Enabling Self-Driving Networks with Machine Learning

Arthur Selle Jacobs*, Ronaldo Alves Ferreira†, Lisandro Zambenetti Granville*,
*Federal University of Rio Grande do Sul, Brazil. †Federal University of Mato Grosso do Sul, Brazil.

***Abstract*—This work aims to enable self-driving networks by tackling the lack of trust that network operators have in Machine Learning (ML) models. We assess and scrutinize the decision-making process of ML-based classifiers used to compose a self-driving network. First, we investigate and evaluate the accuracy and credibility of classifications made by ML models used to process high-level management intents. We propose a novel conversational interface (LUMI) that allows operators to use natural language to describe how the network should behave. Second, we analyze and assess the accuracy and credibility of existing ML models' for network security and performance. We also uncover the need to reinvent how researchers apply ML to networking problems, so we propose a new ML pipeline that introduces steps to scrutinize models using techniques from the emerging field of eXplainable Artificial Intelligence (XAI). Finally, we investigate whether there is a viable method to improve the trust of operators in the decisions made by ML models that enable self-driving networks. Our investigation led us to propose a new XAI method to extract explanations from any given black-box ML model in the form of decision trees while maintaining a manageable size, which we called TRUSTEE. Our results show that ML models widely applied to solve networking problems have not been put under proper scrutiny and can easily break when put under real-world traffic. Such models, therefore, need to be corrected to fulfill their given tasks properly.**

***Index Terms*—Self-Driving Networks, Machine Learning, Explainability, Intent-Based Networking**

I. INTRODUCTION

As modern networks grow in size and complexity, they also become increasingly prone to human errors [1]. This trend has driven both industry and academia to automate management and control tasks, aiming to reduce human interaction with the network and human-made mistakes [2][3]. Ideally, researchers envision a network design that is not only automatic (*i.e.,* dependent of human instructions) but autonomous (*i.e.,* capable of making its own decisions). Autonomous networking has been a goal sought for years, with many different concepts, designs, and implementations, but it was never been fully realized, mainly due to technological limitations [1]. Recent advances in Artificial Intelligence (AI) and Machine Learning (ML) introduced a breath of fresh air into this concept, reemerging as the re-branded concept of *self-driving networks*, in view of its autonomous car counterparts.

While the concept of a self-driving network has no standardized definition, with each company and researcher having its own vision and architecture [1][4][5], a few design elements are common in all instances. As no standard definition has been adopted as standard by the community, we rely on the following definition of a self-driving network, summarizing the main aspects found in the literature: *a self-driving network is 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 that vision, a network would need to fulfill four major requirements: *(i)* understand high-level intents that dictate its behavior, *(ii)* monitor itself based on input intents, *(iii)* predict and identify changing patterns from monitored data, and *(iv)* adapt itself to new behaviors without the intervention of an operator. In Figure 1, we present a high-level design of a self-driving network that summarizes the features and requirements found in the literature.

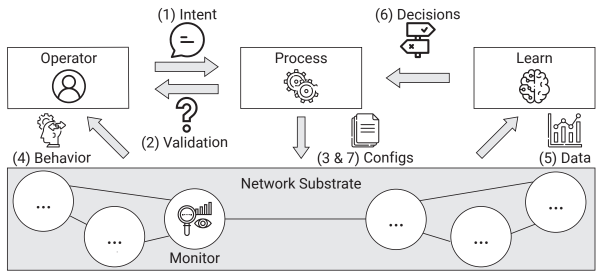


Fig. 1: Self-driving network design.

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) [6]. Enabling operators to freely use natural language to describe network intents requires the network to employ state-of-the-art ML techniques from Natural Language Processing (NLP) [7], 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 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 behavior of traffic to (4) present it for operators to verify if the implemented behavior corresponds to the initial goals. On the right side of Figure 1, the second management loop (5, 6, and

7) begins after intents are deployed into the network substrate, 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 from ML models may include the identification of attack traffic that needs to be blocked or mitigated or even resource usage optimizations based on traffic load. Notice that, despite also relying heavily on error-prone ML techniques, given the time frame and frequency such decisions and configuration deployment would occur to keep up with incoming traffic, it would be impossible to include human validation on this second management loop, in contrast with the first one. In addition, another key aspect to keep in mind is that self-configurations made based on decisions from ML models in the second management loop can undermine and contradict the configurations made through network intents in the first management loop. The possibility of such conflicts arising, coupled with the inability to include human-in-the-loop validations, raises the stakes for any decision made by ML models in the second management loop.

