Runtime Verification for Deep Learning Systems

Birk Torpmann-Hagen[1], Michael A. Riegler[2], Pål Halvorsen[2] and Dag Johansen[1]
[1]*UiT The Arctic University of Norway, Tromsø, Norway*
[2]*SimulaMet, Oslo, Norway*

Keywords: Deep Learning, Runtime Verification, Transparency, Distributional Shift, Multimedia.

Abstract: Deep Neural Networks are being utilized in increasingly numerous software systems and across a wide range of data modalities. While this affords many opportunities, recent work has also shown that deep learning systems often fail to perform up to specification in deployment scenarios, despite initial tests often indicating excellent results. This disparity can be attributed to shifts in the nature of the input data at deployment time and the infeasibility of generating test cases that sufficiently represent data that have undergone such shifts. To address this, we leverage recent advances in uncertainty quantification for deep neural networks and outline a framework for developing runtime verification support for deep learning systems. This increases the resilience of the system in deployment conditions and provides an increased degree of transparency with respect to the system's overall real-world performance. As part of our framework, we review and systematize disparate work on quantitative methods of detecting and characterizing various failure modes in deep learning systems, which we in turn consolidate into a comprehensive framework for the implementation of flexible runtime monitors. Our framework is based on requirements analysis, and includes support for multimedia systems and online learning. As the methods we review have already been empirically verified in their respective works, we illustrate the potential of our framework through a proof-of-concept multimedia diagnostic support system architecture that utilizes our framework. Finally, we suggest directions for future research into more advanced instrumentation methods and various framework extensions. Overall, we envision that runtime verification may endow multimedia deep learning systems with the necessary resilience required for deployment in real-world applications.

1 INTRODUCTION

Deep learning is considered the state-of-the-art approach for a wide variety of tasks and is being utilized in a growing number of software systems. Nevertheless, Deep Neural Networks (DNNs) have been shown to exhibit several shortcomings in deployment settings that are not generally observed at the development stage (Paleyes et al., 2022; Lwakatare et al., 2020). In particular, it has been shown that DNNs readily fail to generalize to data that exhibit properties that are not well-represented in the training data, for instance different lighting-conditions (Ali et al., 2022), demographic stratification (Oakden-Rayner et al., 2020), image-noise (Hendrycks and Dietterich, 2019), or other, more subtle changes (Geirhos et al., 2020). In addition to this, there are also significant issues with respect to fairness (Holstein et al., 2019), privacy (Mireshghallah et al., 2020), and security (Liu et al., 2021). Current state-of-the-art large language models have also been shown to exhibit severe problems with respect to reliability (Liu et al., 2024) and hallucinations (Azamfirei et al., 2023). While deep learning researchers are making strides towards addressing these shortcomings, DNNs still lack the necessary degree of resilience that would warrant their implementation in many software systems (Paleyes et al., 2022). This is of particular relevance to high-stakes domains such as autonomous vehicles, medical systems, and algorithmic decision-making, wherein any of the aforementioned failures could incur significant consequences for stakeholders if not immediately recognized and reacted to by a human in the loop.

Many of these issues can be attributed to the sensitivity of neural networks with respect to shifts in the nature of the data – i.e. *distributional shifts* – and are further complicated by the fact that there is a lack of testing methodologies capable of uncovering these shortcomings at the development stage. This means that these shortcomings are often not identified until

Torpmann-Hagen, B., Riegler, M. A., Halvorsen, P. and Johansen, D.
Runtime Verification for Deep Learning Systems.
DOI: 10.5220/0013195500003928
In *Proceedings of the 20th International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE 2025)*, pages 367-377
ISBN: 978-989-758-742-9; ISSN: 2184-4895

after they have already manifested in a deployment scenario. Although it is common development practice to test neural networks using hold-out datasets or through cross-validation, this does not generally constitute a realistic representation of the network's accuracy during deployment. Indeed, many of the shortcomings outlined above can only be discovered after administering rigorous stress-tests or assessing the constituent networks on an entirely new, manually labeled, and curated out-of-distribution dataset (Ali et al., 2022; Geirhos et al., 2020; Winkler et al., 2019; D'Amour et al., 2020; Li et al., 2022). While large-scale testing of this kind already constitutes a significant improvement over the current standard practice of simply partitioning the training dataset (Li et al., 2022), this is often infeasible in production contexts to the required costs of data-curation and annotation, and is in any case unlikely to account for all the possible edge-cases that may be encountered in a deployment scenario (Li et al., 2022; Riccio et al., 2020). In conventional software systems, such edge-cases are often identified through input fuzzing techniques (Myers et al., 2011), but this is not generally a feasible approach with respect to the testing of neural networks. Whereas conventional software systems are generally characterized by a well-defined state- and input-space from which test-cases can be sampled (Myers et al., 2011), deep learning systems operate on manifolds of high-dimensional data (Brahma et al., 2016). Sampling inputs that sufficiently represent the scope of variability present in a deployment scenario thus requires accurately modeling this manifold, a task of equal difficulty to training a network capable of perfect generalization (Li et al., 2022; Riccio et al., 2020).

Current approaches to the verification of deep learning systems are thus not typically successful in their objective of ascertaining whether or not a given DNN will perform up to specification during deployment, and deep learning systems often fail in deployment scenarios as a result. Taking inspiration from high-stakes systems engineering disciplines (Lindemann et al., 2023; Francalanza et al., 2018; Pike et al., 2012; Falcone et al., 2013), we contend that these issues can be largely mitigated through the implementation of *runtime verification (RV)* as system support. This provides an effective safeguard against insufficient testing at development time, and circumvents the problem of representative test-case generation by simply extracting test-cases as the system is deployed and executing. While these inputs are necessarily unlabeled, recent advances in DNN uncertainty quantification has shown that it is nevertheless possible to extract meaningful surrogates of system performance from DNN representations of the input data alone (Yang et al., 2022; Zhang et al., 2022). By continuously monitoring these surrogates across all data sources within the system at runtime, deviations from specified behavior can be detected and reacted upon as they arise, mitigating – and, given suitable fall-back measures, possibly preventing – system failure.

