mid j \in [1, q]\}$. We use sigmoid as the activation function in the output layer, so the j^{th} neuron in that layer outputs the probabilities in the range $[0, 1]$ of the data instance belonging to λ_j. This is interpretable as a binary classification by setting a cutoff probability threshold value (set to 0.5) for each class label.

V. Experimental Setup

The experiments were all set up using Python (version 3.8.5) and its associated data-science libraries. We use Tensorflow [34] version 2.4.1 with Keras [35] to program all the neural network based implementations.

A. Dataset

We use a publicly hosted dataset[3] obtained via a standard Clearwater test-bed, a visualization for which is presented in 1. While real-world deployments require frequent retraining and readjustment of weights, the nuances of production deployment are beyond the scope of this work. The dataset comprises of raw system resource monitoring telemetric data files that track 26 metrics for each of the 6 monitored VNFs that compose the Clearwater ecosystem, and includes bursts of abnormal behaviour through its integrated stress testing tools to simulate VNF congestions and QoS degradations. The data is sampled every 30 seconds, and spans an overall period of 2 months. This corresponds to 156 features overall, indexed at timestamps, and 177,098 rows of raw data.

B. System Configuration

The experiments were performed in a Docker[4] based containerized environment running atop a bare-metal Linux server with 64 GB RAM, Intel® Xeon® CPU E5-2660 v2 @ 2.20GHz (40 physical processors), 2 NVIDIA® Tesla K20m GPUs, and 500 GB local storage. The Docker image runs an Ubuntu 20.04 LTS operating system, and CUDA version 11.3 for the GPUs.

[3]https://bit.ly/3gPY8c5

[4]www.docker.com

TABLE I: Performance metrics for the best performing model architecture within each category

Model Name	No. of Neurons/Units in the Key Hidden Layers in the Model Architecture	Classification Metrics						Regression Metrics	
		Accuracy	Precision	Recall	AUC-ROC	AUC-PRC	BCE Loss	MAE	RMSE
Linear	130	0.6488	0	0	0.5	0.3512	5.4176	0.3512	0.5926
Dense	2048	0.8500	0.9025	0.6424	0.8074	0.7955	2.0124	**0.1599**	0.3836
LSTM	2048	0.8500	0.9025	0.6424	0.7893	0.7776	1.9082	0.1625	0.3853
Residual LSTM	128	**0.8510**	**0.9082**	**0.6424**	**0.8082**	**0.7982**	1.6747	0.1647	**0.3810**
Residual LSTM (Stacked Sequence)	1024, 512, 256	0.8500	0.9025	0.6424	0.8069	0.7929	**1.3284**	0.1629	0.3839

C. Learning and Adaptation

We adopt a sliding window methodology for time-series forecasting, which is suitable for rapid forecasting. We tested the models on different window sizes, and considering the short-term dynamics of the use-case, the input window size was optimally set to 4, and the models forecast the possibilities of SLO violations over each of the next 4 time-steps. Thus, since the sampling frequency is 30 second intervals, we use the data for all the features over the last 2 minutes to forecast the specific SLOs that may be violated at each step over the next 2 minutes.

We split the available data into training and test sets in the ratio of $80:20$, and the training set is further split into training and validation sets in the ratio of $80:20$. Hence, overall, the data consisting of 177,098 rows is split in the ratio $64:20:16$, corresponding to train, test, and validation splits respectively. Further, given the data has a high degree of very large outliers due to the incorporated stress tests, and the scales vastly vary for each of the features depending on the category of raw metric, we use a non-linear transformation method to transform the very skewed nature of this dataset and map it to a uniform distribution in $[0, 1]$. This pre-processing is done via a Quantile Transformer, and this makes it suitable for learning by neural network methodologies.

