Enabling Self-Driving Networks with Machine Learning

Arthur Selle Jacobs[1], **Ronaldo Alves Ferreira**[2], **Lisandro Zambenedetti Granville**[1]

[1] Institute of Informatics – Federal University of Rio Grande do Sul (UFRGS)

[2]College of Computing – Federal University of Mato Grosso do Sul (UFMS)

Abstract. *This work aims to enable self-driving networks by addressing the lack of trust that network operators have in Machine Learning (ML) models for networking problems. To achieve this, we propose a natural-language conversational interface (*LUMI*) and a new ML pipeline that uses techniques from the emerging field of eXplainable Artificial Intelligence (XAI) to scrutinize models. We also propose a new XAI method called* TRUSTEE *to extract explanations from any given black-box ML model in the form of decision trees of manageable sizes. Our results indicate that ML models used in networking problems need to be put under proper scrutiny and corrected to fulfill their tasks properly.*

1. Introduction

As networks grow larger and more complex, they become more susceptible to human errors. To reduce these errors, both industry and academia are automating network management and control tasks [Apostolopoulos 2020, Mirsky et al. 2018], with the ultimate goal of creating a network that is autonomous and capable of making decisions without human interactions. Autonomous networking has been a longstanding goal, but it has not been fully realized due to technological limitations. Recent advances in Artificial Intelligence (AI) and Machine Learning (ML) have brought new hope to this idea, which has been rebranded as *self-driving networks* [Feamster and Rexford 2018]. Although the definition of a self-driving network varies among companies and researchers [Feamster and Rexford 2018][Huawei 2019] , there are some common design elements in their definitions. To provide clarity, we define a *self-driving network as an autonomous network capable of acting according to high-level intents from an operator and automatically adapting to changes in traffic and user behavior*. To achieve this vision, a self-driving network must be able to *(i)* understand high-level intents, *(ii)* monitor itself based on those intents, *(iii)* predict and identify changing patterns from monitored data, and *(iv)* adapt itself without operator intervention. Figure 1 presents a high-level design of a self-driving network summarizing features and requirements found in the literature.

This self-driving network design is composed of two management loops. On the left side of Figure 1, we see the first management loop (1, 2, 3, and 4), which starts with an operator (1) specifying high-level intents, in natural language, that dictate how the network should behave—*e.g.*, defining goals related to quality of service, security, and performance–without worrying about the low-level details that are necessary to program the network to achieve these goals (*a.k.a.* Intent-Based Networking — IBN). Using natural language to describe network intents requires the network to employ state-of-the-art ML techniques from Natural Language Processing (NLP), which are always prone to generate errors and misclassifications. Hence, after extracting relevant information from input intents, a self-driving network would then (2) validate the extracted data with the

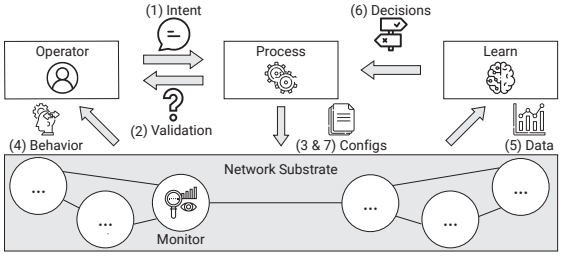


Figure 1. Self-driving network design.

operator before (3) compiling them into actual configurations and deploying them to the network substrate. To close the first management loop, the network would then monitor itself according to the described intents and collect the traffic behavior to (4) present it for operators to verify if the implemented behavior corresponds to the initial goals.

In Figure 1, the second management loop (5, 6, and 7) begins after intents are deployed into the network, where devices are instrumented with monitoring capabilities to collect usage and traffic data. Such data is then (5) analyzed and processed to (6) produce (autonomous) decisions using trained ML models, which would ideally adapt and re-train themselves constantly as new data is collected. Decisions made by such ML models would then be (7) processed and compiled into configurations to fine-tune the network behavior (akin to 3), closing the second management loop. For instance, decisions may include the identification of attack traffic that needs to be blocked or mitigated. Note that, despite relying on error-prone ML techniques, it would be impossible to include human validation on this second management loop given the time frame and frequency such decisions have to occur to keep up with incoming traffic.