As both management loops in the self-driving network presented in Figure 1 rely heavily on ML models to make decisions and classifications that directly 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 [8]. However, despite the topic receiving much attention, 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. Given the high-stakes nature of production networks, it becomes impossible to trust an ML model that may take system-breaking actions automatically, and most important to the scope of this thesis, a prohibitive challenge that must be addressed if a self-driving network design is ever to be achieved.

II. Problem Statement and Research Questions

The present thesis [9] addresses the issue of enabling self-driving networks with ML, tackling the problem of the inherent lack of trust in ML models that empower it by assessing their decision-making process. To that end, we must first investigate and evaluate the accuracy and credibility of classifications made by ML models used to process high-level intents. Then, we must 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 must investigate whether there is a viable method to improve the trust of operators in the decision made by ML models in both management loops. To pave the road towards solving the problem above, we formulated three research questions that guided the development of the thesis. The first research question concerns the use of ML techniques to parse relevant information from input network intents and compile them into a network configuration that fulfills the given intents.

RQ-1: In a self-driving network, can operators trust decisions and classifications made by current Machine Learning models used to configure the network based on high-level intents? To answer this question, we propose an end-to-end intent-based network management system with a conversational interface that allows operators to use natural language to define desired intents for the network.

RQ-2: In a self-driving network, can operators trust decisions and classifications made by current Machine Learning models used to self-configure the network based on monitored data analysis? 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.

RQ-3: Is there a feasible way to increase operator trust in the decision-making process of Machine Learning models that configure 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, which domain experts can use to spot issues in the decision-making process of the black-box model.

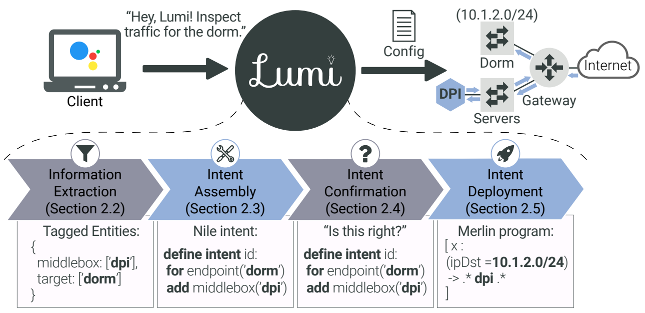
III. Machine Learning for IBN

Deploying policies in modern enterprise networks poses significant challenges for today's network operators. Since policies typically describe high-level goals or business intents, the operators must perform the complex and error-prone job of breaking each policy down into low-level tasks and deploying them in the physical or virtual devices of interest across the entire network. Recently, IBN has been proposed to solve this problem by allowing operators to specify high-level policies that express how the network should behave (*e.g.,* defining goals for quality of service, security, and performance) without having to worry about how the network is programmed to achieve the desired goals [6]. Ideally, IBN should enable an operator to simply tell the network to, for example, *"Inspect traffic for the dorm"*, with the network instantly and correctly breaking down such an intent into configurations and deploying them in the network. Supporting IBN is one of the core principles of a self-driving network as it reduces human intervention to a necessary minimum, which also reduces the number of errors introduced by human mistake.

In its current form, IBN has not yet delivered on its promise of fast, automated, and reliable policy deployment. One of the main reasons for this shortcoming is that, while network policies are generally documented in natural language, we cannot currently use them as input to intent-based management systems. Despite growing interest from some of the largest tech companies [3] and service providers [10], only a few research efforts [11] have exploited the use of natural language to interact with the network, but they lack support for IBN or other crucial features (*e.g.,* operator confirmation and feedback). However, expressing intents directly in natural language has numerous benefits when it comes to network policy deployment. For one, it avoids the many pitfalls of traditional policy deployment approaches, such as being forced to learn new network programming languages and vendor-specific Command-Line Interfaces (CLI), or introducing human errors while manually breaking down policies into configuration commands. At the same time, its appeal also derives from the fact that it allows operators to express the same intent using different phrasings. However, the flexibility makes it challenging to generate configurations, which must capture operator intent in an unambiguous and accurate manner, a feat not easily achieved when relying on ML models to extract information from natural language.