To this end, we introduce a framework for the development of RV support for deep learning systems. Our framework is based on requirements analysis, and involves the composition of various online testing methods that evaluate surrogates of system properties, selected in accordance to the system's requirements and data modalities. Our framework is compatible with several different data modalities, can account for the implementation of active-learning, and can be used to develop RV for complex multimedia systems. We support our arguments with a review of suitable online testing methods extracted from disparate work on distributional-shift detection, fairness in machine learning, adversarial attacks, guardrails for Large Language Models (LLMs), and more. As the efficacy of these methods have already been empirically verified in their respective works, we forego additional empirical analysis in favor of demonstrating the potential of our framework through a proof-of-concept multimedia medical deep learning system architecture.

We summarize our contributions as follows:

- We outline a general-purpose framework for developing robust deep-learning systems with RV, including support for multimedia systems and active learning.
- We review, systematize, and unify previous work on uncertainty quantification and verification of deep neural networks, highlighting existing methods suitable for RV.
- We contextualize our work in terms of an example system architecture consisting of a medical multimedia system.
- We propose several avenues of further investigation and development, including an outline for a probabilistic risk-assessment framework for deep learning systems.

We organize our work as follows: in Section 2, we review shortcomings of deep learning systems, outline existing work on the verification of deep learning systems, and relate the concept of RV in conventional software systems to deep learning systems. In Section 3, we outline our proposed framework, including a review of methods and metrics in the literature that are suitable for use as runtime monitors. In Section 4,

we outline an example of a multi-modal medical deep learning system that implements our framework. Finally, we discuss the potential of this framework towards endowing deep learning systems with operational resilience and propose several promising directions for future research.

2 RELATED WORK

In this section, we outline related work on the shortcomings and verification of deep learning systems, and outline the benefits of RV in other domains.

The shortcomings of DNNs can be understood in terms of the affected system properties. This often includes failures with regards of correctness, robustness, fairness, explainability, scalability, privacy, security, etc. The correctness of a DNN is, for instance, only guaranteed given that the data is identically distributed to the data with which it was trained and validated (Goodfellow et al., 2016). Violations of this assumption results in *generalization failure*. This has, for instance, been studied in the polyp-segmentation domain, where a change in center or lighting conditions was shown to significantly reduce polyp detection rates, despite utilizing generalization-oriented training procedures (Ali et al., 2022). In another study it was shown that medical systems for skin-cancer diagnosis and diabetic retinopathy detection failed to generalize to underrepresented skin-tones and imaging equipment respectively (D'Amour et al., 2020). In general, vision-models have been shown to be sensitive to adversarial attacks and natural corruptions such as additive noise or blurs (Hendrycks and Dietterich, 2019). State-of-the-art LLMs have been shown to fail to generalize to sentence reversal (Berglund et al., 2023), often fabricate information in outputs (Azamfirei et al., 2023), and readily yield harmful or otherwise unethical outputs (Yao et al., 2024). They have also been shown to be prone to generating incorrect answers when prompted with leading questions or continued prompt contradiction, a phenomenon often referred to as *LLM-sycophancy* (Wang et al., 2023). It is often argued that these errors can be elucidated through the use of explanations (Adadi and Berrada, 2018), based on the assumption that incorrect predictions will yield incongruous explanations. This view has been challenged by recent work, however, which has shown that explanations are often misleading (Rudin, 2019) and can be easily fooled by adversarial perturbations (Heo et al., 2019). DNNs have also been shown to often lack the fairness necessary for responsible deployment in domains with socially salient outcomes (Holstein et al., 2019), as exemplified in a aforementioned study on skin-cancer diagnosis. Credit-scoring systems have also been shown to readily learn biases with respect to gender, race, and other such variables, despite these variables not appearing in the training data (Hurlin et al., 2024). Privacy violations, though somewhat less frequently studied, are also commonplace in deep learning systems. It has, for instance, been shown that it is possible to recover confidential training data using model inversion (Song and Namiot, 2023) and that it may be possible to re-identify users from anonymized medical signal data (Ghazarian et al., 2022).

In the majority of the aforementioned works, system failures were only identified after administering rigorous stress-tests and testing the system on unseen, Out of Distribution (OOD) data. Such testing has been shown to generally not be standard practice in the industry due to high costs and a general lack of suitable testing frameworks (Li et al., 2022; Riccio et al., 2020). The majority of deep learning systems are instead typically evaluated by computing simple correctness metrics on hold-out sets, which generally do not represent the variability present in a deployment scenario. While synthetic test-input generation, for instance through generative modeling with OOD seed data (Pei et al., 2017), offer a partial solution, the efficacy of these methods has been contested (Li et al., 2022).

In conventional software engineering and other systems engineering disciplines, too, the state-spaces can be so large that conventional testing is considered insufficient. Where system failures can incur significant consequences, another solution is to use RV. RV is the study of algorithms, metrics, and methods for understanding the behavior of an executing system (Falcone et al., 2013; Falcone et al., 2021). Implementing these methods in software systems allows increased confidence that the system is working as specified at runtime. The RV process is illustrated in Figure 1, and can be distinguished according to two distinct components: the *monitor*, which tracks the events or states of the system and compares these against predefined correctness criteria, and the *instrumentation*, which refers to the process or techniques by which the system's behaviour is characterized. At runtime, the monitor reads from the system instrumentation and yields a *verdict* in accordance with the specified acceptable range of the observations. The verdict can, in turn, be used to activate fall-back measures or generate system-wide alerts.

