Quality monitoring and assessment of deployed Deep Learning models for Network AIOps

Lixuan Yang and Dario Rossi – Huawei Technologies France SASU, first.last@huawei.com

***Abstract*—Artificial Intelligence (AI) has recently attracted a lot of attention, transitioning from research labs to a wide range of successful deployments in many fields, which is particularly true for Deep Learning (DL) techniques. Ultimately, DL models are software artifacts, that, as any software realization, need to be regularly maintained and updated: consequently, as a logical extension of the DevOps software development practices to AI-software applied to network operation and management, AIOps foresee to continuously push evolved models in production networks. While for some network use-cases DL models can be incrementally updated at relatively low cost, the more typical case is that updating deployed DL models has a significant cost that needs to be managed. It follows that, during the lifecycle of DL model deployment, it is important to assess the relative "staleness" of deployed DL models, so to prioritize update of "ageing" models. In this article, we cover the issue of quality assessment and tracking of DL models deployed for network management purposes.**

I. Introduction

A decade ago, Marc Andreessen was rightly anticipating that "software is eating the world": Artificial Intelligence (AI) software has undoubtedly the very same appetite for the next decade. In particular, spectacular successes of Deep Learning (DL) techniques have gained significant share of newspaper attention in the image classification and natural language processing (NLP) fields. Numerous DL models are already deployed in production, fueled by hardware acceleration made possible from domain-specific architectures such as Graphic Processing Units (GPUs) and more recently Tensor Processing Units (TPUs) whose design is tailored for DL-workflows. Almost every technology sector currently dreams that DL can match, if not exceed, these spectacular achievements in his own field, including of course the network domain.

Without willing to undermine the significance of DL achievements, a manifest challenge is rooted in the pace of technology evolution, which is much faster with respect to the evolution of the physical world appearance and human language. For instance, training a DL model to recognize cats requires a significant volume of labeled cat images – but the *Felis* genus, which includes the domestic cat, has not evolved in the last six million years. Similarly, human language is believed to have originated in central and southern Africa between 80,000 and 160,000 years ago with the *Homo sapiens* appearance – while the language keeps evolving, normally acculturated human beings are able to read centuries-old books in their own mother-tongue language without special training. In these slowly evolving realities, image and NLP fields managed to construct large corpuses that are crucial for training DL models, such as ImageNet (comprising tens of millions of images for tens of thousands of classes) and Common Crawl (that ammassed several hundred billions of text tokens). Overall, the spectacular DL-advances have equally benefited from ground-breaking[1] theoretic research turned into software plaforms efficiently exploiting hardware acceleration, and large-scale corpuses to learn from.

Replicating such corpus for other domains, and for networking in particular, proves challenging to say the least. To understand why this happens, it is useful to compare timescales of the physical-world we live in, with the timescale of the virtual-world where network technologies let us constantly communicate in. The success of network services and the Internet since its inception[2], is supported by fast-paced technological evolution: recall that if the widely popular Moore law postulated a exponential growth in the computing capacity[3], the lesser known but equally important Gilder law observes that *"Bandwidth grows at least three times faster than the compute power"*, i.e., faster than Moore law. This growth makes the network field a peculiar one from the viewpoint of the possibilities it opens, as well as of the challenges it brings: after the ARPAnet experiment started in the 70s, the TCP/IP protocol stack created in the 80s allowed the launch of many successful services such as the Web[4] in the 90s, with an evergrowing list since the early 2000s (P2P, Cloud, streaming, social networks) and an envisionable further acceleration due to the upcoming 5G and Internet of Things deployments.