For model training, we use Binary Cross-entropy (BCE) as the loss function to be minimized— a probabilistic loss function that computes the cross-entropy loss between true and predicted labels, and is appropriate for use in a binary classification based setup. Further, we use Nadam as the optimizer for its computational efficiency and adaptive learning, with its default learning rate of 10^{-3}. ReLu (Rectified Linear Unit) is used as the activation function for each of the dense hidden layers due to its computational simplicity and high optimization performance in a multi-layer perceptron (MLP) based setup [30]. For the LSTM layers, we use the default tensorflow initializations for weights, activation, and bias. As mentioned earlier, we use sigmoid as the activation function for the output layer to concurrently output the individual probability of each label's association with the input data instance, thus supporting multi-label classification outputs.

We used a grid search based methodology to arrive at the best configuration for the number of neurons/ LSTM units in each hidden layer as applicable, and to adjust all hyperparameters. To control overfitting as the model gains complexity, we apply the dropout regularization factor of 0.2 between all hidden layers. On the LSTM layers, we also applied a recurrent dropout of 0.4. To further control the degree of overfitting during training, we perform a grid search for the optimal choice of weight regularization hyperparameters for all the hidden layers, and based on the results, apply both L1 and L2 (ElasticNet) weight regularization on each of the hidden layers. The regularization factor was set to 10^{-6} for the LSTM models, and 10^{-5} for the dense feed-forward neural network models.

The batch size for each window was set to 64. While we set the maximum training epochs to 100, we also deploy an early stopping criteria that tracks the macroaveraged AUC-PRC (interpolated average precision) with a maximization objective, and a patience of 10 epochs to ensure that the training is not stopped at a local optimization minima. At the end of training, model weights are restored from the best epoch, which is considered as the best performance achieved during training, before the model began to overfit on the training set.

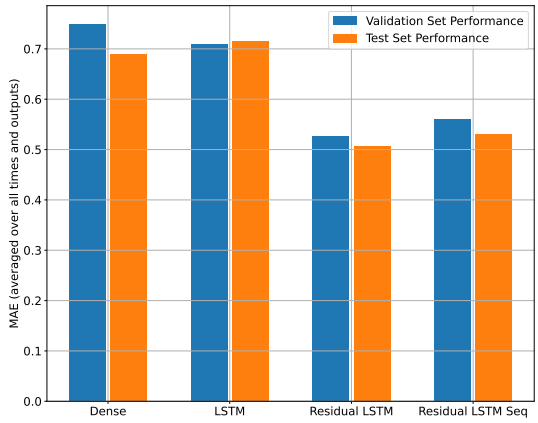
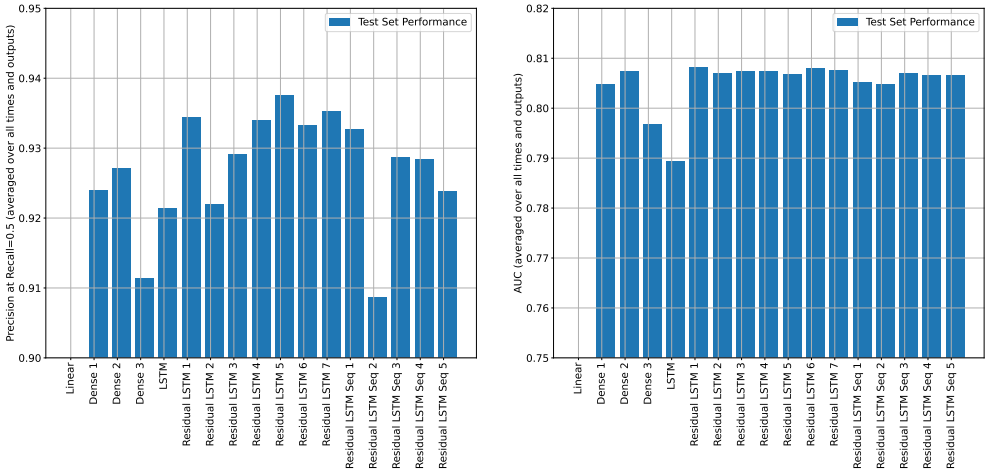


Fig. 3: Performance (MAE) of Residual LSTM models evaluated on their forecasting component against standard methodologies, when configured with the same wide model architecture.

