Online Safety Assurance for Learning-Augmented Systems

Noga H. Rotman
nogar02@cs.huji.ac.il
Hebrew University of Jerusalem

Michael Schapira
schapiram@cs.huji.ac.il
Hebrew University of Jerusalem

Aviv Tamar
avivt@technion.ac.il
Technion

ABSTRACT

Recently, deep learning has been successfully applied to a variety of networking problems. A fundamental challenge is that when the operational environment for a learning-augmented system differs from its training environment, such systems often make badly informed decisions, leading to bad performance. We argue that safely deploying learning-driven systems requires being able to determine, in real-time, whether system behavior is coherent, for the purpose of defaulting to a reasonable heuristic when this is not so. We term this the *online safety assurance problem* (OSAP). We present three approaches to quantifying decision uncertainty that differ in terms of the signal used to infer uncertainty. We illustrate the usefulness of online safety assurance in the context of the proposed deep reinforcement learning (RL) approach to video streaming. While deep RL for video streaming bests other approaches when the operational and training environments match, it is dominated by simple heuristics when the two differ. Our preliminary findings suggest that transitioning to a default policy when decision uncertainty is detected is key to enjoying the performance benefits afforded by leveraging ML without compromising on safety.

ACM Reference Format:
Noga H. Rotman, Michael Schapira, and Aviv Tamar. 2020. Online Safety Assurance for Learning-Augmented Systems. In *The 19th ACM Workshop on Hot Topics in Networks (HotNets '20), November 4–6, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3422604.3425940

1 INTRODUCTION

A recent trend in networking research is employing machine learning (ML) and, more specifically, deep learning (DL), to inform decision making. Example application domains include video streaming [2, 27, 49], traffic management [12, 57], resource scheduling [28], packet classification [26], caching [22], and congestion control [20, 32, 62]. In all these contexts, the ability of deep neural networks (DNNs) to pick up on complex patterns in data has proved instrumental.

Unleashing the power of DL on networking domains, however, requires care. DNNs often require many data samples to learn, and are so typically trained *offline* and then put into practice. When the training environment is faithful to the operational environment, the DNN will likely perform well. However, DL is notorious for not *generalizing* well when the two environments differ. One reason for such a mismatch is the standard practice of training on simulated/emulated environments [20, 27], which fail to capture the intricacies of real-world networks [61]. However, even if training occurs *in situ* on real data, as advocated in [61], the operational environment encountered after training might still greatly differ from the training environment due to variability in network conditions not adequately covered by the finite training data, or the introduction of new factors such as routing changes, network failures, the addition/removal of traffic sources, etc.

Can the benefits of ML-driven networking when training data is sufficiently rich be reaped without suffering the performance costs of bad generalization? We propose a general methodology for the safe deployment of DL-based systems: building into the system the means to detect, in real-time, scenarios it was not trained for and cannot make reliable decisions in. We term this challenge the *online safety assurance problem* (OSAP). Online safety assurance can accommodate the safe deployment of learning-augmented systems by facilitating switching to a reasonable default algorithm/heuristic (e.g., a widely deployed system that has already been "tested in battle" in many operational environments) when the learning-based system's decisions no longer seem sensible. In this sense, online safety assurance is intended to effectively serve as a "safety net" for DL-augmented systems.

A key question when tackling OSAP, is how to best measure decision uncertainty. To ground this question in a formal framework, we consider the standard model for sequential decision making, namely, decision making under a Markov decision process (MDP) [9]. We observe that there are three natural signals within this model that could potentially be indicative of uncertainty: uncertainty regarding the environment state, incoherent choices of actions, and incoherent perceptions of how choices of actions impact performance. We present three notions of uncertainty, corresponding to these different signals, and present a methodology for how these can be utilized for online safety assurance. To the best of our knowledge, ours is the first study to explore uncertainty signals for safe sequential decision making.

To illustrate our approach, we ground our investigation in a concrete application—online bitrate adaptation (ABR) in video streaming—a concrete DL-based ABR algorithm—Pensieve [27]—and a concrete default ABR algorithm—Buffer-Based (BB) [19]. Our preliminary evaluation results corroborate the findings in [61], showing that while Pensieve dominates BB when its training data and test data come from the same distribution, it is typically worse when the two differ, sometimes by a large margin. This motivates our approach by suggesting that incorporating into Pensieve the ability to switch to BB when decision uncertainty is detected can *simultaneously* achieve high performance when Pensieve's training and operational environments match *and* safety when the two differ. Our results for safety-assurance-enhanced variants of Pensieve

HotNets '20, November 4–6, 2020, Virtual Event, USA

ACM ISBN 978-1-4503-8145-1/20/11…$15.00
https://doi.org/10.1145/3422604.3425940

provide evidence that this is indeed the case. Furthermore, our findings reveal that the choice of uncertainty signal can dramatically affect performance.