To manage such fast-paced evolving service heterogenity, the network operation and management (O&M) community has started turning its attention to DL models to relieve and assist human operation for diverse tasks such as, e.g., dynamic resource management and troubleshooting. In a field where a significant fraction of the operations are still involving human intervention, and where such intervention are also responsible for a significant fraction of the errors, automated and autonomous DL decisions are an appealing means to immediately achieve, e.g., troubleshooting guidance or fine-grained resource management at very fast timescales, and later reach error-free (or at least self-healing) operation. However, contrarily to the image and NLP fields, large corpuses of labeled data are not prevalent – quite the opposite, the very same data format of network data is not as standard as in the case of images and text, which means that knowledge gained by DL models during training can be expected to be harder

[1] Y. Bengio, Y. LeCun and G. Hinton, 2018 ACM Turing award

[2] V. Cerf and B. Kahn, 2004 ACM Turing award

[3] For discussion on Moore law in recent times, it is useful to recall the 2017 ACM Turing award lecture by J. L. Hennesey and D. Patterson

[4] T. Berners-Lee, 2017 ACM Turing award

to transfer. Additionally, as per the just observed fast-paced evolution of the network traffic and technological landscape, it is reasonable to expected that any deployed DL model should be regularly updated to remain fit to the job it has been originally designed for, in spite of the unexpected changes of the environmental conditions where it has been deployed.

In this work we consider challenges of continuous DL model update in the network operation and management context. We start by broadly introducing the lifecycle of a DL model (§II), and later focus on techniques to assess and track the quality of deployed DL models (§III). We finally introduce two example use-cases, on the image and network fields, to illustrate DL quality assessment on relevant practical examples (§IV) and summarize the key points (§V).

II. AIOps: the broad picture

AIOps is a catchy terms recently coined by Gartner to identify trends in deployment of AI techniques in a network O&M context using an agile development methodology. We introduce the network AIOps worklflow through the synoptic illustrated in Figure 1.

A. Agile methodologies

AIOps in itself is an extension of DevOps, which applies agile methodologies to combines software development (Dev) and IT operations (Ops), aimed at shortening the systems development life cycle and provide continuous delivery with high software quality.

Whereas prior to DevOps the lifecycle of new product releases was measured in months, DevOps shorten the duration to about two weeks. In particular, DevOps works in sprint where planning and implementation phases (Dev) are followed by quick deployment in operational context (Ops): monitoring the output of the Ops phase serves as input to the next Dev phase, to iteratively improve the overall solution. Business (Biz) aspects can be additionally integrated in an agile manner by an extended BizDevOps loop: alignment with business objectives is done immediately after the Ops monitoring step, to adapt and define the next Dev phase.

B. Agile + Network + AI

AIOps is a particularization of the BizDevOps cycle to take into account specific characteristics of AI-software development. Complementary to the Dev and Ops engineering teams in classic IT and O&M scenarios, AI development requires additional skillsets, which are identified in the Data Engineering and Data Scientist roles respectively. As the teams and the set of skills grow, the interaction in the workflow becomes richer, but also more complex.

Far from providing exhaustive background on AIOps, our goal is rather focus on the *data science* skills, and in particular to identify which of the AIOps steps that are not directly owned by a data scientist, could nevertheless benefit from a data-science approach. Such steps would therefore benefit from transfer of data-science techniques and practices, that can hopefully work with as little data scientist intervention as possible.

