Formal Verification for Deep Learning-based Mobile Network Traffic Prediction

Thanh Le*, Takeshi Matsumura*,
*National Institute of Information and Communications Technology (NICT), Japan

***Abstract*—Deep neural networks (DNNs) have emerged as a promising approach for mobile traffic prediction and capacity forecasting in next-generation wireless networks, leveraging newly developed architectures to capture spatiotemporal traffic demand for network resource provisioning and allocation. However, DNN-based traffic forecasting systems are vulnerable to adversarial attacks in which adversaries inject traffic perturbations via compromised devices, leading to erroneous capacity forecasts and misallocation. Existing defense mechanisms offer only empirical insights and lack formal guarantees, while neural network verification research has primarily focused on classification tasks, leaving regression problems such as mobile traffic forecasting unexplored. We address this gap by proposing a formal verification framework that formulates adversarial traffic injection as hyperrectangle input properties, converts recent deep learning traffic prediction models into a neural network verifier-compatible format, and leverages NeuralSAT to provide robustness proofs for which scenarios the system is robust against adversarial traffic injection. Preliminary proof-of-concept on the Telecom Italia Milan dataset demonstrates that our framework can formally guarantee whether DeepCog, a deep learning capacity provisioning model, is robust against adversarial traffic injection, providing network operators with peace of mind when deploying these models in production environments.**

***Index Terms*—traffic prediction, DNN, formal verification**

I. Introduction

Beyond 5G networks is currently facing the challenge of a growing number of users and devices, but the physical wireless resources are limited. Therefore, analyzing traffic and accurately forecasting user demands is essential for developing an intelligent network [1]. Network traffic prediction models operate in the background, analyzing historical traffic demands to forecast expected future needs, which can be leveraged by downstream network management services and network optimization tools [2], [3]. Machine learning and deep learning models can leverage vast amounts of network measurement data, exploiting temporal correlations in long historical measurements and spatial dependencies among connected nodes [4], [5], [6]. Given that network traffic exhibits complex relationships in both temporal and spatial dimensions, predicting future traffic volumes is a suitable task for deep learning models. These models require less domain knowledge and manual engineering than statistical methods such as Auto-Regressive Integrated Moving Average (ARIMA) [7].

However, the deployment of deep learning in network management raises fundamental security concerns, as deep neural network (DNN)s are vulnerable to adversarial attacks [8]. Such attacks involve introducing minor input modifications that can cause DNNs to produce erroneous predictions. In the context of network traffic prediction, adversaries may infiltrate smartphones or IoT devices within the network coverage area [9]. By orchestrating these compromised devices into a botnet, attackers can strategically inject minimal traffic volumes to corrupt the DNN's forecasting capabilities. These adversarial inputs are deliberately designed to evade anomaly detection mechanisms and remain within data usage constraints, yet they can substantially degrade the model's prediction accuracy.

Consequently, there is increasing interest in explainable deep learning approaches for critical applications such as network optimization and management that demand high reliability. Existing research [9] has investigated the robustness of DNN-based mobile traffic forecasting using explanable AI (XAI) methods, seeking to identify which input features, such as historical traffic demand from specific base stations, are most susceptible to adversarial manipulation. These studies use gradient-weighted class activation mapping (GCAM) and LayeR-wise backPropagation (LRP) techniques to pinpoint base stations that are particularly susceptible to traffic-injection attacks. XAI approaches offer correlational insights to inform adversarial training procedures for model retraining. Nonetheless, non-certifying defense mechanisms, such as adversarial training, have been shown to be circumvented by more sophisticated attack strategies [10]. To address this ongoing arms race, there is growing momentum toward defense strategies employing neural network verification (NNV) [11], [12], [13], [14], [15], which offer provable guarantees that DNNs remain resilient to attacks across all inputs within specified bounds. NNV complement XAI and adversarial training approaches by establishing the robustness of DNNs through rigorous mathematical proofs.

Existing NNV research has largely concentrated on verifying DNN-based systems within the Verification of Neural Networks Competitions (VNN-COMPs) framework [16], with limited exploration of DNN-based network management applications [8], [17]. The VNN-COMPs evaluation benchmarks primarily focus on classification problems in domains such as computer vision, natural language processing, and autonomous aviation systems. To the best of our knowledge, verification of DNNs for regression problems, including mobile network traffic forecasting, remains largely unexplored.