We believe that online safety assurance can facilitate the safe adoption of DL-augmented systems in many domains, and outline exciting directions for future research.

2 ONLINE SAFETY ASSURANCE

We hypothesize that safe deployment of learning-based systems can be facilitated by detecting uncertainty in system decisions and defaulting to a safer policy when a system is outside its "comfort zone". The key question, however, is how to measure uncertainty in sequential decision making.

To ground this question in a formal framework, we consider the standard model for sequential decision making—decision making under a Markov decision process (MDP) [9]. We next describe OSAP in the MDP setting and propose several approaches to online safety assurance, which differ in terms of the signal used for detecting uncertainty.

2.1 Sequential Decision Making

We consider an *agent* that interacts with an environment. At any discrete time step $t = 0, 1, \ldots$, the agent takes *action* a_t from a set of possible actions A. Following the agent's action, the environment, which at time t is at *state* s_t, transitions to a new state s_{t+1} with probability $P(s_{t+1}|s_t, a_t)$. Environment states belong to some set of possible states S. A *reward function* r maps each state-action pair (s, a) to an associated reward $r(s, a)$. The agent's goal is to select actions that lead to high rewards. A common objective for the agent is optimizing the *γ-discounted expected return* $\mathbb{E}^{\pi}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t, a_t)\right]$, where the expectation is taken with respect to the agent's decision making strategy π, henceforth termed the *policy*. In general, π is a mapping from the agents observation history $h_t = s_0, a_0, \ldots, s_t$ to a *probability distribution* over actions. In this work, we do not limit ourselves to a particular objective for the agent, nor to a specific class of policies π.

Many algorithms for finding optimal policies for MDPs employ some form of *value function* estimation [9, 50]. The value function V^{π} maps each state s to a prediction of the future benefit to the agent from visiting this state. Formally, when considering the discounted objective, V^{π} is the expected discounted return when the agent (whose policy is π), starts at that state $V^{\pi}(s) = \mathbb{E}^{\pi}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t, a_t)\middle| s_0 = s\right]$.

The above formulation of decision making is very general and encompasses essentially all learning-based algorithms proposed for networking problems that we are aware of (including [2, 12, 20, 26–28, 61]. In particular, this formulation encompasses algorithms that reflect reinforcement learning [50] (RL), adaptive control [5], or approximate dynamic programming [9] approaches.

2.2 The OSAP Problem

During training, a learning-driven agent observes data from some distribution μ_{train}, where by data we mean the history of observed states, actions, and rewards. During testing, however, the agent might observe data from a *different* distribution μ_{test}. The reason that the training and test distributions might differ can be a consequence of various reasons: the system being trained in simulation/emulation [61], variability in real-world network conditions not adequately covered by the finite training data, or the learned decision making policy driving the agent at test time to environment states that were not explored during training.

Since the learning-based components of the agent are trained on specific data, it is only natural that the farther away the agent is from its training regime, the less accurate the implicit or explicit information on which its decisions are based is.[1] How can we identify when the agent's decisions are not reliable due to a mismatch between its training and test distributions? In OSAP, given access to the agent's training data, the goal is to devise a rule that, based on the agent's test observation history h_t, decides whether the agent's next decision $\pi(h_t)$ is reliable or not.

2.3 What to Measure?

To the best of our knowledge, a standard method for measuring decision uncertainty in sequential decision making has not yet been established. We take a step in this direction by proposing and comparing three approaches to quantifying decision uncertainty, which differ in the observed signal of uncertainty: uncertainty regarding the environment's states, the policy, and the value function. We refer to uncertainty estimation pertaining to these key MDP terms as U_S, U_{π}, and U_V, respectively.