We contribute to the ongoing IBN efforts by describing the design and implementation of LUMI [12], 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 or enhancements.

A. LUMI *in a Nutshell*


"Hey, Lumi! Inspect traffic for the dorm."
Client
Lumi
Config
(10.1.2.0/24)
Dorm
Internet
DPI
Gateway
Servers
Information Extraction (Section 2.2)
Intent Assembly (Section 2.3)
Intent Confirmation (Section 2.4)
Intent Deployment (Section 2.5)
Tagged Entities:
Nile intent:
"Is this right?"
Merlin program:

```
{
 middlebox: ['dpi'],
 target: ['dorm']
}
```

```
define intent id:
 for endpoint('dorm')
 add middlebox('dpi')
```

```
define intent id:
 for endpoint('dorm')
 add middlebox('dpi')
```

```
[ x :
(ipDst =10.1.2.0/24)
 -> .* dpi .*
]
```

Fig. 2: The four modules of LUMI.

Figure 2 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, for the Information Extraction module, we rely on machine learning to extract and label entities from the operator utterances and implement them using a chatbot-like conversational interface. We build on existing ML algorithms for *Named Entity Recognition (NER)* [7] to extract and label entities from the operator's natural language utterances. In particular, we implement NER using a chatbot-like interface with multi-platform support and augment the existing algorithm so that LUMI can learn from operator-provided feedback.

Second, the extracted entities form the input of the Intent Assembly module, where they are used to compose a network intent. We introduce the *Network Intent Language* (Nile) [13], use it as an abstraction layer between natural language intents and network configuration commands for LUMI, and illustrate its ability to account for features critical to network management such as rate-limiting or usage quotas. *Nile* closely resembles natural language and reduces the need for operators to learn new policy languages for different types of networks.

Third, as part of the Intent Confirmation module, the output of the Intent Assembly module (*i.e.,* a syntactically-correct *Nile* intent) is presented to the network operator, and their feedback is solicited. If the feedback is negative, the system and the operator iterate until confirmation, with the system continuously learning from the operator's feedback to improve the accuracy of information labeling over time. Finally, once the system receives confirmation from the operator, the confirmed *Nile* intent is forwarded to the Intent Deployment module. This module's main task is to compile *Nile* intents into network configuration commands expressed in *Merlin* and deploy them in the network.

We evaluate LUMI's accuracy in information extraction, investigate LUMI's ability to learn from operator-provided feedback and measure both the compilation and deployment times in a standard campus topology. Using our own datasets consisting of synthesized intents as well as real-world intents derived from network policies published by 50 different campus networks in the US, we show that LUMI can extract entities with high precision, learn from the feedback provided by the operator, and compile and deploy intents in less than a second. In addition to an in-depth evaluation of LUMI, we also report our main findings of a small-scale user study, with 26 subjects. The study was performed to get feedback from subjects on the perceived value of using natural language for network management with LUMI and soliciting user feedback during the intent confirmation stage.

We implemented our prototype of LUMI using a combination of tools and libraries (*e.g.,* Google Dialogflow, *Scikit-learn* library). The full implementation and all datasets used in our evaluation are available on the project's website [12]. Together, the results of our evaluation and user study show that LUMI is a promising step towards realizing the vision of IBN of achieving fast, automated, and reliable policy deployment. By allowing operators to express intents in natural language, LUMI makes it possible for operators to simply talk to their network and tell it what to do, thus simplifying the jobs of network operators and also saving them time.

B. *Selected Results*

Evaluating systems like LUMI is challenging because of (i) a general lack of publicly available datasets that are suitable for this problem space (several operators we contacted in industry and academia gave proprietary reasons for not sharing such data), and (ii) difficulties in generating synthetic datasets that reflect the inherent ambiguities of real-world Natural Language Intents (NLIs).