This work addresses this research gap by leveraging cutting-edge formal verification methodologies for DNN-based traffic forecasting systems. Our evaluation framework assesses whether trained DNNs meet performance specifications under diverse levels of adversarial perturbation. We begin by encod-

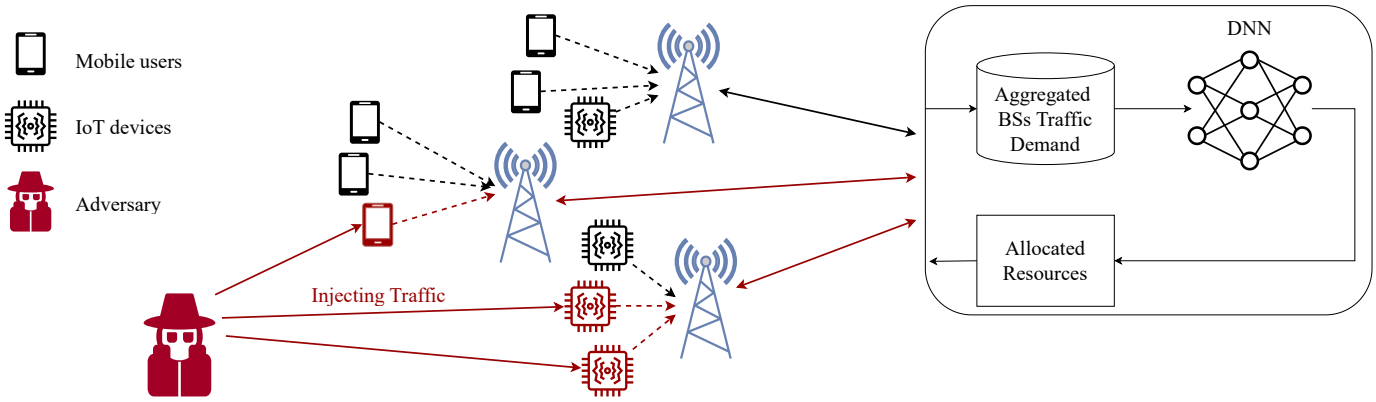

Fig. 1: Overview of system model of deep-learning-based mobile traffic prediction under adversarial traffic injections.

ing various adversarial perturbation levels as hyper-rectangle input constraints.

II. Preliminaries

A. System Model

A DNN is trained to forecast network traffic volumes at time step t using historical past traffic volumes from previous time steps. Formally, for a network slice s, let $\delta^s(t)$ denote the traffic snapshot at time t, which contains traffic demands at all base stations for slice s. The DNN takes T historical traffic snapshots $\{\delta^s(t-T), \delta^s(t-T+1), \ldots, \delta^s(t-1)\}$ as input and produces a capacity forecast for time t. Let $N(\cdot;\theta)$ be the DNN function parameterized by θ, which maps the sequence of historical snapshots to a capacity forecast $c^s(t)$. This framework for mobile traffic prediction is particularly beneficial for the management of next-generation wireless networks, where accurate capacity forecasting enables efficient resource allocation and network optimization.

DeepCog [18] is a popular DNN architecture specifically designed for capacity forecasting in network slicing scenarios. The architecture follows an encoder-decoder structure tailored to capture spatiotemporal patterns in mobile network traffic. The *input* to DeepCog consists of historical traffic snapshots $\delta^s(t-1), \ldots, \delta^s(t-T)$ for network slice s, where each snapshot $\delta^s(t)$ contains traffic demands at all base stations at time t. These snapshots are transformed into a 3D tensor (two spatial dimensions and one temporal dimension) that preserves traffic spatial-temporal correlations, enabling the 3D convolutional neural network to exploit them to accurately forecast future demands.