RV is widely used in high-stakes systems. NASA's *copilot* (Perez et al., 2020), for instance, is a form of RV designed to detect and react to system failures aboard space shuttles. Similar methods

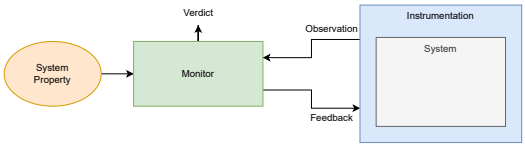

Figure 1: The RV process.

are also often utilized in aviation (Pike et al., 2012) and semi-autonomous vehicles (Lindemann et al., 2023). Decentralized and distributed systems also often implement some form of RV, typically to ensure that crashed or faulty nodes do not adversely affect the overall system or to resolve conflicting messages within the network (Francalanza et al., 2018). RV has a distinct advantage compared to conventional testing, precisely in that the system can be verified on whatever data is required *at runtime*. With conventional testing, it would be necessary to generate a test case that represents this data – and, depending on requirements, all possible data – but with RV, test cases are extracted from the execution itself, with the main obstacle being the design of test criteria. Though RV may be less robust than conventional testing as a result, it significantly mitigates any undesirable behavior from untested inputs, as exemplified in the aforementioned applications.

3 RUNTIME VERIFICATION FOR DEEP LEARNING SYSTEMS

So far, we have shown that DNNs exhibit a number of shortcomings that greatly hinder their utility as components of software systems. We have also outlined how they are difficult to test sufficiently due to their large input spaces, and how similarly complex systems in other disciplines greatly benefit from RV. It therefore stands to reason that that deep learning systems, too, stand to benefit from implementing RV methods as a system support. In this section, we thus outline a framework for the development of RV for deep learning systems based on requirements analysis, and provide an overview of suitable methods thereto as described in other literature, which we organize according to data modalities.

The framework is summarized in Figure 2. The process starts at the requirements specification stage. For the system requirements to be satisfied at runtime, the corresponding properties – e.g. correctness, fairness, safety, etc – must be actively monitored.

Next, it is necessary to identify any parts of the system that may be prone to failures with respect to each of the aforementioned requirements, for instance, data sources, intermediate results in pipelined models, explanations, system outputs, and user inputs. We refer to these collectively as *data paths*. Increasingly complex systems contain larger amounts of these data paths, each of which may require monitoring. Consider, for instance, a system consisting of a single model that requires two input data sources. It is not sufficient to monitor only one of these data sources, and analyzing both sources simultaneously requires data fusion (Baltrušaitis et al., 2017; Lahat et al., 2015), which may introduce additional errors. Similarly, intermediates in pipelined models – for example, where the output of one DNN is used as the input to another – must be monitored in order to avoid error propagation (Srivastava et al., 2020). As implementing monitors for each requirement in each data path is likely to incur significant computational costs, it is, however, necessary to assess what system properties are at risk at each path according to a threat model. For example, if it is unlikely that a data source contains privacy violations, there is no point in implementing a privacy monitor on this path. If a threat to the system's ability to meet its specified requirements has been identified for a given data path, it is considered a *critical* data path and is determined to require monitoring. Distinguishing between threats across data sources and domains is particularly important in systems composed of multiple modalities – i.e. multimedia systems – due to the increased complexity that arises due to cross-modal contexts. It has, for instance, been shown that state-of-the-art multimodal LLMs can be jail-broken if certain images are provided alongside a prompt that the guardrails would otherwise deem harmful (Li et al., 2024b). In this case, monitoring both images and prompts would be necessary for the attack to be detected.

Once these critical data paths have been identified, the next step is to implement suitable monitors for each of them. This requires a means by which the incidence of system failures can can be characterized. Whereas this may be fairly straightforward in conventional systems, one cannot generally characterize such violations precisely in deep learning systems. Instead, it is necessary to develop sufficiently expressive surrogates for each system requirement as part of the instrumentation development process. To aid in this, we provide a brief overview of potential threats to overall system performance in terms of system properties alongside suitable instrumentation algorithms as presented in other literature in Table 1. To benefit a wide array of different application domains, we organize these methods according to data modalities. We note that this list is by no means comprehensive and that the development of specialized monitoring methods

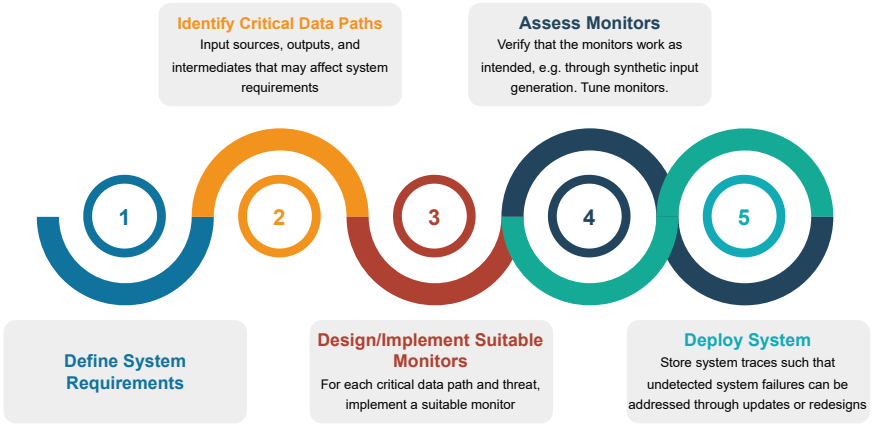

Figure 2: RV development process for deep learning systems.

may be required in production systems. Moreover, it is worth noting that the scope of efficacy of these methods is not yet fully understood. The credibility of the reported accuracy of many adversarial attack detectors has, for instance, been contested (Tramèr, 2022).

After suitable monitors have been implemented at each critical data path, the next step is to verify that the monitors perform to a sufficient standard. Though this might appear to counteract the stated benefits of using RV methods in the first place – i.e., not requiring extensive testing – it is nevertheless necessary, as RV are in and of themselves software artifacts (Falcone et al., 2021). Testing RV methods is generally easier, however, as monitors can generally be unit-tested and are characterized by more well-defined problem statements, as evidenced by the extensively developed methodologies conventionally used to evaluate Distributional Shift Detectors (DSDs). Generating suitable test cases is also expedited by the fact that the monitors yield a single binary verdict, and that monitors operate on surrogates for system properties as opposed to the specifications themselves. For instance, verifying that a correctness monitor works as expected may simply requite testing whether a given DSD correctly identifies synthetically augmented data as OOD, rather than testing whether the DNN yields accuracies within a specific interval for all plausible input data.