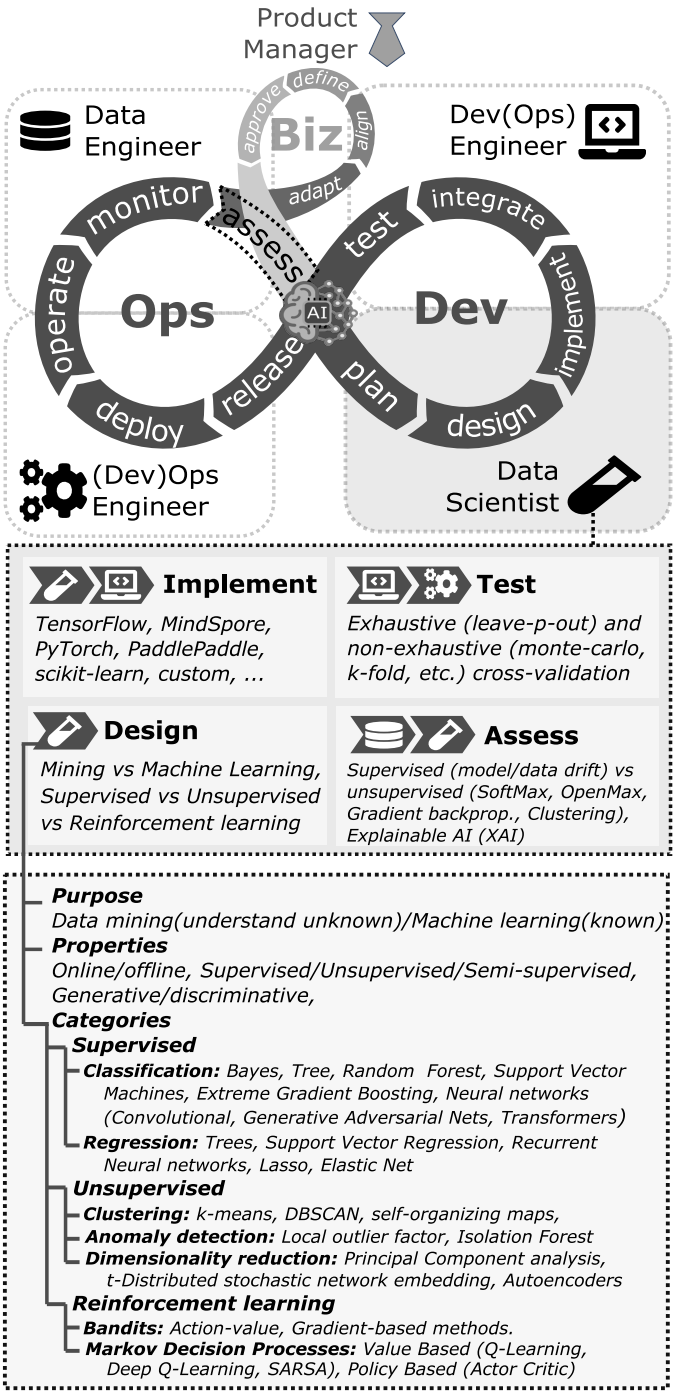


Fig. 1. *Synoptic of Network AI workflow*: (top) main actors in the BizDevOps agile method, with special focus on the AIOps roles and interactions; (bottom) high-level taxonomy of data-science tools, practices and techniques useful at different stages of the BizDevOps loop. While data scientists have full control over the *design* of the AI solution, data science methods are still useful in several steps of the workflow (e.g., *implement, test, assessment*) that are out of the hands of the data scientist (e.g., under control of developer, IT operator and data engineer respectively). Crucially, the BizDevOps and DevOps methodologies fork after the monitoring of deployed models (*adapt* vs *assess*): however, data engineer and product manager profiles lack data science skills to interpret the monitored data, hence weakening the alignment decisions. In this work, we focus on data science techniques for automated model *assessment*, empowering Biz deciders to take data-driven decisions about prioritization of data-engineering effort (e.g., labeling, data quality, etc.), ameliorating cross-team interaction and the AIOps loop effectiveness overall.

1) Design: AI design is the primary role of the Data Scientist, in accordance with business requirements as input. It is out of the scope of this article to provide a full background on the usual data scientist "weapons", whose selection depends on the problem at hand, and that broadly pertain to the data mining and machine learning fields. The bottom part of Fig.1 reports a succint (as these areas are well overviewed in the literature) taxonomy of such techniques, including but not limited to popular DL techniques. In particular, most of the succesfull and well known DL models are subject to a supervised training phase, in which the DL model is presented with relevant and abundant examples of the objects to recognize (classification) or the real-valued function to learn (regression). With respect to AIOps, understanding if and when deployed DL models need to be retrained is thus an important task.

2) Implement: Implementation and integration are the primary role of Dev Engineers. To make a smooth transition from proof-of-concept development to integration into products, there exist a well established set of DevOps tooling – from an AI perspective, it sufficient to agree on the AI software platform and allowed libraries, using best practices to simplify replicability (i.e., use of containers or well-defined library environments). Clearly, in some cases custom implementation are required to comply with further product constraints, which ultimately can lead to alter the PoC DL models.

3) Test: Testing prior to deployment is an essential tasks of Dev Engineers: as DL models, or the data pipeline, or the data itself may evolve during integration with respect to the design phase, testing guarantees the code to be ready for release and operational deployment. In this case, data scientists can contribute by sharing best practices, which are the AI-flavored version of classic unit/regression testing in the more general software industry: one such practice is the use of cross-validation techniques, specifying the way in which the training/validation/testing data should be partitioned to ensure statistical relevance of the results, avoid overfit and biases.