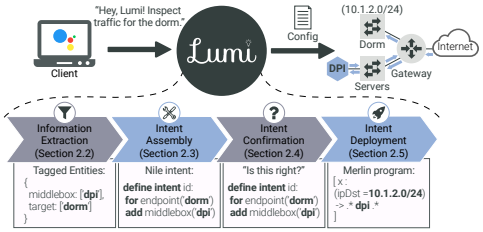
As both management loops in the self-driving network presented in Figure 1 rely heavily on ML models to make decisions and classifications that impact the network, one particular issue becomes prominent with this design: *trust*. Applying ML to solve networking management tasks, such as the ones described above, has been a popular trend among researchers recently [Boutaba et al. 2018]. However, industry operators have been reluctant to take advantage of such solutions, mainly because of the black-box nature of ML models that produce decisions without any explanation or reason. In production networks, the high stakes make it impossible to trust an ML model that can take system-breaking actions automatically. This poses a significant challenge that must be addressed to achieve a self-driving network design, which is crucial to the scope of this thesis.

2. Problem Statement and Contributions

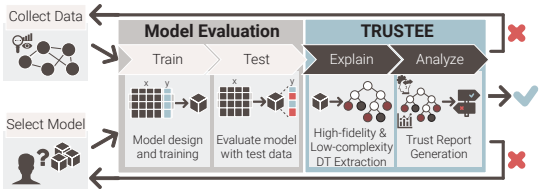
This thesis addresses the challenges of creating trustworthy self-driving networks with ML. We first investigate and evaluate the accuracy and credibility of classifications made by ML models used to process high-level intents. Then, we analyze and assess the accuracy and credibility of decisions made by ML models used to self-configure the network according to monitored data. Lastly, we investigate whether there is a viable method to improve the trust of operators in the decisions made by ML models in both management loops. In this thesis, we make the following main contributions.

- We develop an end-to-end IBN management system with a conversational interface that allows operators to use natural language to define desired intents for the network, called LUMI (Fig. 2a).

- We survey the existing literature on the use of ML techniques for network security and scrutinize several use cases to analyze the credibility of decisions made by highly-accurate ML models that enable a self-driving network.
- We introduce a novel model-agnostic XAI method to produce explanations from any given black-box ML model in the form of decision trees, called TRUSTEE (Fig. 2b), which domain experts can use to spot issues in the decision-making process of the black-box model.



(a) The four modules of LUMI.



(b) TRUSTEE overview.

Figure 2. Thesis contributions overview.

3. Machine Learning for IBN

Modern enterprise networks face significant challenges when deploying policies as operators need to break down high-level policies into low-level tasks and deploy them across the network. Intent-based networking (IBN) has been proposed to address this issue, allowing operators to specify high-level policies (*e.g.,* defining goals for quality of service, security, and performance) without worrying about how the network is programmed to achieve the desired goals. [Clemm et al. 2019]. IBN should enable operators to simply state the desired behavior, with the network breaking down the intent into configurations and deploying them correctly. Supporting IBN is a core principle of a self-driving network, as it minimizes human intervention and reduces errors caused by human mistakes.

One of the main reasons that IBN has not yet fulfilled its promise of fast, automated, and reliable policy deployment is its inability to use network policies documented in natural language as input to intent-based management systems. While some research efforts have used natural language to interact with the network [Liu 2021, Birkner et al. 2018], they lack support for IBN or other critical features like operator confirmation and feedback. However, expressing intents in natural language offers benefits such as allowing operators to express the same intent using different phrasings while avoiding the pitfalls of traditional policy deployment approaches (*e.g.,* learning new network programming languages or introducing human errors while manually breaking down policies into configuration commands). On the other hand, generating configurations from natural language input can be challenging due to the need for unambiguous and accurate capturing of operator intent, which is not easily achieved through ML models.