Finally, it is necessary to continuously collect system logs for each critical data path and monitor the verdict once the system has been deployed. This facilitates the continued improvement of the monitors themselves by permitting analysis of system failures the monitors have failed to detect. If the system implements some form of active learning or other forms of feedback loops, these logs can then also inform corresponding monitor updates. This may, for instance, be achieved through re-calibration of the monitors at set intervals and be incorporated as a regular form of system maintenance.

4 EXAMPLE: RUNTIME VERIFICATION FOR A MEDICAL MULTIMEDIA SYSTEM

To illustrate the potential benefits of RV in production scenarios, we include an example of a system architecture that implements our framework for a diagnostic support system for heart disease detection. The system inputs consist of several data modalities, including time-series data in the form of cardiographs, visual data in the form of echocardidograms and tabular data in the form of patient information such as medical history, age, etc. The central model in this architecture, LLaVA-Med (Li et al., 2023), accepts this data as input and yields a diagnosis, an initial prognosis, and suggests treatment if required, all in accordance with a clinician's prompt(s).

In accordance with our framework, we start with defining some system requirements. Critical data paths are then identified, along with suitable monitors for each path. Though we have provided examples of suitable monitor algorithms, we note that in a production scenario, these choices should ideally be informed by prototyping. The results are shown in Table 2, and the resulting architecture is shown in in Figure 3.

Though we do not assess this architecture empirically, the results in each of the cited works for each monitor indicate that this system architecture would, in a deployment context, exhibit a significantly improved degree of robustness, safety, privacy, trans-

Table 1: Overview of typical threats to deep learning system performance and examples of viable RV monitors, separated according to data modality. * = survey papers; ? = unable to find suitable monitors.

Modality	Property	Threat(s)	Suitable Monitors
Images & Video	Correctness	Generalization Failure (D'Amour et al., 2020; Geirhos et al., 2020; Kauffmann et al., 2020)	(Yang et al., 2022)*
	Robustness	Corruptions, Noise (Hendrycks and Dietterich, 2019)	(Yang et al., 2022)*
	Privacy	Model Inversion (Song and Namiot, 2023)	(Song and Namiot, 2024)
	Safety	Adversarial Attacks (Biggio et al., 2013), data poisoning (Schwarzschild et al., 2021)	(Harder et al., 2021; Zheng and Hong, 2018; Metzen et al., 2017), (Paudice et al., 2018; Steinhardt et al., 2017; Wang et al., 2019)
	Explainability	Misleading explanations	(Zhou et al., 2021)*
	Fairness	Unfair predictions, model bias, unfair performance	(Chen et al., 2024), ?
	Compliance	Intellectual Property violations	?
Text	Correctness	Generalization Failure (Berglund et al., 2023; Perez-Cruz and Shin, 2024), Misinformation	(Achintalwar et al., 2024; Li et al., 2024a)*
	Robustness	Sentence Reversal (Berglund et al., 2023)	?
	Privacy	Model Inversion (Song and Namiot, 2023)	?
	Safety	Jail-breaking, unsafe output	(Alon and Kamfonas, 2023; Inan et al., 2023; Yuan et al., 2024; Li et al., 2024b), (Achintalwar et al., 2024)*
	Explainability	Sycophancy(Wang et al., 2023)	?
	Fairness	Latent Bias (Chu et al., 2024)	(Chu et al., 2024)*
	Compliance	Intellectual Property violations	?
Audio	Correctness	Generalization Failure	(Yang et al., 2022)
	Robustness	Noise and Poor Acoustics (Xu et al., 2021)	?
	Privacy	User-identification (Williams et al., 2021)	?
	Safety	Adversarial Attacks (Carlini and Wagner, 2018)	(Ma et al., 2021; Harder et al., 2021)
	Compliance	Intellectual Property violations (Blumenthal, 2024)	?
Signals	Correctness	Generalization Failure	(Yang et al., 2022)*(Provotar et al., 2019)
	Robustness	Sensor Drift (Zheng and Paiva, 2021)	(Yang et al., 2022; Marathe et al., 2021)
	Privacy	Re-identification (Ghazarian et al., 2022)	?
Tabular Data	Correctness	Incomplete/Invalid data	(Hynes et al., 2017)
	Safety	Data Poisoning (Schwarzschild et al., 2021)	(Steinhardt et al., 2017)
	Fairness	Algorithmic Bias (Holstein et al., 2019)	(Chen et al., 2024)
	Explainability	Misleading Explanations (Heo et al., 2019)	?

parency, and trustworthiness over a naive, monitor-free implementation. A full picture of the value of such a system architecture does, however, require integration testing, ideally as part of a field study, though this is beyond the scope of this paper.

Table 2: System requirements and suitable monitors for a diagnostic support system.

Property	Requirement	Threat(s)	Critical Data Paths	Monitor
Correctness	The system should assign correct diagnoses at rates comparable to a trained clinician	Distributional Shift, incomplete data, leading prompts	All data sources	OOD-detector (Yang et al., 2022), data-lintering (Hynes et al., 2017), sycophancy-monitoring (Achintalwar et al., 2024)
Robustness	The model should be robust to minor perturbations	Noise, blurs, adversarial attacks	Echocardiogram data, EKG data	Adversarial attack detection (Harder et al., 2021)
Explainability	The factors informing diagnosis need to be communicated with respect to each data stream	Stakeholders do not trust system/ insufficient regulatory compliance	Explanation outputs	Explanation quality metrics (Zhou et al., 2021)
Safety	The system should raise alerts when yielding diagnoses with high-impact or otherwise risky treatment plans	System suggests treatments that can interact adversely with patient history/medication or otherwise exhibit high risks	LLaVA-Med outputs	Database check of treatment plans, their side-effects, and overall risk assessment
Privacy	The patient's privacy and the privacy of previous patients must be maintained, including training data	Privacy violations in generated outputs	LLaVA-Med outputs, Explanations	Privacy Guard-rails (Achintalwar et al., 2024)
Safety/ Extensibility	If monitors indicate impeded performance, the clinician's manual diagnosis and patient data should be used to update the model	Data sources have been tampered with or are otherwise poor quality, clinician has failed to enter ground-truth correctly	Clinician diagnosis and treatment plan, data sources	Data poisoning detection(Paudice et al., 2018)
Fairness	Outputs should be invariant to non-causal patient properties, e.g. race, income	System yields unfair outcomes	LLaVA-Med outputs	Fairness tests(Chen et al., 2024)