4) Assess: After models are deployed by Ops engineer in production system, the Data engineers *monitor* data from deployed DL models. In the BizDevOps loop, the monitored data is used to adapt, align, define and agree the next Dev phase – interposing between the Ops and Dev phases. At the same time, it is easy to observe that Data Engineer and Product managers lack the necessary data-science skills to gather profound observations from the monitored data.

The missing yet necessary link in the BizDevOps loop is represented by the *assessment* of the quality of deployed DL model. However, the deployed DL model may have been altered through re-implementation and testing and deployment data may differ from data made available at the Data Scientist in the previous iteration: as such, there is need for *automated assessment* techniques, so that the monitoring output can be automatically enriched with data-science analytics, e.g., allowing to understand if models are still fit for the environment where they have been deployed or if they have become stale. Such techniques, on which we focus on the remainder of this paper, are crucial to ensure that business alignment is taken on the ground of objective, accurate, significant and easily interpretable output.

III. Quality assessment of DL models in AIOps

A. Quality assessment challenges

Quality assessment methods need to address a number of challenges, that we now overview.

1) Label cost: Depending on the task, gathering new labels can either be a relatively simple or a very complex operation. For instance, gathering a stream of labels is possible for *forecasting* (e.g., predicting the future value based on previous inputs) and *regression* tasks (e.g., predicting the current value of an observable based on other inputs). In such cases, the current stacks supporting AIOps operations such as MLFlow[5] and MindSpore[6] are already well suited for continuous learning, combating model "drift" by seamlessly integrating data collection and model update.

Conversely, label can be costly to obtain for tasks such as *classification*: in such scenarios, a Biz decision ought to be made concerning which of the deployed models are in more urgent need for further labeling, for which a measure of models "staleness" would be of significant help. Open-set recognition (OSR) techniques developed in the data science community [1] are a good fit for this: shortly, OSR approaches are aware that DL models are trained with an incomplete view of the world (a finite set of "known" classes), and defines way to effectively deal with previously unseen classes (or "unknown" at training time) that can be submitted to an algorithm in real deployment – OSR capabilities are thus crucial in AIOps.

2) Training risk: Whereas models can be incrementally trained, the quality of the labeling process affects the model quality. For instance, the natural class imbalance present in the data can bias the models and lead to unfairness/low performance unless properly accounted for – e.g., by stratified input sampling, or by using appropriate loss functions in the DL models. Similarly, samples can be mislabeled (inadvertently, or even on purpose) which is referred to as adversarial training, and can greatly confuse the model. The aforementioned OSR techniques can assess DL model staleness with respect to a new stream of labels: roughly quantifying the magnitude of changes that models should undergo to properly account for the new labels, could be useful to both assess the retraining cost, and also assist ameliorating the quality of the labeling process (e.g., suspicious label detection).

3) Interpretability: Explainability of AI methods [2], also known as XAI, has been identified as crucial for AI application in many domains. While issues tied to fairness and ethical concerns are not predominant in the typical set of network O&M tasks, however a set of management-related aspects (such as auditability, accountability, legal and forensic matters, etc.) would greatly benefit from model interpretability. Similarly to individual model output/decisions, assessment of DL model quality should abid the same interpretability criteria, to simplify interaction with non-AI experts (e.d., Ops and Biz primarily): in turn, combining OSR and XAI for accurate and interpretable DL model quality assessment would allow to better prioritize business decisions, fill a crucially missing link of the BizDevOps loop.

[5] https://mlflow.org/
[6] https://github.com/mindspore-ai

4) Deployability: From a practical viewpoint, it is desirable for DL quality assessment techniques to work on pre-existing and unmodified models – as the need to alter models, by e.g., using a specific DL architecture, could negatively affect its adoption. Similarly, it is reasonable to assume that quality assessment techniques can access existing models, their input and outputs, but should otherwise rely on the least possible amount of meta-information about the model – for instance, access to training data may be very hard in practice. Finally, flexibility to assess quality of its individual inferences, as well as ability to track the overall model quality over time are important practical features.