We contribute to the ongoing IBN efforts by designing and implementing LUMI, a system that enables an operator "to talk to the network," focusing on campus networks as a use case. LUMI takes as input an operator's intent expressed in natural language, correctly translates these natural language utterances into configuration commands, and deploys the latter in the network to carry out the operator's intent. We designed LUMI in a modular fashion, with the different modules performing, in order: information extraction,

intent assembly, intent confirmation, and intent compilation and deployment. Our modular design allows for easy plug-and-play, where existing modules can be replaced with alternative solutions, or new modules can be included. As a result, LUMI's architecture is extensible and evolvable and can easily accommodate further improvements.

3.1. LUMI in a Nutshell

Figure 2a illustrates the high-level goal of LUMI with the intent example *"Hey, Lumi! Inspect traffic for the dorm"* and shows the breakdown of the workflow by which LUMI accomplishes the stated objective. Below, we provide a brief overview of the four key components that define this workflow (*i.e.,* the LUMI pipeline) and the contributions we make in addressing the various challenges described above.

First, we use ML for the Information Extraction module to extract and label entities from operator utterances through a chatbot-like conversational interface. We use existing ML algorithms for Named Entity Recognition (NER) to extract and label entities from the operator's natural language utterances and augment them for LUMI to learn from operator-provided feedback. Second, the Intent Assembly module composes a network intent using the previously extracted entities. To that end, we developed the *Network Intent Language* (*Nile*) to serve as an abstraction layer between natural language intents and network configuration commands for LUMI. *Nile* resembles natural language, accounts for critical network management features, such as rate-limiting or usage quotas, and eliminates the need for operators to learn new policy languages. Third, the Intent Confirmation module presents the syntactically-correct *Nile* intent from the Intent Assembly module to the operator and asks for feedback. If the feedback is negative, the system iterates with the operator until confirmation, continuously learning from the operator's feedback to improve labeling accuracy. Once confirmed, the *Nile* intent is forwarded to the Intent Deployment module, which compiles it into Merlin configuration commands and deploys them in the network.

We evaluate LUMI's accuracy in information extraction, its ability to learn from operator feedback, and its compilation and deployment times in a campus topology. Our datasets consist of synthesized and real-world intents from 50 different campus networks in the US. We find that LUMI can extract entities accurately, learn from feedback, and compile and deploy intents in less than a second. We also report findings from a small user study with 26 subjects that solicited feedback on the perceived value of using natural language for network management with LUMI and the intent confirmation stage. We developed LUMI using a combination of tools and libraries such as Google Dialogflow and *Scikit-learn* library. The full implementation and datasets used in our evaluation are publicly available. Our evaluation and user study demonstrate that LUMI is a promising step towards achieving fast, automated, and reliable policy deployment in IBN. By enabling operators to express their intents in natural language, LUMI simplifies their job and saves them time.

3.2. Selected Results

Evaluating systems like LUMI is challenging due to a lack of publicly available datasets and difficulties in generating synthetic datasets that reflect real-world Natural Language Intents (*e.g.,* operators we contacted gave proprietary reasons for not sharing their data). To address this problem, we created two hand-annotated datasets for information extraction: the *alpha* dataset with 150 synthetic examples and the *campi* dataset with real-world

intents obtained by crawling 50 US university websites. We manually tagged the entities in each of the 200 intents to train and validate our information extraction model. We tested our NER model using both datasets, alone and combined, with a 75%-25% training-testing random split. The results in Table 1 demonstrate the high accuracy of LUMI's information extraction module for the *alpha* dataset, *campi* dataset, and their combination. The excellent performance for the *alpha* dataset was expected due to the way we created the training examples. However, even with the unstructured nature and smaller number of examples in the *campi* dataset, LUMI's performance remained close to 0.99. This success can be attributed to recent advances in machine learning for natural language processing and the small yet expressive set of LUMI-defined entities.