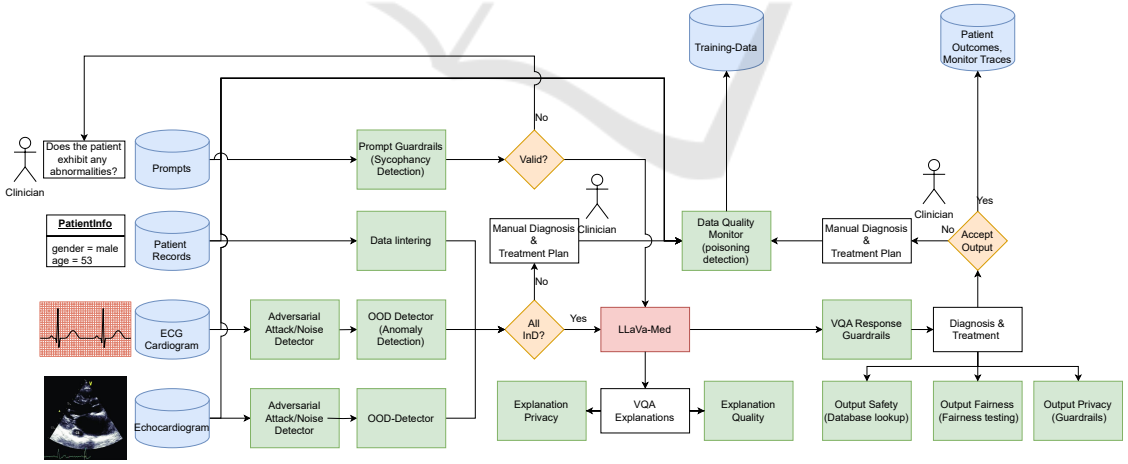

Figure 3: Medical Diagnostic Support System. Green blocks correspond to monitors. For simplicity, we have not specified fallback measures for all monitors except where positive verdicts raise the need for further processing.

5 DISCUSSION

As illustrated by the example in Section 4 and the various suitable monitors reviewed in Section 3, RV already has the potential to endow deep learning multimedia systems with an increased degree of fault-tolerance, increasing the feasibility of real-world de-

ployment of deep learning systems in performance-critical scenarios. In contrast to the majority of current implementations of deep learning systems, wherein system failures manifest silently, our framework permits the detection of system failures at runtime. Monitor verdicts can then be utilized as signals to engage fallback measures, for instance in the form of human intervention or user alerts. In the medical system illustrated in Section 4, the use of monitors may contribute to more efficient and accurate diagnoses and treatment of patients, ensure regulatory compliance, and expedite any required analysis should there be a need to troubleshoot or improve the system.

Beyond medical domains, we also envision that our framework may greatly benefit systems for automated journalism and content creation. One may, for instance, develop and implement monitors that assess the factual accuracy of generated content, identify potential sources of bias, and/or highlight ethical considerations, for instance, built on the same algorithmic principles as LLM guardrails. Automated content filtering and recommendation systems might also benefit, in particular, given sufficient monitoring at all critical data paths. Incitements to violence, hate-speech, deep-fakes, or other undesired content may be monitored holistically and with respect to both the audio, video, and textual modalities of social media platforms given sufficient algorithmic development. Individual monitors may in these cases be easily updated to adapt to changes in how undesirable content manifests with regards to terminology, imagery, or stakeholder values, facilitated by monitor trace logging and the compartmentalization afforded by our framework.

We envision that RV system support can become as ubiquitous in deep learning systems as it is in other critical software systems. Further research is required to this end, however, in particular with regards to the development of instrumentation that can serve as effective surrogates for system requirements. A large majority of existing research in this regard has been scattered around the development of robustness- and correctness monitors for image-based systems in the form of DSDs (Yang et al., 2022) and adversarial attack detectors (Harder et al., 2021; Zheng and Hong, 2018; Metzen et al., 2017). There has, however, been sparse research towards equivalent methods for large language models, and sparser still with respect to tabular data and time-series data. Of equal concern is the scarcity of monitoring methods for more cross-cutting system properties, such as security, privacy, and fairness. Though this has recently been more extensively studied in the context of LLMs guard-rails, there is a need for equivalent efforts for other modalities. To the best of our knowledge, there are for instance currently no existing methods of detecting fairness violations in the form of correctness discrepancies across sub-populations, nor any methods of monitoring for privacy violations in image data, such as the detection of uncensored watermarks, faces, or logos. For reference, consider the sparsity of citations in certain entries of Table 1.

Though in this work, we have included a number of suitable examples of runtime monitors and instrumentation, a more extensive survey of instrumentation methods is warranted. Given sufficient further development, a library implementing a suite of such methods may also be useful, both for the purposes of engineering and for more granular evaluation of deep neural networks in academic contexts.

Further research into monitor instrumentation is hindered by a lack of sufficient benchmarks. As mentioned in Section 3, test-cases are required in order to sufficiently evaluate runtime monitors. For the detection of distributional shift and adversarial attacks, generating test cases is fairly simple, often only requiring the synthetic modification of training data. On the other hand, obtaining test cases for other system properties may be challenging. The development of a suitable explanation quality monitor, for instance, would require some notion of what a suitable explanation entails. This requires qualitative assessments from end-users, which is time-consuming and labour-intensive. Equivalently, verifying that a fairness monitor correctly detects unfair behaviour requires a test-bed that contains representative examples of the various forms of fairness violations that may occur. We thus also call for the development of datasets and benchmarks that lend themselves to the development of RV methods. This could, for instance, entail packaging datasets with two separate model checkpoints and/or dataset folds, one that satisfies some system requirement and one that does not. Profiling these methods with respect to execution time and resource requirements is also warranted. Many multimedia systems have strict technical requirements, and existing monitoring methods have not generally been developed with execution time and resource usage in mind. In domains where real-time execution is necessary, there may also be a trade-off with respect to the overhead introduced by the added monitors and the reductions in the DNN's parameter counts and auxiliary processing this overhead would necessitate.