The *neural network architecture* comprises two main components: (i) an *encoder* with three 3D-CNN layers that extract spatiotemporal features from the input tensor, and (ii) a *decoder* with fully connected layers that generate capacity forecasts. The *output* of DeepCog is a capacity forecast $c^s(t) = \{c_s^1(t), \ldots, c_s^M(t)\}$ for network slice s at time t, where $c_s^j(t)$ represents the forecasted capacity needed at datacenter $j \in M$ to accommodate future traffic demands. DeepCog employs a *custom loss function* that explicitly balances the trade-off between resource overprovisioning and underprovisioning (service-level agreement (SLA) violations), rather than minimizing standard prediction errors like Mean Absolute Error or Mean Squared Error. The loss function is parameterized by $\alpha = \beta/\gamma$, where β is the fixed cost of an SLA violation and γ is the cost per unit of overprovisioned capacity. For a forecast error $x = c_s^j(t) - d_s^j(t)$ at datacenter j, where $d_s^j(t)$ is the actual demand, the loss function penalizes underestimation (SLA violations) with cost α and overestimation (overprovisioning) proportionally to the excess capacity. This design enables operators to tune the balance between overprovisioning and unserved demand to align with their economic priorities.

B. Threat Model

We consider an adversary who gains access to compromised networked devices capable of injecting traffic into a mobile network (Fig. 1). The attacker's primary objective is to disrupt normal network operations through denial-of-service attacks or, at a minimum, degrade the quality of service. Examples of such compromised devices include IoT devices and smartphones infected with malicious software.

The adversary possesses sufficient technical capabilities to infiltrate and compromise a subset of network-connected devices, thereby gaining the ability to associate with the network infrastructure. After establishing network connectivity, the attacker strategically injects traffic designed to cause service disruptions by introducing carefully crafted perturbations to the network load patterns. The feasibility of such attacks has been demonstrated by real-world incidents, most notably the Mirai botnet [19]. The adversary can compromise a fraction of the total devices, and once compromised, traffic injection becomes straightforward. The malware enabling these attacks can be distributed through various vectors: embedded in user-installed applications, distributed via official app stores, or incentivized by offering users a way to sideload applications from untrusted web sources. These distribution mechanisms have been extensively documented in the security literature [20], [21], [22].

We assume that the attacker does not necessarily possess knowledge of the trained DNN architecture or parameters used for traffic prediction. The injected traffic must remain minimal

to avoid detection, as the attacker is constrained by the need to prevent users from noticing excessive data consumption. Additionally, mobile data plans typically impose usage limits and incur costs, further constraining the volume of traffic that can be injected without raising suspicion. The attacker injects a limited amount of data to perturb the historical traffic measurements that serve as input to the traffic prediction model. These manipulated historical data points may fall outside the distribution of clean training data, causing the DNN to produce inaccurate future traffic predictions. The resulting prediction errors lead to either over- or under-provisioning of network resources, which, in turn, causes resource misallocation, service degradation, service disruptions, and network instability. When prediction errors persist over time, they can establish a continuous feedback loop with the resource allocation algorithm, amplifying long-term misallocation of resources.

C. NNV - An Overview

The *NNV problem* concerns determining whether a given property ϕ holds for a DNN N. Properties are typically expressed as implications of the form $\phi_{in} \implies \phi_{out}$, where ϕ_{in} specifies constraints on the inputs of N and ϕ_{out} specifies constraints on the outputs of N. This formulation enables encoding safety and security requirements for DNNs [15]. A DNN verifier searches for a *counterexample* input that satisfies the input property ϕ_{in} but causes the output to violate ϕ_{out}. If no such counterexample can be found, the property is unsatisfiable (`UNSAT`), indicating that the DNN is proven robust; otherwise, the property is satisfiable (`SAT`).

NNV methods can be broadly categorized into two classes: (1) probabilistic guarantees and (2) deterministic guarantees. Probabilistic approaches, such as randomized smoothing [12], provide high-probability certificates that a DNN remains robust to L_2-norm perturbations within a specified radius. In contrast, deterministic methods offer absolute guarantees of safety against any L_p-norm-constrained perturbations. Deterministic NNV techniques fall into three main categories: (1) constraint-based approaches [14], (2) abstraction-based approaches [11], [13], and (3) hybrid approaches [15]. Constraint-based methods can be computationally expensive, with solution time increasing significantly as the DNN size grows [14]. Abstraction-based methods address scalability by employing abstract domains such as polytopes (e.g., DeepPoly [11] and CROWN [13]) to enable verification of larger networks, though at the expense of some accuracy. The core principle of abstraction-based verification is to construct polyhedral over-approximations using linear inequalities that tightly bound the possible outputs of non-linear activation functions, such as rectified linear unit (ReLU). NeuralSAT [15] integrates constraint-based and abstraction-based techniques to extend the range of DNN sizes that can be verified.