To deal with this problem, we created two hand-annotated datasets for information extraction. The dataset *alpha* is "semi-realistic" in the sense that it is hand-crafted, consisting of 150 examples of network intents that we generated by emulating an actual operator giving commands to LUMI. In contrast, the *campi* dataset consists of real-world intents we obtained by crawling the websites of 50 US universities, manually parsing the publicly available documents that contained policies for operating their campus networks, and finally extracting one-phrase intents from the encountered public policies. From those 50 universities, we were able to extract a total of 50 different network intents. While some universities did not yield any intents, most universities published network policies related to usage quotas, rate-limiting, and ACL, and we were able to express all of them as *Nile* intents. We manually tagged the entities in each of these 200 intents to train and validate our information extraction model.

We used both datasets, separately and combined, to evaluate our NER model, with a 75%-25% training-testing random split. The small size of each dataset precludes us from performing conventional cross-validation. Table I shows the results for the datasets *alpha* dataset, *campi*, and a combination of them and illustrates the high accuracy of LUMI's information extraction module. Given the way we created the training examples for the *alpha* dataset, the excellent performance in terms of Precision, Recall, and F1-score is reassuring but not surprising. In creating the intents, we paid attention to extracting all entities defined in LUMI and also creating multiple intents for each entity class.

TABLE I: 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

At the same time, despite the largely unstructured nature and smaller number of intent examples in the *campi* dataset, the results for that dataset confirm the above observation. Even though the example intents were not designed with the NER model in mind, LUMI's performance remains excellent and is essentially insensitive to the differences in how the intent examples were generated. We attribute this success of LUMI at the information extraction stage to both continued advances in using machine learning for natural language processing and the fact that the complete set of LUMI-defined entities is relatively small and at the same time sufficiently expressive.

By demonstrating that LUMI can successfully deal with a wide range of network policies, this section represents a promising step towards realizing the vision of intent-based networking with natural language. Given the minimal human interaction from operators when expressing network intents to manage a self-driving network, it is possible to rely on humans to verify classifications from ML models applied to extracting relevant information from natural language intents, and correct them when necessary by providing feedback. This feedback loop enables operators to verify and, therefore, trust decisions and classifications made by such models.

IV. Machine Learning for Network Security

In the last few years, we have witnessed a growing tension in the networking community. Recent research has demonstrated the superiority of AI and ML models over simpler rule-based heuristics in identifying complex network traffic patterns for a wide range of network problems [14], [8]. At the same time, we have seen reluctance among operators when it comes to adopting these ML-based research artifacts in production settings [15]. The black-box nature of most of these proposed solutions is the primary reason for this lack of enthusiasm. 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 problems but applies more generally to any learning models, especially when their decision-making can have severe societal implications (*e.g.,* healthcare, credit rating, job applications, the criminal justice system). At the same time, this tension has also driven recent efforts to "crack open" the black-box learning models, explaining why and how they make their decisions (*e.g.,* "interpretable ML", "explainable AI (XAI)", and "trustworthy AI"). However, to ensure that these efforts are of practical use in particular application domains of AI/ML, such as network security, is challenging and requires further qualifying notions such as (model) interpretability or trust (in a model) [16] and also demands solving several fundamental research problems in these new areas of AI/ML.

In this section, we focus on the application of AI/ML to network security and the challenges that it poses. For one, there exists an obvious mismatch between the black-box nature of some of the most commonly considered AI/ML models and what network practitioners expect from or look for in a new technology like AI/ML. 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"* [16]. Given this specific definition of what it means for an AI/ML model to engender trust, we next address research challenges related to quantitatively deciding when an end user is comfortable with relinquishing control to a given AI/ML model. To this end, a focus of this section is on determining whether or not a given AI/ML model suffers from the *problem of underspecification* [17].

The problem of underspecification in modern AI/ML refers to determining whether the success of a trained model (*e.g.,* high accuracy) is indeed due to its innate ability to encode some essential structure of the underlying system or data or is simply the result of some inductive biases that the trained model happens to encode. 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 [18] or spurious correlations [19]). Such inductive biases imply that their presence in trained AI/ML models prevents these models from being credible or trustworthy; that is, they generalize as expected in deployment scenarios. 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 a first step towards achieving this ambitious goal.