Constructing monitors with other monitor outputs as instrumentation components may also contribute to greater insight into the system's function, in particular in domains requiring multi-modal data. Fairness with respect to correctness may, for instance,

be monitored by keeping track of any differences in prediction rates of distributional-shift detectors with respect to any pertinent variables. In the system outlined in Section 4, for instance, one might verify that the system yields fair outputs by monitoring for disparate distributional-shift detector prediction rates with respect to patient data such as age, gender, race, etc. The explanation quality monitor might, in an equivalent manner, be improved by utilizing noise detector and prompt monitor outputs, for instance, by preventing explanation-affecting adversarial attacks (Heo et al., 2019) or mitigating false explanations due to leading questions in the prompts.

A possible additional benefit of utilizing RV for deep learning is that it may facilitate the development of a probabilistic risk assessment framework. Probabilistic risk assessment is common in high-stakes domains (Stamatelatos et al., 2011) and involves statistical analysis of the rates of system failures. RV methods can be useful in this regard by providing empirical probability estimates at each point of failure simply by recording and analyzing monitor verdicts over time. In turn, this can be used to quantitatively specify system risk, increase the overall trustworthiness of deep learning systems, and inform judicial decisions in the event that these failures incur significant consequences. Given a sufficiently rigorous methodology, these probability estimates may also constitute sufficient evidence for a given system's overall performance and thus assure regulatory compliance and inform health and safety evaluations.

Overall, we conjecture that the development of RV methods for multimedia deep learning systems is an area of significant potential with respect to further research. Continued efforts to this end may, in time, facilitate the operationalization of such systems and ameliorate many of the shortcomings preventing their application in production domains.

6 CONCLUSION

In this paper, we have proposed an RV framework for deep learning systems. We have reviewed recent work in the field on how deep learning systems readily fail in practical deployment and how existing methods for testing these systems generally fall short. Implementing deep learning systems with RV methods can, however, improve the system's trustworthiness, transparency, fault-tolerance and overall utility, as evidenced by the recent successes with regard to the development of methods of detecting distributional shift. By building monitors for each system requirement and at each critical data path, system failures can be detected and treated, improving the overall trustworthiness of the system. We envision that runtime monitors may, in time, become necessary components of deep learning systems, similar to how they are critical components of safety-critical or otherwise high-stakes systems in other fields. Continued research and development of RV methods may, over time, ease the difficulty of regulatory compliance, improve system stability and overall performance, and reduce the risks involved with the real-world deployment of deep learning systems.

REFERENCES

Achintalwar, S., Garcia, A. A., Anaby-Tavor, A., Baldini, I., Berger, S. E., Bhattacharjee, B., Bouneffouf, D., Chaudhury, S., Chen, P.-Y., Chiazor, L., Daly, E. M., de Paula, R. A., Dognin, P., Farchi, E., Ghosh, S., Hind, M., Horesh, R., Kour, G., Lee, J. Y., Miehling, E., Murugesan, K., Nagireddy, M., Padhi, I., Piorkowski, D., Rawat, A., Raz, O., Sattigeri, P., Strobelt, H., Swaminathan, S., Tillmann, C., Trivedi, A., Varshney, K. R., Wei, D., Witherspooon, S., and Zalmanovici, M. (2024). Detectors for safe and reliable llms: Implementations, uses, and limitations.

Adadi, A. and Berrada, M. (2018). Peeking inside the black-box: A survey on explainable artificial intelligence (xai). *IEEE Access*, 6.

Ali, S., Ghatwary, N., Jha, D., Isik-Polat, E., Polat, G., Yang, C., Li, W., Galdran, A., Ballester, M.-A. G., Thambawita, V., Hicks, S., Poudel, S., Lee, S.-W., Jin, Z., Gan, T., Yu, C., Yan, J., Yeo, D., Lee, H., Tomar, N. K., Haithmi, M., Ahmed, A., Riegler, M. A., Daul, C., Halvorsen, P., Rittscher, J., Salem, O. E., Lamarque, D., Cannizzaro, R., Realdon, S., de Lange, T., and East, J. E. (2022). Assessing generalisability of deep learning-based polyp detection and segmentation methods through a computer vision challenge.

Alon, G. and Kamfonas, M. (2023). Detecting language model attacks with perplexity.

Azamfirei, R., Kudchadkar, S. R., and Fackler, J. (2023). Large language models and the perils of their hallucinations. *Critical Care*, 27(1):120.

Baltrušaitis, T., Ahuja, C., and Morency, L.-P. (2017). Multimodal machine learning: A survey and taxonomy.

Berglund, L., Tong, M., Kaufmann, M., Balesni, M., Stickland, A. C., Korbak, T., and Evans, O. (2023). The reversal curse: Llms trained on "a is b" fail to learn "b is a".

Biggio, B., Corona, I., Maiorca, D., Nelson, B., Šrndić, N., Laskov, P., Giacinto, G., and Roli, F. (2013). Evasion attacks against machine learning at test time. *Lecture Notes in Computer Science*, page 387–402.

Blumenthal, R. (2024). Is copyright law the new turing test? *SIGCAS Comput. Soc.*, 52(3):10–11.

Brahma, P. P., Wu, D., and She, Y. (2016). Why deep learning works: A manifold disentanglement perspective. *IEEE Transactions on Neural Networks and Learning Systems*, 27(10).

Carlini, N. and Wagner, D. (2018). Audio adversarial examples: Targeted attacks on speech-to-text.