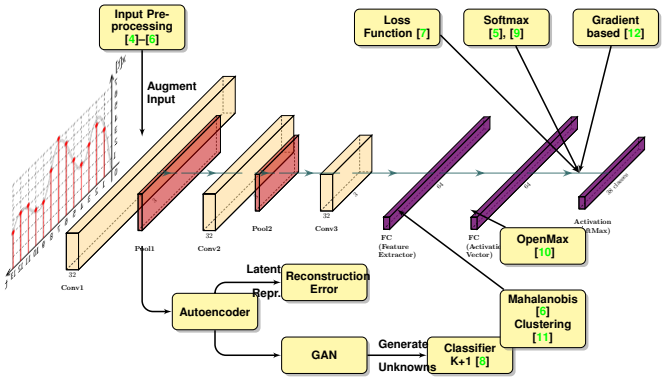
5) Efficiency and timeliness: From a computational viewpoint [3], it appears that DL model quality assessment should be a relatively simple operation. Ideally, quality assessment of a model output should have a cost comparable to the inference cost of the same model – which would allow to track the quality of deployed models at a fine grain. Similarly, quality assessment operations should be significantly much simpler with respect to model training – as otherwise, quality assessment step could be safely skipped, to directly plan a fully-fledged incremental training instead.

Finally, a related but complementary property concern the timeliness of the model quality detection: in other terms, irrespectively of the cost of individual assessment operation for a specific input/output pair, in some mission-critical cases it would be desirable to have techniques able to immediately detect model drifts from the very first few samples.

B. Quality assessment solutions

Solutions for DL quality assessment able to overcome the above challenges can be cast to the more specific problem of (i) evaluating the adequacy of a trained model to handle the typical input of the environment it is deployed to and (ii) doing that in a flexible, deployable, interpretable and efficient manner. These tasks can be achieved by building on recent work pertain to the OSR [1] and XAI [2] fields respectively. With respect to [1], Figure 2 focuses on more recent work that exploits latest advances in DL, and that tackles quality assessment at the DL input, architecture or output stages.

1) Input: A first set of work tackles the problem directly in the input space, complementing the supervised DL models building a clustering representation of the input space: the further input samples are from all clusters, the less fit is the model for this data A key challenge when using input-clustering is the creation of clusters fitting well the different classes: to better control this, instead of working directly on the original input space [4] (which is independent of the DL model and thus inherently non-explainable), some proposals apply transformations on the input so control models output, e.g. by mean of temperature-scaled [5] or Mahalanobis [6] scores (that are closer to the model and thus better interpretable). However, the main drawback of these OSR proposals is their additional computational cost tied to the evaluation of the distance of new samples from all clusters – which makes model quality assessment rather complex.



	(Ref.) Technique	Deployment	Complexity
In	[4] Input clustering	Unmodified ML/DL	High
	[5] Temperature scaling	Unmodified DL	High
	[6] Mahalanobis distance	Unmodified DL	High
Arch.	[7] Clustering loss	Modified ML/DL	Low
	[8] K+1 Classifier	Modified ML/DL	Low
Out	[9] SoftMax	Unmodified ML/DL	Low
	[10] OpenMax (AV)	Unmodified ML/DL	Medium
	[11] Clustering (FV)	Unmodified DL	High
	[12] Gradient-based	Unmodified DL	Low

Fig. 2. Overview and taxonomy of OSR techniques. Acronyms: Feature Vector (FV); Activation Vector (AV); Autoencoders (AE), Generative adversarial networks (GAN).

2) Inner: At their core, DL methods project input data into a *latent space* where is easier to separate data based on class labels: a set of work proposes specific ways to alter this latent space to purposely simplify OSR task. For instance, [7] use special clustering loss functions to constraint points of the same class to be close to each other, to simplifying the clustering task. Other work [8] instead uses Generative Adversarial Networks (GAN) to generate counterfactual samples of a fictious "unknown class". Methods in this class require specific architectures and extra training – so they might relevant for new deployments, but cannot be applied to existing ones.