$\mathbf{U}_S$. As the agent constantly observes the changes in the environment's state, a natural signal for uncertainty is a dissimilarity between the observed sequences of states in the training and test data. Deciding whether a test sample is too different from some training data (a.k.a., is an outlier, or out-of-distribution) is a standard (*unsupervised*) ML task termed *novelty detection* (ND) [44, 52]. U_S is a natural extension of traditional ND to sequential decision making, and amounts to deciding whether the history of observed states in test is an outlier with respect to its observations on the training distribution. Uncertainty in U_S pertains to the agent's *input*, that is, the observed environment state.

$\mathbf{U}_{\pi}$ and $\mathbf{U}_V$. Unlike U_S, U_V and U_{π} measure uncertainty with respect to the agent's *output*, i.e., its decisions. The rationale is that, ultimately, it is the output we wish to be certain of. In principle, even if the training and test distributions are different, the learned policy might still provide high performance in test. Thus, measuring uncertainty in the output could potentially prevent the agent from defaulting when this is not beneficial. Alternatively, U_V and U_{π} might potentially detect uncertainty in policy/value even when the training and test distributions are *the same* (e.g., due to suboptimality of the learning algorithm), triggering transition to a safer, better performing policy.

2.4 How to Measure?

Measuring $\mathbf{U}_S$ as novelty detection. We choose to use a classic ND method, namely, the one-class support vector machine (OC-SVM) [44]. OC-SVM enables learning a function that outputs +1 in a small region capturing most of the data points, and −1 elsewhere. See formal exposition in [44].

[1] We remark that the standard notion of *generalization* in ML, and established generalization guarantees, are only applicable when the training and test distributions are the same. When outside the training distribution, as explored here, there are no generalization guarantees [4, 8, 36].

Methods for measuring uncertainty in the output of learned functions have been developed for a variety of specific learning models [6, 15, 39, 46]. We adopt a general approach that is applicable to arbitrary learning models, including deep neural networks, and can be applied to quantify both value and policy uncertainty. Our method is based on training an *ensemble* of functions in the same training environment, and quantifying uncertainty in terms of the extent to which the outputs of the different functions are in agreement [33, 47].

Measuring U_π via agent ensembles. Consider an ensemble of i different agents trained in the same training environment, where the only difference in the training process is the initialization of the neural network variables. Intuitively, given a state s_t and an observation history h_t, if each agent is certain in the output of its individual policy π^i, it is expected that the resulting probability vectors over actions outputted by different agents be similar. This would indicate that the policy learned by each agent is not highly dependent on how the neural network was initialized. To quantify similarity between probability vectors we use the Kullback–Leibler (KL) divergence [23], as in many previous studies [45, 47]. Given probability distributions over actions $\bar{a}_1, \ldots, \bar{a}_i$ outputted by the i agents comprising the agent-ensemble, the uncertainty according to U_π is the sum of distances (in terms of KL divergence) of these outputs from the distribution over actions $\bar{a}$ obtained by taking the average over all agent outputs.

Measuring U_V via value-function ensembles. Many algorithms for finding optimal policies for MDPs employ some form of value function estimation [50]. The value function V^π in such algorithms is often represented as a neural network and is trained by observing the rewards derived from selecting actions according to the agent's policy π (on the agent's training data). Importantly, even if an agent does not explicitly estimate state values, a value function for that agent can still be trained *externally* by observing the history of states, actions, and rewards resulting from the agent-environment interaction while training.

To measure uncertainty with regards to the value function, we train i value functions on the training distribution μ_{train}. As with agent ensembles, the difference in training between different value functions is in how the neural network is initialized. The level of agreement between the outputs of the different value functions $V^{(\pi,i)}$ is used as an indication of certainty regarding the state value. Given i state values outputted by the i value functions comprising the value-function ensemble, the uncertainty according to U_V is the total sum of distances between these i values and the average value.

2.5 Setting Thresholds for Defaulting

OSAP advises the agent about its uncertainty in its current decisions. To ultimately decide whether to default to another policy or not, we must threshold the uncertainty estimate somehow. To avoid premature transitions to the default policy because of sporadic or noisy data points, our thresholding techniques (illustrated in the ABR context in Section 3), incorporate two ideas (1) considering *sequences* of data points; feeding into the OC-SVM samples in the form of the k last observed environment states, and examining the variance in U_π and U_V across the last k time steps, for some predetermined $k > 0$, and (2) only defaulting if uncertainty is detected l consecutive times, for some predetermined $l > 0$. Setting the threshold for defaulting thus involves configuring k, l, and also, for U_π and U_V (which are continuous measures, unlike the binary U_S), the bar above which the action/value is considered uncertain.