Chen, Z., Zhang, J. M., Hort, M., Harman, M., and Sarro, F. (2024). Fairness testing: A comprehensive survey and analysis of trends. *ACM Trans. Softw. Eng. Methodol.* Just Accepted.

Chu, Z., Wang, Z., and Zhang, W. (2024). Fairness in large language models: A taxonomic survey.

D'Amour, A., Heller, K., Moldovan, D., Adlam, B., Alipanahi, B., Beutel, A., Chen, C., Deaton, J., Eisenstein, J., Hoffman, M. D., Hormozdiari, F., Houlsby, N., Hou, S., Jerfel, G., Karthikesalingam, A., Lucic, M., Ma, Y., McLean, C., Mincu, D., Mitani, A., Montanari, A., Nado, Z., Natarajan, V., Nielson, C., Osborne, T. F., Raman, R., Ramasamy, K., Sayres, R., Schrouff, J., Seneviratne, M., Sequeira, S., Suresh, H., Veitch, V., Vladymyrov, M., Wang, X., Webster, K., Yadlowsky, S., Yun, T., Zhai, X., and Sculley, D. (2020). Underspecification presents challenges for credibility in modern machine learning.

Falcone, Y., Havelund, K., and Reger, G. (2013). A tutorial on runtime verification. *Engineering dependable software systems*, pages 141–175.

Falcone, Y., Krstić, S., Reger, G., and Traytel, D. (2021). A taxonomy for classifying runtime verification tools. *International Journal on Software Tools for Technology Transfer*, 23(2).

Francalanza, A., Pérez, J. A., and Sánchez, C. (2018). *Runtime Verification for Decentralised and Distributed Systems*, pages 176–210. Springer International Publishing, Cham.

Geirhos, R., Jacobsen, J.-H., Michaelis, C., Zemel, R., Brendel, W., Bethge, M., and Wichmann, F. A. (2020). Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673.

Ghazarian, A., Zheng, J., Struppa, D., and Rakovski, C. (2022). Assessing the reidentification risks posed by deep learning algorithms applied to ecg data. *IEEE Access*, 10.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Harder, P., Pfreundt, F.-J., Keuper, M., and Keuper, J. (2021). Spectraldefense: Detecting adversarial attacks on cnns in the fourier domain. In *2021 International Joint Conference on Neural Networks (IJCNN)*.

Hendrycks, D. and Dietterich, T. (2019). Benchmarking neural network robustness to common corruptions and perturbations.

Heo, J., Joo, S., and Moon, T. (2019). Fooling neural network interpretations via adversarial model manipulation. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R., editors, *NeurIPS*, volume 32. Curran Associates, Inc.

Holstein, K., Wortman Vaughan, J., Daumé, H., Dudik, M., and Wallach, H. (2019). Improving fairness in machine learning systems: What do industry practitioners need? In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, page 1–16, New York, NY, USA. Association for Computing Machinery.

Hurlin, C., Pérignon, C., and Saurin, S. (2024). The fairness of credit scoring models.

Hynes, N., Sculley, D., and Terry, M. (2017). The data linter: Lightweight automated sanity checking for ml data sets.

Inan, H., Upasani, K., Chi, J., Rungta, R., Iyer, K., Mao, Y., Tontchev, M., Hu, Q., Fuller, B., Testuggine, D., and Khabsa, M. (2023). Llama guard: Llm-based input-output safeguard for human-ai conversations.

Kauffmann, J., Ruff, L., Montavon, G., and Müller, K.-R. (2020). The clever hans effect in anomaly detection.

Lahat, D., Adali, T., and Jutten, C. (2015). Multimodal data fusion: An overview of methods, challenges, and prospects. *Proceedings of the IEEE*, 103(9).

Li, C., Wong, C., Zhang, S., Usuyama, N., Liu, H., Yang, J., Naumann, T., Poon, H., and Gao, J. (2023). Llava-med: Training a large language-and-vision assistant for biomedicine in one day.

Li, S., Guo, J., Lou, J.-G., Fan, M., Liu, T., and Zhang, D. (2022). Testing machine learning systems in industry: an empirical study. In *Proceedings of the 44th International Conference on Software Engineering: Software Engineering in Practice*, ICSE-SEIP '22, page 263–272, New York, NY, USA. Association for Computing Machinery.

Li, T., Pang, G., Bai, X., Miao, W., and Zheng, J. (2024a). Learning transferable negative prompts for out-of-distribution detection.

Li, Y., Guo, H., Zhou, K., Zhao, W. X., and Wen, J.-R. (2024b). Images are achilles' heel of alignment: Exploiting visual vulnerabilities for jailbreaking multimodal large language models.

Lindemann, L., Qin, X., Deshmukh, J. V., and Pappas, G. J. (2023). Conformal prediction for stl runtime verification. In *ICCPS '23*, page 142–153, New York, NY, USA. Association for Computing Machinery.

Liu, X., Xie, L., Wang, Y., Zou, J., Xiong, J., Ying, Z., and Vasilakos, A. V. (2021). Privacy and security issues in deep learning: A survey. *IEEE Access*, 9.

Liu, Y., Tantithamthavorn, C., Liu, Y., and Li, L. (2024). On the reliability and explainability of language models for program generation. *ACM Transactions on Software Engineering and Methodology*.

Lwakatare, L. E., Raj, A., Crnkovic, I., Bosch, J., and Olsson, H. H. (2020). Large-scale machine learning systems in real-world industrial settings: A review of challenges and solutions. *Information and Software Technology*, 127:106368.

Ma, P., Petridis, S., and Pantic, M. (2021). Detecting adversarial attacks on audiovisual speech recognition.

Marathe, S., Nambi, A., Swaminathan, M., and Sutaria, R. (2021). Currentsense: A novel approach for fault and drift detection in environmental iot sensors. In *IoTDI*, IoTDI '21, page 93–105, New York, NY, USA. Association for Computing Machinery.

Metzen, J. H., Genewein, T., Fischer, V., and Bischoff, B. (2017). On detecting adversarial perturbations.