3) Output: The most interesting class of OSR approaches leverages model output. The basic approach is thresholding SoftMax outputs [9], which simple and directly explainable, yet is not robust to overconfidence of the DL models. To counter this problem, OpenMax [10] revises SoftMax activation vectors adding a special "synthetic" unknown class (by using weigthing induced by Weibull modeling) while [11] performs clustering at the output of the neural network (with a PCA dimensionality reduction). In [12] we combine XAI and OSR by using a gradient-based method, where gradient backpropagation is used to detect and explain large changes in the feed-forward model due that are due to the new input, limiting the backpropagation to the last DL layer to keep computation simple. Output-based OSR approaches have the advantage of a limited complexity, of direct explainability as they readily leverage model output as opposite to surrogate models, and of not requiring modification of a pre-trained model – which makes them particularly relevant and appealing for network AIOps.

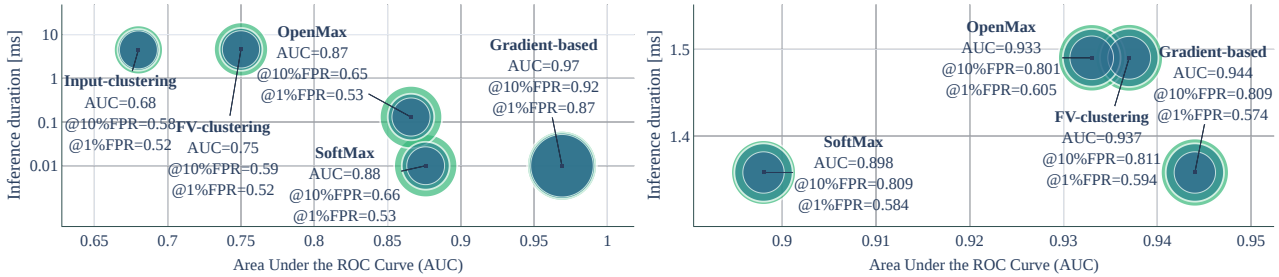


Fig. 3. *Individual inference:* Performance-complexity scatter plot of several OSR techniques for traffic classification (left) and image recognition (right).

TABLE I
USE-CASES AT A GLANCE.

	Traffic classification	**Image recognition**
Dataset	Huawei (Proprietary)	MNIST (Public)
Classes	835 applications	10 digits
Labels	4.4M labeled flows	60k labeled images
Known K	200 apps, 4.5M flows	6 digits, 42K images
Unknown U	635 apps, 37K flows	4 digits, 28K images
U/(U+K)%	76% apps, 0.8% flows	40% digits, 46% images
CNN model	3 $\text{Conv}_{3\times1}$ + FC_{128}	2 $\text{Conv}_{5\times5}$ + FC_{400} + FC_{84}

IV. EXAMPLE AIOPS USE-CASES

We illustrate how OSR building blocks can be combined to address automated quality assessment of deployed DL models considering two complementary use cases, related to *network traffic classification* and *image recognition*, that we compactly summarize and contrast in Table I.

A. Use-case definition

1) Network traffic classification: We consider the case of encrypted traffic classification, that is still relevant and active in the networking community nowadays [13]. In contrast to identification of known applications, which is a well investigated subject and for which classic supervised methods are well suited, the network community has only very limitedly [4], [14] dealt with handling applications that were never presented to the model during training, an OSR problem that is known as zero-day application detection. In particular, the current state of the art [4] performs k-means clustering on unmodified input, and is thus worth contrasting to data-science OSR solutions [9]–[12]. A complementary approach is instead taken in [14], which assumes a continuous stream of labels and incrementally trains model to combate concept drift of known classes, but is unable of zero-day traffic detection: OSR in this case would help detecting suspicious labels, or explain which of the existing classes are reponsible for the most significant model changes.

As consistently found in the state of the art, the size and direction of the first packets of a flow are well suited [13]. As baseline model, we use a 1-dimensional Convolutional Neural Network (CNN) that is consistently found in the literature to have good accuracy for the identification of known applications [12], [13], and train the model to recognize the top-200 applications. To test OSR performance, we additionally fed the trained CNN model with 37k samples from 635 applications that were never presented at training, that represent the zero-day traffic. As typical in network traffic skew, the wide majority of samples correspond to few popular applications, while a long tail of zero-day application (76%) account for only few flows (<1%).