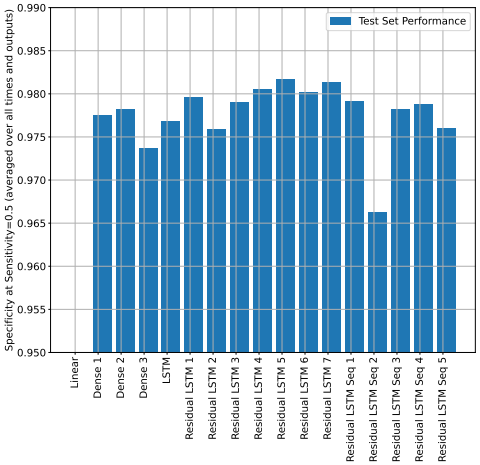
VI. Results and Discussion

Figure 3 presents an evaluation of Residual LSTM models against standard methods, when configured with the same wide model architecture (2048 neurons/LSTM units), and implemented purely as a forecasting component. The LSTM architecture when supplemented with residual connections has a clear advantage with the nature of the use-case here, both when the output horizon is forecast at the last time-step after going through all inputs, or sequentially. The latter among



(a) Precision at Recall=0.5

(b) AUC-ROC



(c) Specificity at Sensitivity=0.5

Fig. 4: Performance benchmarking of the varied model architectures and configurations, evaluated within each category.

them, however, has a higher Mean Absolute Error (MAE) due to the fact that the model architecture was wide, and sequential outputs of LSTM layers work best when layers need to be stacked in a deeper model.

Figure 4 shows the performance benchmarking of the varied model architectures and configurations as evaluated during grid search. Generally, models that leverage residual connections perform better in almost all model configurations when tested against standard methods. Table I presents the classification and regression performance metrics for the model architecture that performed best among those tested with each category of models on the test set data windows. The linear model makes linear projections based on the input window, and was used as a baseline to compare all models. With an area under the receiver operating characteristic (AUC-ROC) value of 0.5, it has the skill level of a random classifier. The results demonstrate the suitability of Residual LSTM based architectures against all other, on almost all metric categories in both evaluation groups. The Residual LSTM approach achieves an improvement of 31.1% over the baseline on the forecast classification accuracy, 127.28% on the interpolated average precision, and 61.64% on the AUC-ROC. It also showed a 0.63% increase in precision over the standard LSTM methodology, a 2.65% improvement on AUC-PRC, and 2.39% improvement on the AUC-ROC.

Further, given the use-case and the nature of the data, wide model architectures for LSTM demonstrate an edge over deep architectures. While adding more LSTM units in a layer tends to increase the chances of overfitting, we tackled those with appropriate regularization as applicable, as mentioned in the preceding section.

VII. Conclusion and Future Work

In this work, we compose a multivariate time-series forecasting model that forecasts the evolution of system monitoring features for the Clearwater VNF over the next 4 time steps, followed by a multi-label classification model that predicts the individual categories of SLO violations at each step over a 2 minute future horizon. We demonstrate the suitability of a Residual LSTM model over other MLP and LSTM based

methodologies in such a scenario that involves fine-grained rapid forecasting, and reason that the high level of granularity in predicting SLOs as multi-label outputs would help ensure a balance in precise provisioning while maintaining reliability in latency-critical NFV applications.

The methodology is transferable to other verticals within the high-availability network slice, and in or future work we plan to validate this on a different use-case. We also plan to extend the deployment to a bigger test-bed setup, aiming to incorporate the external network features by setting up the SDN block, and distributed application scenarios.