A. TRUSTEE

To detect underspecification issues in learning models for network security problems, we develop TRUSTEE (TRUSt-oriented decision TreE Extraction) [20]. TRUSTEE provides a means for carefully inspecting black-box learning models for the presence of inductive biases. Figure 3 shows how TRUSTEE augments the traditional ML pipeline to examine the trustworthiness of a given ML model. Specifically developed with network security in mind, TRUSTEE takes a given black-box model and the dataset that has been used to train that model as input and outputs a "white-box" model in the form of a high-quality decision tree (DT) explanation.

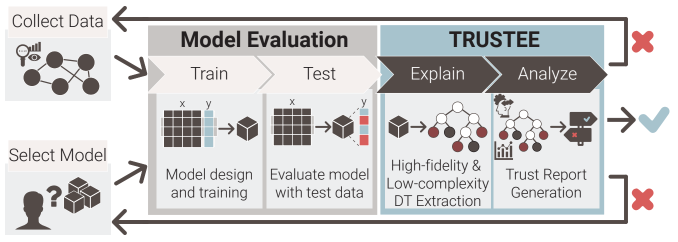

Collect Data
Select Model
Model Evaluation
Train
Test
Model design and training
Evaluate model with test data
TRUSTEE
Explain
Analyze
High-fidelity & Low-complexity DT Extraction
Trust Report Generation

Fig. 3: TRUSTEE overview.

Importantly, in synthesizing these decision trees, TRUSTEE's focus is first and foremost on ensuring their practical use, which, in turn, requires leveraging domain-specific observations to strike a balance between *model fidelity* (*i.e.,* accuracy of the decision tree) and *model complexity*. 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 the problem of 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.

While our work contributes to the rapidly growing ML literature on model explainability/interpretability and is inspired by ongoing developments in this area, our efforts and objectives differ from existing approaches in a number of significant ways. First, given the inherent complexity of learning problems for networking, the existing approaches of replacing black-box models with "white-box" models such as decision trees are generally impractical. For example, among existing methods that are concerned with *local interpretability* (*i.e.,* can at best explain the decisions of a trained AI/ML model in a local region near a particular data point; *e.g.,* see SHAP [21], LEMNA [22]), are not suitable for examining the various instances of the underspecification problem. Second, though our effort is inspired by prior work that focus on *global interpretability* (*i.e.,* explains how a trained ML model makes decisions as a whole [23] [24]), they are also not suited for the problem at hand. These methods are either only applicable to a specific class of learning models (*e.g.,* reinforcement learning) or suffer from poor fidelity. In their current form, these existing methods are all insufficient for providing the level of model explainability that we demand so that network operators can decide if they are comfortable with relinquishing control to a given black-box model.

B. *Use Cases Summary*

We apply TRUSTEE to scrutinize a number of recently published black-box models that have been developed for network security-related problems and are accompanied by publicly available artifacts that are required for assessing whether the models are credible. All datasets, models, and results presented in this section are made publicly available [20].

Table II summarizes our use cases. 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). It highlights the importance of having an in-depth understanding of the data used to train a model. The second use case analyzes a black-box model (*i.e.,* random forest) trained using the synthetic dataset CIC-IDS-2017 [25] 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 analyzes an approach that advocates using bit-level feature representations of the input data instead of carefully engineered and semantically meaningful features [26]. This use case warns against the indiscriminate use of the high-dimensional feature spaces that result from

TABLE II: 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

such representations because they allow black-box models to identify and exploit spurious correlations between features. The fourth use case concerns the application of a complex ensemble of neural networks [27] to perform traffic anomaly detection (*e.g.,* Mirai attack). By showing that this model is also vulnerable to o.o.d. samples, we corroborate previously-reported criticism of this model [15] and support it with further evidence. We also used TRUSTEE to analyze other ML-based models for networking and network security problems in the literature, which we briefly describe in the thesis. Because of a lack of space, we do not show the decision tree explanations extracted from the models nor their validation, but they can be found at [20].

The use cases analyzed show that ML models widely applied to solve networking problems fail to fulfill their tasks when put under minimal adverse circumstances. Since it is unfeasible for an operator to verify every decision in a zero-touch management loop manually, our results indicate that operators are, in fact, correct not to trust ML models to configure the network based solely on monitored data automatically. However, by employing a new research pipeline for AI/ML provided by TRUSTEE, one can expose decisions made by black-box ML classifiers and therefore increase trust in those decisions. Given this knowledge of the "inner-workings" of ML models, operators can choose to relinquish control to the ML model if they agree with the decision made by the model.