2) Image recognition: The traffic classification use-case represents a specific evaluation very relevant for the network O&M field, but is conducted on a large commercial-grade proprietary dataset and thus not reproducible. We thus additionally consider an image-related use-case, which is less relevant for the domain expert but allows to appreciate generality of the methods, and is furthermore performed on public datasets, which enables reproducibility. In particular, the MNIST handwritten digits and characters dataset is consistently used in the OSR literature [9]–[11] as a benchmark. In this case, the problem is for any input, to assess whether is part of the handwritten digits and characters shown to the model during training, or if is an entirely new symbol.

We use LeNet [15], a classic 2d-CNN architecture often used as a classification benchmark. Notice that CNN architectural choices are qualitatively similar in both use-cases, although the hyperparametrization (e.g., filter size and depth, layer size, etc.) quantitatively differ. In this case, we train the model to recognize 6 digits, and assess OSR ability to detect the other 4 digits, with a more balanced fraction of unknown labels (40%) and samples (46%).

B. Experimental results

We assess (i) how well and at which cost OSR techniques can detect novelty on individual samples, as well as (ii) how to combine detection of novelty for individual inferences to allow quality assessment of an overall model.

1) Individual inference: At high level, the OSR output is used to detect novel input never presented to the CNN model at training: i.e, OSR invalidates an individual inference of the CNN model, overriding an erroneous known label with a zero-day label. It is however possible that the OSR technique is wrong (e.g., the app is not a zero-day) so that a correct CNN inference is wasted: the performance of the OSR methods is thus well represented by the Area under the ROC curve (AUC), that counts both correct zero-day identification as well as false positive identification (and consequent CNN inference waste).

Figure 3 reports a performance-complexity scatter plot of the different OSR techniques for both use-cases. The outer

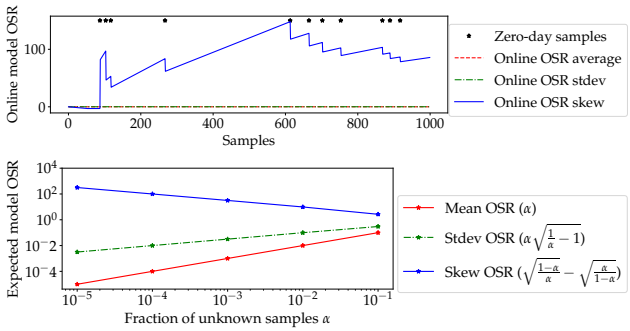


Fig. 4. *Overall model quality tracking*: online OSR time evolution (top) and expect OSR moments for different zero-day rates (bottom).

circle represent the overall AUC, whereas inner circles report the AUC value for a maximum waste (i.e., 1% and 10% false positive rate). As the 1d-traffic classification dataset is much larger (in classes) and skewed (in samples per class) than the 2d-image, the different use-cases let us constrast complementary OSR operational points.

In terms of performance, for the 1-dimensional time-series problem of the traffic classification use-case, OSR techniques are well dispersed: gradient-based [12] performs the best (significantly better than SoftMax [9] and OpenMax [10] with very similar complexity) and input-clustering [4] the worst (due to curse of dimensionality and the high number of clusters, over which output-clustering [11] only provides a very limited advantage). In the bi-dimensional image use-case, performance are in a closer range (with the exception of input clustering, not shown), but the gradient-based method still consistently shows superior performance and lower complexity.

In term of execution times for the traffic classification use-case, notice that SoftMax and Gradient techniques allow in excess of 100,000 OSR zero-day operations per second, and are thus amenable to assess the quality of the model at *each inference* – i.e., they can thus be used for exhaustive tracking. Conversely, clustering techniques[7] only allow about 100 OSR operations per second, so that they would require to *significantly down-sample* (by a factor of 1000×) inference results to be validated by OSR – in turn possibly limiting the model validation timeliness.