III. Proposed Verification Framework

A. Local Robustness Properties Formulation

Following the threat model described in Section II, we assume that an attacker can inject additional traffic into the network through compromised devices in their botnet. The adversary's objective is to introduce carefully crafted perturbations to the historical traffic data fed into the DNN, thereby causing erroneous capacity forecasts that lead to resource misallocation and service degradation.

To formally verify the robustness of the considered DNN-based traffic forecasting scheme, we define input specifications $\phi_{in} := [\eta_L, \eta_U]$ to encode permitted adversarial traffic injection levels. Specifically, η_L and η_U are vectors having the same dimensionality as the input traffic snapshot $\delta^s(t)$, where each component represents the lower and upper bounds on the additional traffic that can be injected at each base station. The adversary seeks to generate an adversarial traffic snapshot $\delta'^s(t)$ within a constrained neighborhood of the original traffic $\delta^s(t)$ such that the DNN output becomes erroneous. This adversarial input is constructed by adding a bounded perturbation $\delta'^s(t) = \delta^s(t) + \eta$ to the original traffic snapshot $\delta^s(t)$, where $\eta \in [\eta_L, \eta_U]$ represents the adversarial noise corresponding to the additional traffic injected by the botnet. The input property ϕ_{in} thus constrains the perturbed traffic to lie within the hyperrectangle $[\eta_L, \eta_U] = [\delta^s(t), \delta^s(t) + \eta]$, ensuring that the verification covers all possible adversarial traffic injection scenarios within the specified bounds.

B. Output Properties

Recall that for a forecast error $x = c_s^j(t) - d_s^j(t)$ at datacenter j, where $c_s^j(t)$ is the forecasted capacity and $d_s^j(t)$ is the actual demand, DeepCog's loss function $\ell'(x)$ is defined as:

$$\ell'(x) = \begin{cases} \alpha - \epsilon \cdot x & \text{if } x \leq 0 \\ \alpha - \frac{1}{\epsilon} x & \text{if } 0 < x \leq \epsilon\alpha \\ x - \alpha\epsilon & \text{if } x > \epsilon\alpha \end{cases} \tag{1}$$

where $\alpha = \beta/\gamma$ is the ratio of SLA violation cost β to overprovisioning cost per unit γ, and ϵ is a small constant that enables gradient-based training.

To complement the robustness property, we construct two output properties that constrain the predicted capacity relative to actual demand:

1) *Overprovisioning Bound:* The predicted capacity should not exceed $(1 + \zeta)$ times the actual demand to avoid wasting resources:

$$\phi_{\text{over}}: \quad c_s^j(t) \leq (1 + \zeta) \cdot d_s^j(t) \tag{2}$$

2) *Underprovisioning Bound:* The predicted capacity should not fall below $(1 - \zeta)$ times the actual demand to prevent SLA violations:

$$\phi_{\text{under}}: \quad c_s^j(t) \geq (1 - \zeta) \cdot d_s^j(t) \tag{3}$$

Together, these properties enforce:

$$(1 - \zeta) \cdot d_s^j(t) \leq c_s^j(t) \leq (1 + \zeta) \cdot d_s^j(t) \tag{4}$$

This dual-property approach provides a practical guarantee: avoiding excessive overprovisioning while ensuring sufficient capacity to meet user SLAs.

C. Verifier

To solve the verification problem formulated with input property ϕ_{in} and output property ϕ_{out}, we adopt state-of-the-art neural network verifier `NeuralSAT` [15], [23] as a black-box tool. `NeuralSAT` is a top-performing verifier from the latest VNN-COMPs [16] that employs GPU-based linear relaxations and branch-and-bound techniques to efficiently verify DNN properties.

We convert our verification problem into the standard VNN-COMPs format, which consists of three components: (1) the DeepCog DNN model in ONNX format, (2) input properties ϕ_{in} specified in VNNLIB format that encode the adversarial traffic injection bounds $[\delta^s(t)+\eta_L, \delta^s(t)+\eta_U]$, and (3) output properties ϕ_{out} in VNNLIB format that encode the loss function threshold constraint $\mathcal{L}(c^s(t), d^s(t)) \leq \zeta \cdot \mathcal{L}(c_0^s(t), d^s(t))$. The conversion of DeepCog's piecewise linear loss function into linear and ReLU layers, as described in the previous subsection, ensures compatibility with `NeuralSAT`'s verification engine. Since 3D convolution is not supported, we reimplement it as two consecutive 2D convolutional layers. This does not reduce the forecast accuracy, while allowing the exported model to be compatible with `NeuralSAT`.