Table 1. Information extraction evaluation using the *alpha* and *campi* dataset.

Dataset	# of Entries	Precision	Recall	F1
alpha	150	0.996	**0.987**	**0.991**
campi	50	**1**	0.979	0.989
alpha + *campi*	200	0.992	0.969	0.980

Since human interaction is expected to be minimal to express network intents in a self-driving network, humans can verify and correct the classifications made by ML models that extract relevant information from natural language intents. This feedback loop enables operators to trust the decisions made by these models.

4. Machine Learning for Network Security

Recent research has demonstrated the superiority of AI/ML models over simpler rule-based heuristics in identifying complex network traffic patterns for a wide range of network problems [Boutaba et al. 2018]. However, reluctance among operators to adopt these models in production settings persists due to their black-box nature [Arp et al. 2022]. More concretely, the inability to explain how and why these models make their decisions renders them a hard sell compared to existing simpler but typically less effective rule-based approaches. This tension is not unique to networking but applies to any learning model that can have significant societal implications. Efforts to solve this issue have led to the emergence of areas such as "interpretable ML," "explainable AI (XAI)," and "trustworthy AI." Ensuring the practical use of these efforts in specific AI/ML domains, like network security, requires addressing several fundamental research problems and qualifying notions such as model interpretability or trust [Lipton 2018].

Applying AI/ML to network security poses challenges due to the mismatch between the black-box nature of some models and the need for practitioners to understand them. While black-box learning models are inherently incapable of providing insights into their "inner workings" or underlying decision-making process, operators and security experts are particularly keen on gaining a basic understanding of how these proposed models work in practice so they can be trusted in real-world production settings. For this thesis, we equate *"an end user having trust in an AI/ML model"* with *"an end user being comfortable with relinquishing control to the model"* [Lipton 2018]. We focus on research challenges related to quantitatively deciding when an end user is comfortable with relinquishing control to a given AI/ML model, including identifying if the model suffers from the problem of underspecification [D'Amour et al. 2020].

The problem of underspecification in AI/ML refers to whether the success of a trained model is due to its innate ability to encode some essential structure of the underlying system or data or is the result of inductive biases that the trained model encodes. In practice, inductive biases typically manifest themselves in an inherent inability for out-of-distribution (o.o.d.) generalizations (*i.e.,* test data distribution is unknown and different from the training data distribution) which, in turn, often reveals itself in the form of specious learning strategies (*e.g.,* shortcut learning or spurious correlations). Such inductive biases imply that their presence in trained AI/ML models prevents these models from being credible or trustworthy. Thus, for establishing the specific type of trust in an ML model considered in this section, it is critical to identify these inductive biases, and this section takes the first step toward achieving this ambitious goal.

4.1. TRUSTEE

To detect underspecification in learning models for network security problems, we develop TRUSTEE (TRUSt-oriented decision TreE Extraction). TRUSTEE augments the traditional ML pipeline (Figure 2b) by taking a black-box model and training data as input and outputting a high-quality decision tree explanation. TRUSTEE focuses on balancing model fidelity and complexity while ensuring decision rules are intelligible and in agreement with domain knowledge. Here, complexity refers to the decision tree's size and aspects of the tree's branches. In particular, when viewing the tree's branches as decision rules, we are concerned with their explicitness/intelligibility and coverage; that is, we require these rules to be readily recognizable by domain experts, be largely in agreement with the experts' domain knowledge, and describe how the given black-box model makes a significant number of its decisions. TRUSTEE also outputs a trust report associated with the decision tree explanation, which operators can use to determine whether there is evidence that the given black-box model suffers from underspecification. If such evidence is found, the information provided in the trust report can be used to identify components of the traditional ML pipeline (*e.g.,* training data, model selection) that need to be modified to improve upon an ML model that TRUSTEE has found to be untrustworthy.