REFERENCES

[1] W. Jiang, B. Han, M. A. Habibi, and H. D. Schotten, "The Road Towards 6G: A Comprehensive Survey," *IEEE Open Journal of the Communications Society*, vol. 2, pp. 334–366, 2021.

[2] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, "Network Function Virtualization: State-of-the-Art and Research Challenges," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 236–262, 2016.

[3] T. Zhang, H. Qiu, L. Linguaglossa, W. Cerroni, and P. Giaccone, "NFV Platforms: Taxonomy, Design Choices and Future Challenges," *IEEE Transactions on Network and Service Management*, vol. 18, pp. 30–48, Mar. 2021.

[4] M. Di Mauro, G. Galatro, M. Longo, F. Postiglione, and M. Tambasco, "IP Multimedia Subsystem in a containerized environment: availability and sensitivity evaluation," in *2019 IEEE Conference on Network Softwarization (NetSoft)*, (Paris, France), pp. 42–47, IEEE, June 2019.

[5] R. Mijumbi, S. Hasija, S. Davy, A. Davy, B. Jennings, and R. Boutaba, "Topology-Aware Prediction of Virtual Network Function Resource Requirements," *IEEE Transactions on Network and Service Management*, vol. 14, pp. 106–120, Mar. 2017.

[6] N. Jalodia, S. Henna, and A. Davy, "Deep Reinforcement Learning for Topology-Aware VNF Resource Prediction in NFV Environments," in *2019 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*, (Dallas, TX, USA), pp. 1–5, IEEE, Nov. 2019.

[7] S. Cherrared, S. Imadali, E. Fabre, and G. Goessler, "LUMEN: A global fault management framework for network virtualization environments," in *2018 21st Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN)*, (Paris), pp. 1–8, IEEE, Feb. 2018.

[8] J. Suomalainen, A. Juhola, S. Shahabuddin, A. Mammela, and I. Ahmad, "Machine Learning Threatens 5G Security," *IEEE Access*, vol. 8, pp. 190822–190842, 2020.

[9] L.-V. Le, D. Sinh, B.-S. P. Lin, and L.-P. Tung, "Applying Big Data, Machine Learning, and SDN/NFV to 5G Traffic Clustering, Forecasting, and Management," in *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, (Montreal, QC), pp. 168–176, IEEE, June 2018.

[10] A. A. Gebremariam, M. Usman, and M. Qaraqe, "Applications of Artificial Intelligence and Machine Learning in the Area of SDN and NFV: A Survey," in *2019 16th International Multi-Conference on Systems, Signals & Devices (SSD)*, (Istanbul, Turkey), pp. 545–549, IEEE, Mar. 2019.

[11] J. Vergara-Reyes, M. C. Martinez-Ordonez, A. Ordonez, and O. M. Caicedo Rendon, "IP traffic classification in NFV: A benchmarking of supervised Machine Learning algorithms," in *2017 IEEE Colombian Conference on Communications and Computing (COLCOM)*, (Cartagena), pp. 1–6, IEEE, Aug. 2017.

[12] G. Ilievski and P. Latkoski, "Efficiency of Supervised Machine Learning Algorithms in Regular and Encrypted VoIP Classification within NFV Environment," *Radioengineering*, vol. 29, pp. 243–250, Apr. 2020.

[13] D. Ferreira, A. Braga Reis, C. Senna, and S. Sargento, "A Forecasting Approach to Improve Control and Management for 5G Networks," *IEEE Transactions on Network and Service Management*, vol. 18, pp. 1817–1831, June 2021.

[14] J. Bendriss, I. G. Ben Yahia, and D. Zeghlache, "Forecasting and anticipating SLO breaches in programmable networks," in *2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN)*, (Paris), pp. 127–134, IEEE, Mar. 2017.

[15] M. Abbasi, A. Shahraki, and A. Taherkordi, "Deep Learning for Network Traffic Monitoring and Analysis (NTMA): A Survey," *Computer Communications*, vol. 170, pp. 19–41, Mar. 2021.