For each verification instance, `NeuralSAT` returns one of three possible outcomes: `SAT`, `UNSAT`, or `timeout`. A `SAT` result indicates that the verification problem is satisfiable, meaning there exists at least one adversarial input within the specified bounds $[\eta_L, \eta_U]$ that causes the DeepCog model to violate the output property ϕ_{out}, *e.g.*, the loss function exceeds the threshold ζ. This result demonstrates a concrete vulnerability: adversarial traffic injection can degrade capacity forecasting performance beyond the acceptable threshold. An `UNSAT` result indicates that the verification problem is unsatisfiable, meaning no adversarial input within the specified bounds can violate the output property. This result provides a formal mathematical guarantee that the DeepCog model is robust against all possible adversarial traffic injection scenarios within the given input constraints. A `timeout` result indicates that the verification instance exceeds the computational time limit, meaning the verifier cannot determine within the allocated resources whether the property holds.

The soundness of `NeuralSAT`'s verification procedure ensures that when it returns `UNSAT`, the result is mathematically sound, meaning no adversary can successfully attack the DeepCog model under the specified input and output conditions. This soundness guarantee is fundamental to our verification framework: if `NeuralSAT` certifies that a DeepCog model is robust (returns `UNSAT`), then the claim is guaranteed to be true, providing network operators with confidence when deploying the verified model for resource provisioning in production environments. This formal assurance goes beyond empirical testing methods, which can only demonstrate the existence of vulnerabilities through specific adversarial examples but cannot prove their absence across the entire continuous input space.

IV. Evaluation Results

A. Evaluation Setups

1) Dataset: We evaluate our verification framework using the Telecom Italia Milan dataset, which is a publicly available mobile traffic dataset widely used in the literature [18]. The dataset contains mobile traffic data collected in 2014 from Milan, Italy, covering 1,728 base stations and aggregated into a grid of approximately 10,000 square cells using Voronoi tessellation techniques. The data includes SMS, voice calls, and Internet activities recorded at 10-minute granularity. Following standard practice in mobile traffic prediction research, we use Internet activities as a proxy for mobile traffic volume. This dataset provides a spatial-temporal pattern of internet usage, making it well-suited for training and evaluating DeepCog-based traffic forecasting models.

2) DNN Hyperparameters, input/output properties: A DeepCog model is trained using an Adam optimizer with a learning rate of $3e^{-4}$ over 50 epochs, employing Rectified Linear Unit (ReLU) as the activation function for all layers. We use a standard 80:20 training-testing split, where each sample represents a traffic-demand snapshot within a 10-minute time interval.

For the prediction methodology, we select a representative area $A_{\text{Milan}} \in G_{\text{Milan}}$ of 5×5 cells similar to [9]. Within this area, we train small models on 5×5 grids, where each model forecasts the capacity/traffic of the central cell only.

The DeepCog predictor employs a parameter α that represents the amount of overprovisioned capacity units that determine a penalty equivalent to one SLA violation. A larger α implies higher SLA violation fees for the operator, thus influencing the balance between overprovisioning and SLA violations. For the Milan dataset, we set $\alpha = 2$ as suggested in [18] to prioritize avoiding SLA violations while allowing minimal overprovisioning.

3) Evaluation Metrics: For each verification instance, `NeuralSAT` returns one of three possible outcomes: `SAT`, `UNSAT`, or `timeout`. The primary performance metric is the number of instances that return `SAT` (vulnerability found), `UNSAT` (formal robustness guarantee), or `timeout` (unknown within the time limit). These are commonly used to compare the performance of different verification methods [16], [23] and provides a standardized evaluation framework for assessing the robustness of DNN-based traffic forecasting models.