Our work contributes to the growing ML literature on model explainability/interpretability and differs from existing approaches in significant ways. First, replacing black-box models with "white-box" models like decision trees is generally impractical for complex networking learning problems. Second, existing methods that provide *local interpretability* (*e.g.,* SHAP [Shapley 2016]) are not suitable for examining instances of the underspecification problem as they can explain the decisions of a trained AI/ML model in a local region near a particular data point. Finally, methods for *global interpretability* [Bastani et al. 2018] are either limited to specific learning models (*e.g.,* reinforcement learning) or suffer from poor fidelity. These approaches are insufficient for providing the level of model explainability needed for network operators to decide if they are comfortable with relinquishing control to a black-box model.

4.2. Use Cases Summary

We apply TRUSTEE to scrutinize a number of recently published black-box models developed for network security-related problems that are accompanied by publicly available artifacts required for assessing whether the models are credible. All datasets, models, and results presented in this section are publicly available (see thesis for the repository).

Table 2. Case Studies.

Problem	Dataset(s)	Model(s)	Trustee Fidelity	Type of inferred inductive bias
Detect VPN traffic	ISCX VPN-nonVPN dataset	1-D CNN	1.00	Shortcut learning
Detect Heartbleed traffic	CIC-IDS-2017	RF Classifier	0.99	Out-of-distribution samples
Detect Malicious traffic (IDS)	CIC-IDS-2017, Campus dataset	nPrintML	0.99	Spurious correlations
Anomaly Detection	Mirai dataset	Kitsune	0.99	Out-of-distribution samples
OS Fingerprinting	CIC-IDS-2017	nPrintML	0.99	Potential out-of-distribution samples
IoT Device Fingerprinting	UNSW-IoT	Iisy	0.99	Likely shortcut learning
Adaptive Bit-rate	HSDPA Norway	Pensieve	0.99	Potential out-of-distribution samples

Table 2 summarizes our use cases, which demonstrate the limitations of ML models for networking problems. The first use case illustrates how an apparently high-performant neural network learns shortcuts to distinguish between two types of traffic (VPN vs. Non-VPN). The second use case analyzes a black-box model (*i.e.,* random forest) trained with a popular synthetic dataset CIC-IDS-2017 [Sharafaldin et al. 2018] and shows that the developed model is vulnerable to o.o.d. samples. This use case cautions against an over-reliance on synthetic datasets that often include measurement artifacts that commonly-considered black-box models exploit to achieve high accuracy. The third use case warns against the indiscriminate use of high-dimensional feature spaces built from bit vectors instead of carefully engineered and semantically meaningful features [Holland et al. 2021]. The fourth use case corroborates previous criticism [Arp et al. 2022] of a complex neural network ensemble [Mirsky et al. 2018] to perform traffic anomaly detection (*e.g.,* Mirai attack) and shows that the model is also vulnerable to o.o.d. samples. We used TRUSTEE to analyze other ML-based models for networking and network security problems in the literature, which we discuss in the thesis.

The use cases analyzed show that ML models used for networking problems fail to perform under minimal adverse conditions. This finding indicates that operators are correct not to trust ML and cannot trust ML models to configure the network based solely on monitored data. However, TRUSTEE exposes decisions made by black-box ML classifiers, increasing trust in those decisions. With this knowledge, operators can choose to relinquish control to the ML model if they agree with its decision.

5. Scientific Production

The following list shows the main papers and articles related to this thesis.