[16] F. Wei, G. Feng, Y. Sun, Y. Wang, and S. Qin, "Proactive Network Slice Reconfiguration by Exploiting Prediction Interval and Robust Optimization," in *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, (Taipei, Taiwan), pp. 1–6, IEEE, Dec. 2020.

[17] A. Binsahaq, T. R. Sheltami, and K. Salah, "A Survey on Autonomic Provisioning and Management of QoS in SDN Networks," *IEEE Access*, vol. 7, pp. 73384–73435, 2019.

[18] C. Sun, J. Bi, Z. Zheng, and H. Hu, "SLA-NFV: an SLA-aware High Performance Framework for Network Function Virtualization," in *Proceedings of the 2016 ACM SIGCOMM Conference*, (Florianopolis Brazil), pp. 581–582, ACM, Aug. 2016.

[19] B. Yi, X. Wang, K. Li, S. k. Das, and M. Huang, "A comprehensive survey of Network Function Virtualization," *Computer Networks*, vol. 133, pp. 212–262, Mar. 2018.

[20] E. Kapassa, M. Touloupou, and D. Kyriazis, "SLAs in 5G: A Complete Framework Facilitating VNF- and NS- Tailored SLAs Management," in *2018 32nd International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, (Krakow), pp. 469–474, IEEE, May 2018.

[21] I. G. Ben Yahia, J. Bendriss, A. Samba, and P. Dooze, "CogNitive 5G networks: Comprehensive operator use cases with machine learning for management operations," in *2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN)*, (Paris), pp. 252–259, IEEE, Mar. 2017.

[22] J. Bendriss, I. G. Ben Yahia, P. Chemouil, and D. Zeghlache, "AI for SLA Management in Programmable Networks," in *DRCN 2017 - Design of Reliable Communication Networks; 13th International Conference*, pp. 1–8, 2017.

[23] M. Boucadair, C. Jacquenet, and X. Xu, eds., *Emerging Automation Techniques for the Future Internet:*. Advances in Wireless Technologies and Telecommunication, IGI Global, 2019.

[24] J. Bendriss, *Cognitive management of SLA in software-based networks*. Theses, Institut National des Télécommunications, June 2018. Issue: 2018E0003.

[25] J. Bendriss, I. G. Ben Yahia, and D. Zeghlache, "Forecasting and anticipating SLO breaches in programmable networks," in *2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN)*, (Paris), pp. 127–134, IEEE, Mar. 2017.

[26] D. Xu, Y. Shi, I. W. Tsang, Y.-S. Ong, C. Gong, and X. Shen, "Survey on Multi-Output Learning," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–21, 2019.

[27] N. Jalodia, M. Taneja, and A. Davy, "A Deep Neural Network-Based Multi-Label Classifier for SLA Violation Prediction in a Latency Sensitive NFV Application," *IEEE Open Journal of the Communications Society*, vol. 2, pp. 2469–2493, 2021.

[28] B. Beyer, C. Jones, J. Petoff, and N. Murphy, *Site Reliability Engineering: How Google Runs Production Systems*. O'Reilly Media, Incorporated, 2016.

[29] "3GPP - The 3rd Generation Partnership Project, A Global Initiative." https://www.3gpp.org/, accessed May 2021.

[30] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[31] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, pp. 1735–1780, Nov. 1997.

[32] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *arXiv:1512.03385 [cs]*, Dec. 2015. arXiv: 1512.03385.

[33] J. Kim, M. El-Khamy, and J. Lee, "Residual LSTM: Design of a Deep Recurrent Architecture for Distant Speech Recognition," *arXiv:1701.03360 [cs]*, June 2017. arXiv: 1701.03360.

[34] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Y. Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng, *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. 2015.

[35] F. Chollet and others, *Keras*. 2015.