B. Robustness Analysis under Adversarial Traffic Injection

Tab. I and Tab. II present the verification results for adversarial traffic injection scenarios, where input traffic volumes are perturbed by varying percentages. Each cell reports the proportion of instances yielding `SAT`, `UNSAT`, or `TIMEOUT` for a given tolerance parameter ζ and injection level η. A `SAT` outcome indicates that the verifier identified at least one adversarial input causing the model to violate the output property (i.e., forecast error exceeding ζ). Conversely, `UNSAT`

	Percentage of injected traffic				
	+1%	+5%	+10%	+15%	+20%
$\zeta = 1\%$	0.0/1.0/0.0	0.0/1.0/0.0	0.0/1.0/0.0	0.0/1.0/0.0	0.0/1.0/0.0
$\zeta = 5\%$	0.0/1.0/0.0	0.0/1.0/0.0	0.0/1.0/0.0	0.0/1.0/0.0	0.0/1.0/0.0
$\zeta = 10\%$	0.0/1.0/0.0	0.0/1.0/0.0	0.0/1.0/0.0	0.0/1.0/0.0	0.0/1.0/0.0

TABLE I: Fraction of UNSAT, SAT, TIMEOUT for underestimation of user demand

	Percentage of injected traffic				
	+1%	+5%	+10%	+15%	+20%
$\zeta = 1\%$	1.0/0.0/0.0	1.0/0.0/0.0	1.0/0.0/0.0	0.0/1.0/0.0	0.0/1.0/0.0
$\zeta = 5\%$	1.0/0.0/0.0	1.0/0.0/0.0	1.0/0.0/0.0	0.0/0.0/1.0	0.0/1.0/0.0
$\zeta = 10\%$	1.0/0.0/0.0	1.0/0.0/0.0	1.0/0.0/0.0	1.0/0.0/0.0	0.0/0.0/1.0

TABLE II: Fraction of UNSAT, SAT, TIMEOUT for overestimation of user demand

certifies robustness under all perturbations within the specified bounds, while `TIMEOUT` denotes inconclusive verification within the allotted time.

- *Underestimation scenario (Tab. I):* Across all injection levels and underestimation tolerances, the verifier consistently returns `SAT`. Although we set α to guide DeepCog to reduce underestimation, the model was still vulnerable on all properties. This demonstrates that adversary can easily induce capacity underestimation beyond acceptable thresholds, exposing the model to potential denial-of-service or quality-of-service degradation attacks.
- *Overestimation scenario (Tab. II):* In contrast, the model exhibits strong robustness against overestimation. For small injection levels (+1%, +5%, +10%), `NeuralSAT` frequently returns `UNSAT`, confirming the model's ability to avoid excessive overprovisioning and resource waste. However, this property is less critical in practice, as service providers typically tolerate minor overprovisioning. At higher injection levels (+15% and +20%), the results shift toward `SAT` or `TIMEOUT`, indicating that extreme perturbations can compromise robustness and require significant overprovisioning.

Overall, these findings reveal an asymmetric robustness profile: the model is provably resilient to overestimation under moderate perturbations but remains highly vulnerable to underestimation attacks.

V. Conclusion

We presented a formal verification framework for deep learning–based mobile traffic prediction, enabling provable robustness against adversarial traffic injection. By encoding input perturbations and defining output properties that enforce resource allocation bounds, our approach leverages NeuralSAT to provide sound robustness guarantees. Evaluation on the Telecom Italia Milan dataset reveals an asymmetric robustness profile: the model is resilient to overestimation under moderate perturbations but highly vulnerable to underestimation attacks. Future work will extend this study with more comprehensive evaluations across a wider range of datasets and models to generalize our findings and improve robustness in diverse network scenarios.

References

[1] O. Aouedi, V. A. Le, K. Piamrat, and Y. Ji, "Deep learning on network traffic prediction: Recent advances, analysis, and future directions," *ACM Computing Surveys*, vol. 57, no. 6, pp. 1–37, 2025.

[2] Y. Ji, H. H. Tran, P. Le Nguyen, J. C. Lui, et al., "Achieving multi-time-step segment routing via traffic prediction and compressive sensing techniques," *IEEE Transactions on Network and Service Management*, vol. 21, no. 2, pp. 1534–1549, 2023.

[3] D. L. Nguyen, P. Le Nguyen, Y. Ji, et al., "Traffic engineering in large-scale networks via multi-agent deep reinforcement learning with joint-training," in *2024 33rd International Conference on Computer Communications and Networks (ICCCN)*, IEEE, 2024, pp. 1–9.

[4] T. T. Le, P. Le Nguyen, H. T. T. Binh, R. Akerkar, Y. Ji, et al., "Gcrint: Network traffic imputation using graph convolutional recurrent neural network," in *ICC 2021-IEEE International Conference on Communications*, IEEE, 2021, pp. 1–6.