- **A. S. Jacobs**, R. L. dos Santos, R. J. Pfitscher, M. F. Franco, E. J. Scheid, L. Z. Granville. Affinity measurement for NFV-enabled networks: A criteria-based approach. In Symposium on Integrated Network and Service Management (IM), Lisbon, Portugal, 2017, pp. 125-133.
- **A. S. Jacobs**, R. J. Pfitscher, R. L. dos Santos, M. F. Franco, E. J. Scheid, L. Z. Granville. Artificial neural network model to predict affinity for virtual network functions. In Network Operations and Management Symposium (NOMS). Taipei, Taiwan, 2018, pp 1-9.
- **A. S. Jacobs**, R. J. Pfitscher, R. A. Ferreira, L. Z. Granville. Refining network intents for self-driving networks. In ACM SIGCOMM Afternoon Workshop on Self-Driving Networks (SelfDN). Budapest, Hungary, 2018, pp 15-21. **Best paper award**.
- **A. S. Jacobs**, R. J. Pfitscher, R. A. Ferreira, L. Z. Granville. Refining network intents for self-driving networks. In ACM SIGCOMM CCR, Volume 48, Issue 5, 2018, pp 55–63.
- **A. S. Jacobs**, R. J. Pfitscher, R. H. Ribeiro, R. A. Ferreira, L. Z. Granville, W. Willinger, S. Rao. Hey, Lumi! Using Natural Language for Intent-Based Network Management. In USENIX Annual Technical Conference (ATC). Virtual Conference, 2021, pp 625-639.

- **A. S. Jacobs**, R. Beltiukov, W. Willinger, R. A. Ferreira, A. Gupta, L. Z. Granville. AI-ML and Network Security: The Emperor has no Clothes. In ACM Conference on Computer and Communications Security (CCS). Los Angeles, CA, 2022, pp 1537–1551. **CCS'22 Best Paper Honorable Mention. 2023 IETF/IRTF Applied Networking Research Prize (ANRP).**
- **A. S. Jacobs**, R. J. Pfitscher, R. H. Ribeiro, R. A. Ferreira, L. Z. Granville, W. Willinger, S. Rao. On the Use of Natural Language for Intent-Based Networking. In IEEE Communications Surveys & Tutorials. (Under review)

Acknowledgements

This work was supported in part by the Brazilian CNPq proc. 142089/2018-4 (Arthur's fellowship), and by FAPESP procs. 2015/24494-8 and 2020/05152-7.

References

Apostolopoulos (2020). Improving Networks with Artificial Intelligence. https://bit.ly/3IEYT5e.

Arp et al. (2022). Dos and Dont's of Machine Learning in Computer Security. In *USENIX Security 22*.

Bastani et al. (2018). Verifiable Reinforcement Learning via Policy Extraction. In *NIPS'18*.

Birkner et al. (2018). Net2Text: Query-Guided Summarization of Network Forwarding Behaviors. In *USENIX NSDI '18*.

Boutaba et al. (2018). A Comprehensive Survey on Machine Learning for Networking: Evolution, Applications and Research Opportunities. *JISA*, 9(1).

Clemm, A., Ciavaglia, L., Granville, L. Z., and Tantsura, J. (2019). Intent-Based Networking - Concepts and Definitions. Internet-draft, Internet Engineering Task Force.

D'Amour et al. (2020). Underspecification Presents Challenges for Credibility in Modern Machine Learning. In *arXiv 2011.03395*.

Feamster and Rexford (2018). Why (and How) Networks Should Run Themselves. In *ACM ANRW '18*.

Holland et al. (2021). New Directions in Automated Traffic Analysis. In *ACM CCS '21*.

Huawei (2019). Huawei Core Network Autonomous Driving Network White Paper.

Lipton (2018). The Mythos of Model Interpretability: In Machine Learning, the Concept of Interpretability is Both Important and Slippery. *Queue*, 16(3).

Liu (2021). The Practice of Network Verification in Alibaba's Global WAN. https://bit.ly/3XNjfiw.

Mirsky et al. (2018). Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection. In *NDSS'18*.

Shapley (2016). *A Value for n-Person Games*. Princeton U. Press.

Sharafaldin et al. (2018). Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. In *ICISSP*.