2) Overall model quality: We now show that by tracking OSR output of individual inferences (e.g., with SoftMax, OpenMax or Gradient methods), it is possible to robustly assess the *overall quality* of a model (e.g., by simple online average of OSR over all samples), as well as *timely track* subtle changes in the environment (e.g., from the very first unknown samples, by tracking higher moments of the OSR results). We stress that, in the traffic classification use-case, the bulk of popular applications constitute the majority of samples. As the fraction α of zero-day samples is rare ($\alpha < 1\%$), the first unknown samples are going to be presented to the DL models after several thousands of known inputs (or even more for use-cases with further class imbalance, or DL models with better class coverage).

[7]Since the number of clusters is proportional to the number of known classes, the complexity of the method is more apparent for traffic classification than for MNIST, where the number of classes is 20× smaller.

We illustrate model quality tracking in Figure 4. The top portion shows an excerpt of the zero-day arrival process for a batch of 1,000 flows, along with the unbiased online estimators of the OSR output average, standard deviation and skew: changes in the OSR skew are clearly visible at each new zero-day sample arrival (black star). The bottom picture of Figure 4 further reports the expected moments of the OSR statistics, for a fraction of unknown samples equal to α (for traffic classification, $\alpha{\approx}10^{-4}$). It can be seen that the average model-level OSR decreases as a function of the zero-day rate (more precisely equals α): as such, is suited to express *significant degradation of the model quality*. Conversely, model-level OSR skew grows for decreasing α (precisely as $\sqrt{\frac{1-\alpha}{\alpha}} - \sqrt{\frac{\alpha}{1-\alpha}}$): as such, it is well suited to especially appreciate rare events and can be leveraged as a *reliable and timely indicator of early model degradation*.

V. Conclusion

This article discusses AIOps, illustrating how data science best practices and techniques can be shared across all Dev, Ops and Biz stackholders. In particular, performed by Ops engineer to inform Biz managers in the planning of the next Dev phase, DL model quality assessment is key to improve the AIOps efficiency, and would greatly benefit from automated data science tools. We review the state of the art in techniques for quality assessment, and propose effective and flexible techniques able to cover the full spectrum of quality assessment needs: we comparatively apply these technique on two radically different use cases –namely, traffic classification, and image recognition– showing the feasibility of autonomous, accurate and timely DL model quality assessment.

References

[1] M. A. Pimentel *et al.*, "A review of novelty detection," *Signal Processing*, vol. 99, pp. 215–249, 2014.

[2] A. B. Arrieta *et al.*, "Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI," *Information Fusion*, vol. 58, pp. 82–115, 2020.

[3] Y. Huang *et al.*, "GPU: A new platform for real-time optimization in wireless networks," *IEEE Network*, vol. 34, no. 6, Nov. 2020.

[4] J. Zhang *et al.*, "Robust network traffic classification," *IEEE/ACM Transactions on Networking*, vol. 23, no. 4, pp. 1257–1270, jul 2015.

[5] S. Liang *et al.*, "Principled detection of out-of-distribution examples in neural networks," in *ICLR*, 2017.

[6] K. Lee *et al.*, "A simple unified framework for detecting out-of-distribution samples and adversarial attacks," in *NIPS*, vol. 31, 2018.

[7] M. Hassen and P. K. Chan, "Learning a neural-network-based representation for open set recognition," in *SIAM ICDM*, 2020.

[8] L. Neal *et al.*, "Open set learning with counterfactual images," in *ECCV*, 2018.

[9] D. Hendrycks and K. Gimpel, "A baseline for detecting misclassified and out-of-distribution examples in neural networks," *ICLR*, 2017.

[10] A. Bendale and T. E. Boult, "Towards open set deep networks," in *IEEE CVPR*, 2015.

[11] J. Zhang *et al.*, "Autonomous unknown-application filtering and labeling for dl-based traffic classifier update," in *IEEE INFOCOM*, 2020.

[12] L. Yang *et al.*, "Deep learning for encrypted zero-day traffic classification," *arXiv 2104.03182*, 2021.

[13] M. Shen *et al.*, "Optimizing feature selection for efficient encrypted traffic classification," *IEEE Network*, vol. 34, no. 4, July 2020.

[14] V. Carela-Español *et al.*, "A streaming flow-based technique for traffic classification applied to 12+ 1 years of internet traffic," *Telecommunication Systems*, vol. 63, no. 2, pp. 191–243, 2016.

[15] Y. LeCun *et al.*, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, pp. 541–551, 1989.