V. CONCLUSIONS

This thesis has investigated the use of ML techniques in the general design of a self-driving network. The indiscriminate use of ML to solve network management tasks raises an inherent lack of trust in the black-box classifiers among network practitioners. To tackle this problem this lack of trust, this thesis analyzed and evaluated the decision-making process of ML-based classifiers that compose a self-driving network. We also propose a novel method to uncover the "inner-workings" of any given black-box ML model, which operators and domain experts can leverage to gain trust in decisions made by such ML models. Our results indicate that ML models employed to extract information from natural language intents can be trusted, with minimal safeguarding and human-in-the-loop verification. However, ML models that automatically solve network security and performance problems have not been put under proper scrutiny and can easily break when put under stress, failing to fulfill their given tasks properly.

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

[1] N. Feamster et al. Why (and How) Networks Should Run Themselves. ANRW '18. ACM, 2018.

[2] J. Apostolopoulos. Improving Networks with Artificial Intelligence, Oct 2020. https://blogs.cisco.com/networking/improving-networks-with-ai.

[3] Juniper Net. What Is Intent-Based Networking? https://juni.pr/3Vnfz5s.

[4] Huawei. Huawei Core Network Autonomous Driving Network White Paper, Nov 2019. Accessed at 2022-07-06.

[5] N. Foster, et al. Using Deep Programmability to Put Network Owners in Control. *SIGCOMM CCR*, 50(4), October 2020.

[6] A. Clemm, et al. Intent-Based Networking - Concepts and Definitions. Internet-draft, Internet Engineering Task Force, December 2019.

[7] D. Jurafsky et al. *Speech and Language Processing*. 3rd edition, 2019.

[8] R. Boutaba, et al. A comprehensive survey on machine learning for networking: evolution, applications and research opportunities. *Journal of Internet Services and Applications*, 9(1), 2018.

[9] A. S. Jacobs, et al. Enabling Self-Driving Networks with Machine Learning, 2022. PhD Thesis. **Avaialble at https://bit.ly/3UhFlXq**.

[10] H. H. Liu. The Practice of Network Verification in Alibaba's Global WAN, May 2021. https://bit.ly/3XNjfiw.

[11] R. Birkner, et al. Net2Text: Query-Guided Summarization of Network Forwarding Behaviors. NSDI '18. USENIX, 2018.

[12] A. S. Jacobs, et al. Hey, Lumi! Using Natural Language for Intent-Based Network Management. USENIX ATC 21, July 2021.

[13] A. S. Jacobs, et al. Refining Network Intents for Self-driving Networks. *SIGCOMM CCR*, 48(5), January 2019. **Best Paper Award.**

[14] A. S. Jacobs, et al. Artificial neural network model to predict affinity for virtual network functions. NOMS '18, April 2018.

[15] D. Arp, et al. Dos and Dont's of Machine Learning in Computer Security. In *USENIX Security 22*, August 2022.

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

[17] A. D'Amour, et al. Underspecification Presents Challenges for Credibility in Modern Machine Learning, 2020.

[18] R. Geirhos, et al. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11), Nov 2020.

[19] M. Arjovsky, et al. Invariant Risk Minimization, 2020.

[20] A. S. Jacobs, et al. AI/ML for Network Security: The Emperor Has No Clothes. ACM CCS '22, 2022. **Best Paper Honorable Mention.**

[21] L. S. Shapley. *A Value for n-Person Games*. Princeton U. Press, 2016.

[22] W. Guo, et al. LEMNA: Explaining Deep Learning Based Security Applications. ACM CCS '18, 2018.

[23] O. Bastani, et al. Verifiable Reinforcement Learning via Policy Extraction. NIPS'18. Curran Associates Inc., 2018.

[24] H. Lakkaraju et al. "How Do I Fool You?": Manipulating User Trust via Misleading Black Box Explanations. ACM AIES '20, 2020.

[25] I. Sharafaldin., et al. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. In *ICISSP*, 2018.

[26] J. Holland, et al. New Directions in Automated Traffic Analysis. ACM CCS '21, 2021.

[27] Y. Mirsky, et al. Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection. NDSS'18, 2018.