Mireshghallah, F., Taram, M., Vepakomma, P., Singh, A., Raskar, R., and Esmaeilzadeh, H. (2020). Privacy in deep learning: A survey.

Myers, G. J., Sandler, C., and Badgett, T. (2011). *The art of software testing*. John Wiley & Sons.

Oakden-Rayner, L., Dunnmon, J., Carneiro, G., and Ré, C. (2020). Hidden stratification causes clinically meaningful failures in machine learning for medical imaging. In *Proceedings of the ACM conference on health, inference, and learning*, pages 151–159.

Paleyes, A., Urma, R.-G., and Lawrence, N. D. (2022). Challenges in deploying machine learning: A survey of case studies. *ACM Comput. Surv.*, 55(6).

Paudice, A., Muñoz-González, L., Gyorgy, A., and Lupu, E. C. (2018). Detection of adversarial training examples in poisoning attacks through anomaly detection.

Pei, K., Cao, Y., Yang, J., and Jana, S. (2017). Deepxplore: Automated whitebox testing of deep learning systems. In *SOSP*, page 1–18, New York, NY, USA. Association for Computing Machinery.

Perez, I., Dedden, F., and Goodloe, A. (2020). Copilot 3. Technical report.

Perez-Cruz, F. and Shin, H. S. (2024). Testing the cognitive limits of large language models. BIS Bulletins 83, Bank for International Settlements.

Pike, L., Niller, S., and Wegmann, N. (2012). Runtime verification for ultra-critical systems. In *Runtime Verification*, pages 310–324, Berlin, Heidelberg. Springer Berlin Heidelberg.

Provotar, O. I., Linder, Y. M., and Veres, M. M. (2019). Unsupervised anomaly detection in time series using lstm-based autoencoders. In *2019 IEEE International Conference on Advanced Trends in Information Theory (ATIT)*.

Riccio, V., Jahangirova, G., Stocco, A., Humbatova, N., Weiss, M., and Tonella, P. (2020). Testing machine learning based systems: a systematic mapping. *Empirical Software Engineering*, 25(6).

Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5).

Schwarzschild, A., Goldblum, M., Gupta, A., Dickerson, J. P., and Goldstein, T. (2021). Just how toxic is data poisoning? a unified benchmark for backdoor and data poisoning attacks. In Meila, M. and Zhang, T., editors, *Proc. of ICML*, volume 139, pages 9389–9398. PMLR.

Song, J. and Namiot, D. (2023). A survey of the implementations of model inversion attacks. In *Distributed Computer and Communication Networks*, pages 3–16, Cham. Springer Nature Switzerland.

Song, J. and Namiot, D. (2024). On real-time model inversion attacks detection. In *DCCN*, pages 56–67, Cham. Springer Nature Switzerland.

Srivastava, M., Nushi, B., Kamar, E., Shah, S., and Horvitz, E. (2020). An empirical analysis of backward compatibility in machine learning systems. In *Proc. of ACM SIGKDD*, KDD '20, page 3272–3280, New York, NY, USA. Association for Computing Machinery.

Stamatelatos, M., Dezfuli, H., Apostolakis, G., Everline, C., Guarro, S., Mathias, D., Mosleh, A., Paulos, T., Riha, D., Smith, C., et al. (2011). Probabilistic risk assessment procedures guide for nasa managers and practitioners. Technical report.

Steinhardt, J., Koh, P. W. W., and Liang, P. S. (2017). Certified defenses for data poisoning attacks. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *NeurIPS*, volume 30. Curran Associates, Inc.

Tramèr, F. (2022). Detecting adversarial examples is (nearly) as hard as classifying them.

Wang, B., Yao, Y., Shan, S., Li, H., Viswanath, B., Zheng, H., and Zhao, B. Y. (2019). Neural cleanse: Identifying and mitigating backdoor attacks in neural networks. In *2019 IEEE Symposium on Security and Privacy (SP)*.

Wang, B., Yue, X., and Sun, H. (2023). Can ChatGPT defend its belief in truth? evaluating LLM reasoning via debate. In Bouamor, H., Pino, J., and Bali, K., editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 11865–11881, Singapore. Association for Computational Linguistics.

Williams, J., Yamagishi, J., Noe, P.-G., Botinhao, C. V., and Bonastre, J.-F. (2021). Revisiting speech content privacy.

Winkler, J., Fink, C., Toberer, F., Enk, A., Kränke, T., Hofmann-Wellenhof, R., Thomas, L., Lallas, A., Blum, A., Stolz, W., and Haenssle, H. (2019). Association between surgical skin markings in dermoscopic images and diagnostic performance of a deep learning convolutional neural network for melanoma recognition. *JAMA Dermatology*, 155.

Xu, B., Tao, C., Feng, Z., Raqui, Y., and Ranwez, S. (2021). A benchmarking on cloud based speech-to-text services for french speech and background noise effect.

Yang, J., Zhou, K., Li, Y., and Liu, Z. (2022). Generalized out-of-distribution detection: A survey.

Yao, Y., Duan, J., Xu, K., Cai, Y., Sun, Z., and Zhang, Y. (2024). A survey on large language model (llm) security and privacy: The good, the bad, and the ugly. *High-Confidence Computing*, page 100211.

Yuan, Z., Xiong, Z., Zeng, Y., Yu, N., Jia, R., Song, D., and Li, B. (2024). Rigorllm: Resilient guardrails for large language models against undesired content.

Zhang, J. M., Harman, M., Ma, L., and Liu, Y. (2022). Machine learning testing: Survey, landscapes and horizons. *IEEE Transactions on Software Engineering*, 48(1).

Zheng, H. and Paiva, A. (2021). Assessing machine learning approaches to address iot sensor drift.

Zheng, Z. and Hong, P. (2018). Robust detection of adversarial attacks by modeling the intrinsic properties of deep neural networks. In *NeurIPS*, volume 31. Curran Associates, Inc.

Zhou, J., Gandomi, A. H., Chen, F., and Holzinger, A. (2021). Evaluating the quality of machine learning explanations: A survey on methods and metrics. *Electronics*, 10(5).