[5] T. T. Le, P. Le Nguyen, H. T. T. Binh, Y. Ji, et al., "Multi-time-step segment routing based traffic engineering leveraging traffic prediction," in *2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, IEEE, 2021, pp. 125–133.

[6] P. Le Nguyen, Y. Ji, et al., "Deep convolutional lstm network-based traffic matrix prediction with partial information," in *2019 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, IEEE, 2019, pp. 261–269.

[7] H. Zare Moayedi and M. Masnadi-Shirazi, "Arima model for network traffic prediction and anomaly detection," in *2008 International Symposium on Information Technology*, vol. 4, 2008, pp. 1–6. DOI: 10.1109/ITSIM.2008.4631947.

[8] T. Le, H. Duong, Y. Ji, T. Nguyen, and J. Lui, "Fggm: Formal grey-box gradient method for attacking drl-based mu-mimo scheduler," *arXiv preprint arXiv:2510.26075*, 2025.

[9] S. M. Gholian, C. Fiandrino, N. Vallina-Rodriguez, M. Fiore, and J. Widmer, "Deexp: Revealing model vulnerabilities for spatio-temporal mobile traffic forecasting with explainable ai," *IEEE Transactions on Mobile Computing*, 2025.

[10] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," in *International conference on machine learning*, PMLR, 2018, pp. 274–283.

[11] G. Singh, T. Gehr, M. Püschel, and M. Vechev, "An abstract domain for certifying neural networks," *Proceedings of the ACM on Programming Languages*, vol. 3, no. POPL, pp. 1–30, 2019.

[12] J. Cohen, E. Rosenfeld, and Z. Kolter, "Certified adversarial robustness via randomized smoothing," in *International conference on machine learning*, PMLR, 2019, pp. 1310–1320.

[13] K. Xu et al., "Automatic perturbation analysis for scalable certified robustness and beyond," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1129–1141, 2020.

[14] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. J. Kochenderfer, "Reluplex: A calculus for reasoning about deep neural networks," *Formal Methods in System Design*, vol. 60, no. 1, pp. 87–116, 2022.

[15] H. Duong, T. Nguyen, and M. B. Dwyer, "NeuralSAT: A High-Performance Verification Tool for Deep Neural Networks," in *International Conference on Computer Aided Verification*, 2025, to appear.

[16] C. Brix, S. Bak, T. T. Johnson, and H. Wu, "The Fifth International Verification of Neural Networks Competition (VNN-COMP 2024): Summary and Results," *arXiv preprint arXiv:2412.19985*, 2024.

[17] J. Kim, H.-S. Lim, and K. Choi, "Certified Robustness of Antenna Selecting Neural Networks for Massive MIMO Wireless Communications," *IEEE Access*, 2025.

[18] D. Bega, M. Gramaglia, M. Fiore, A. Banchs, and X. Costa-Perez, "Deepcog: Cognitive network management in sliced 5g networks with deep learning," in *IEEE INFOCOM 2019-IEEE conference on computer communications*, IEEE, 2019, pp. 280–288.

[19] M. Antonakakis et al., "Understanding the mirai botnet," in *26th USENIX security symposium (USENIX Security 17)*, 2017, pp. 1093–1110.

[20] J. Gamba, M. Rashed, A. Razaghpanah, J. Tapiador, and N. Vallina-Rodriguez, "An analysis of pre-installed android software," in *2020 IEEE symposium on security and privacy (SP)*, IEEE, 2020, pp. 1039–1055.

[21] E. Blázquez et al., "Trouble over-the-air: An analysis of fota apps in the android ecosystem," in *2021 IEEE Symposium on Security and Privacy (SP)*, IEEE, 2021, pp. 1606–1622.

[22] S. Farooqi, Á. Feal, T. Lauinger, D. McCoy, Z. Shafiq, and N. Vallina-Rodriguez, "Understanding incentivized mobile app installs on google play store," in *Proceedings of the ACM internet measurement conference*, 2020, pp. 696–709.

[23] H. Duong, D. Xu, T. Nguyen, and M. B. Dwyer, "Harnessing neuron stability to improve dnn verification," *Proc. ACM Softw. Eng.*, vol. 1, no. FSE, 2024. DOI: 10.1145/3643765.