Setting the defaulting threshold involves inherent tension between optimizing performance when the training and test environments are similar and controlling the possible damage when this is not so. Even when the training and test distributions are identical, uncertainty might still occasionally arise due to differences in the specific samples drawn from this distribution in training and test. Hence, if the threshold is set to be "too low", the agent will default to another policy often even when its learned policy is most relevant. In contrast, if the threshold is "too high", the agent might stick with its learned policy even when the circumstances no longer justify this. **How the threshold is set should thus reflect the system's designer/operator desired balance between performance and risk.** Determining the defaulting threshold with respect to a specific learning-based agent, default policy, environment, and uncertainty estimation metric, is an empirical question. We revisit this challenge in the concrete example of ABR in Section 3.

A key question when evaluating safety assurance is how to fairly compare approaches that are based on different signals for uncertainty. We address this through *calibration*: in our experiments, online safety assurance with respect to U_S, U_π, and U_V is calibrated to attain *the same performance* when $\mu_{training} = \mu_{test}$, and is evaluated based on its performance on the test distribution. This is inspired by traditional ND literature, in which a standard evaluation method is to set the threshold to achieve a prescribed true positive detection rate (say, 95%), and report the true negative detection rate. Our calibration approach can be regarded as an adaptation of this common practice to sequential decision making, in which performance is quantified by achieved rewards.

3 CASE STUDY: VIDEO STREAMING

We consider the case-study of adaptive bitrate (ABR) selection in video streaming, a subject of extensive recent attention [27, 61, 63]. In ABR video streaming, videos are encoded in different resolutions (bitrates) and segmented into chunks of (roughly) equal duration. ABR algorithms at video clients determine at which bitrate to download each video segment based on their local observations about network throughput. Variability in network throughput can be detrimental to QoE, potentially driving ABR algorithms to undershoot when selecting resolutions, overshoot and suffer video rebuffering, or change the bitrate too often.

Recently, applying DL to ABR in various forms has been proposed (deep reinforcement learning in [27] and deep network throughput prediction in [61]). In our ABR case study, we use Pensieve [27] as the learned policy and Buffer-Based [19] (BB) as the default ("safe") policy.

We chose Pensieve for two reasons: (1) While Pensieve performs well when its training environment reflects the test environment [27], it suffers from bad performance when the two differ [61], undermining its practical deployability and motivating the need for online safety assurance. (2) Pensieve employs an actor-critic [29]

reinforcement learning algorithm, and so has built-in reward optimization and value estimation (obviating the need for quantifying these externally to the protocol). BB is a simple ABR strategy that performs remarkably well in practice across a variety of network conditions [61] and is thus a suitable default policy.

3.1 Evaluation Framework

Datasets. For real-world data, we use two publicly available datasets: a 3G/HSDPA mobile dataset collected in Norway [40] (we use the data generated in [27]), and a 4G/LTE mobile dataset collected in Belgium [58]. For both datasets, 70% of the data was used for training, while the remaining 30% was used for testing. Validation was done on 30% of the training set. In addition, we generated 4 synthetic datasets by sampling network throughput i.i.d. from different distributions: Gamma with shape 1 and scale 2, Gamma with shape 2 and scale 2, Logistic with $\mu = 4$ and scale 0.5, and Exponential with scale 1.

Network emulation. We build upon the experimental framework in [27], where MahiMahi [30] is used to emulate network conditions from input network traces, with a 80ms RTT between video client and server.

QoE metric. We consider the conventional linear QoE metric from previous studies [27, 63]:

$$QoE = \sum_{n=1}^{N} R_n - \mu \sum_{n=1}^{N} T_n - \sum_{n=1}^{N-1} |R_{n+1} - R_n|,$$

where N is the number of chunks in the video, R_n is the bitrate at which video chunk n was downloaded, and T_n is the rebuffering time that resulted from downloading chunk n at bitrate R_n. Intuitively, optimizing this QoE metric corresponds to maximizing the average bitrate (first term) while minimizing rebuffering time (the second term) and the jitter between different bitrates (the third term).

Video. We use the "EnvivioDash3" video from the DASH–246 JavaScript reference client [1] (the same video used in [27]). The video is encoded in six different bitrates ($\{240, 360, 480, 720, 1080, 1400\}$p), and is divided into forty-eight video chunks, each around 4-seconds long. To prolong the duration of the video we created a new video by concatenating the original video five times.

Learned and default ABR algorithms. We use Pensieve [27] and Buffer-Based (BB) [19], as implemented in [27], as the learned and default ABR policies, respectively.

Online safety assurance schemes. We generate three safety-enhanced variants of Pensieve by incorporating into Pensieve online safety assurance with respect to each of our uncertainty metrics, U_S, U_π, and U_V.

U_S is determined by a OC-SVM (see Section 2.4) implemented using the SciPy package [59]. During test, at each time step t, the mean and standard deviation of the 10 most recent network throughputs are calculated, and a sample consisting of the k latest [mean, deviation] pairs is fed into the (trained) OC-SVM model. In our experiments, $k = 5$ for the empirical distributions and $k = 30$ for the synthetic distributions (we observed that a longer window was required for good performance on the synthetic data, which we attribute to the high variance in these distributions). Given such a sample, the OC-SVM model provides a binary answer, classifying the sample as either in-distribution or out-of-distribution (OOD). When samples are classified as OOD for $l = 3$ consecutive time steps, the system defaults to BB.

To compute U_π and U_V, an ensemble (of agents and value functions, respectively) of size $i = 5$ is trained. At each time step t, the two outputs (actions/values) whose distance from the average is highest are discarded and U_π and U_V are computed with respect to the three surviving outputs as explained in Section 2.4. The system defaults to BB when the variance of this value across the last $k = 5$ time steps exceeds a certain threshold α for l consecutive times. α and l for each of the two schemes are determined as explained below.

We henceforth refer to the safety enhancements that use U_S, U_π, and U_V, as *novelty detection* (ND), *agent ensemble* (A-ensemble), and *value ensemble* (V-ensemble), respectively.

Threshold calibration. To allow for fair comparison (see Section 2.5), we calibrate the defaulting thresholds for the evaluated U_π-based and U_V-based safety assurance schemes to match the performance (QoE) of the U_S-based scheme with respect to each considered training distribution. We defer the thorough investigation of how different thresholding strategies impact performance to future research.

Remark: offline and online running times. Our safety assurance schemes require *offline* training of novelty detection schemes (for U_S, OC-SVM in our case) or RL agents and value functions (for U_π and U_V, respectively) and also *online* computation to determine whether to transition to the default policy. Training, in our experiments, was executed (offline) on 24-core Intel(R) Xeon(R) CPU E5-2630 v3 @ 2.40GHz w. Nvidia Tesla M60 GPU machines, and required less than eight seconds for OC-SVM (for U_S-based safety assurance), approximately eight hours for an RL agent (for U_π-based safety assurance), and approximately four hours for a value function (for U_V-based safety assurance). The online decision making required far fewer resources, and was therefore executed on a single core Intel(R) Core(TM) i5-8500 CPU @ 3.00GHz w. GeForce GT 730 GPU machine. Each decision required approximately 0.5ms for U_S-based safety assurance, 3ms for U_π-based safety assurance, and 4ms for U_V-based safety assurance. Since ABR decisions typically occur at the granularity of seconds, these running times are orders of magnitude lower than needed for accommodating timely decisions.

3.2 Pensieve with safety assurance still outperforms BB in-distribution

We first consider the performance of our safety-enhancements to Pensieve when *in-distribution*, i.e., when the training and test datasets come from *the same* distribution. Figure 1 plots the performance in test of "vanilla" Pensieve (with no safety assurance), ND, A-ensemble, V-ensemble, and BB, for all six (training,test) pairs of datasets that come from the same distribution. As expected, Pensieve, whose training environment here faithfully captures its test environment, outperforms BB in all experiments. Recall that the defaulting thresholds for the agent and value ensembles (denoted by A-ensemble and V-ensemble, respectively) are calibrated so as to match the performance of ND (with the fixed thresholding strategy described in Section 3.1) on the training distribution. Therefore,

when in-distribution, the three schemes always achieve the same level of performance, as seen in the figure. Observe that this level of performance is higher than that provided by BB and lower than that provided by Pensieve. As discussed in Section 2.5, the latter is a necessary price for accomplishing safety when *out-of-distribution* (OOD). Different choices of thresholds would strike different trade-offs between performance in distribution and safety when out of distribution (see Section 2.5).

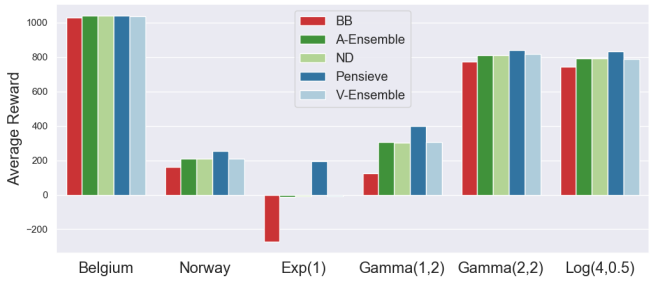


Figure 1: Pensieve with and without safety assurance *vs.* BB when the training and test distributions are the same.

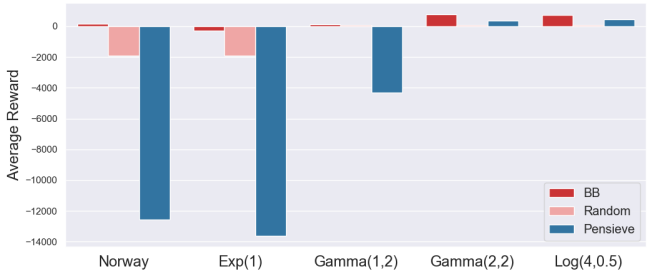
3.3 Pensieve is dominated by BB when out-of-distribution

We next show that when evaluated on a test distribution that does not match its training distribution, Pensieve can fail catastrophically, in some cases even performing worse than a naive baseline that always selects the next bitrate *uniformly at random*, termed "Random" henceforth. To illustrate this, Figure 2 contrasts the performance (QoE) achieved by Pensieve, BB, and Random for two training distributions: Belgium and Gamma(2,2). Observe that, with but one exception, Pensieve is outperformed by BB, and is sometimes even (significantly) outperformed by Random. This attests to Pensieve's inability to generalize well, as also shown empirically in [61].

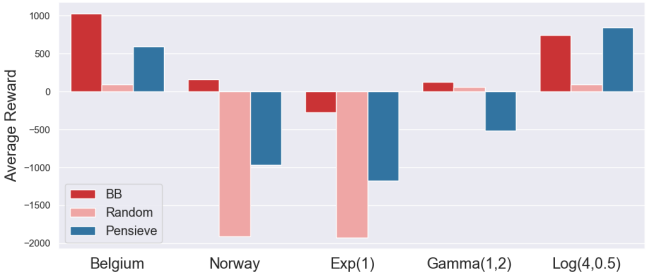
Hereafter, to clearly compare Pensieve's (with and without safety assurance) and BB's performance across different distributions, and contend with QoE scores being sometimes positive and sometimes negative, we normalize performance as follows. In all subsequent figures, a performance value of 0 corresponds to Random's performance (on the relevant dataset), whereas a performance of 1 corresponds to the gap between BB's performance and Random's performance. Figure 3 depicts Pensieve's normalized scores when trained on different distributions and evaluated on *other* distributions. When Pensieve's score is lower than 1, this indicates that it is outperformed by BB, whereas when its score is lower than 0, it is outperformed also by Random. Note that the scale on the y-axis is linear in the region [-1,1] and *log scale* elsewhere. As can be seen in Figure 3, Pensieve is typically outperformed by BB when OOD.

3.4 Contrasting the three safety assurance schemes when OOD

Figure 4 depicts the normalized maximal, minimal, mean, and median performance of the three safety-enhanced Pensieve variants



(a) Pensieve trained on Belgium and evaluated on the other datasets.



(b) Pensieve trained on Gamma(2,2) and evaluated on the other datasets.

Figure 2: Illustration of Pensieve's (problematic) generalization to other environments

Figure 3: Pensieve's performance across all datasets.

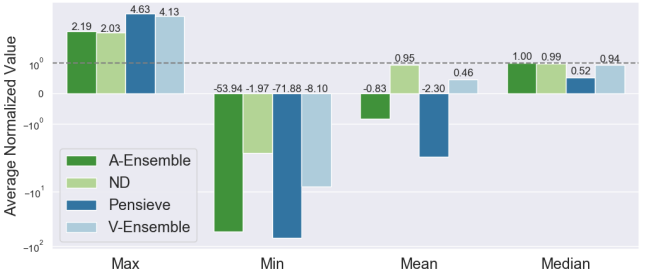


Figure 4: Comparison of different safety-enhanced variants of Pensieve OOD.

across all 30 combinations of training and test datasets that *do not* come from the same distribution. We also present the performance

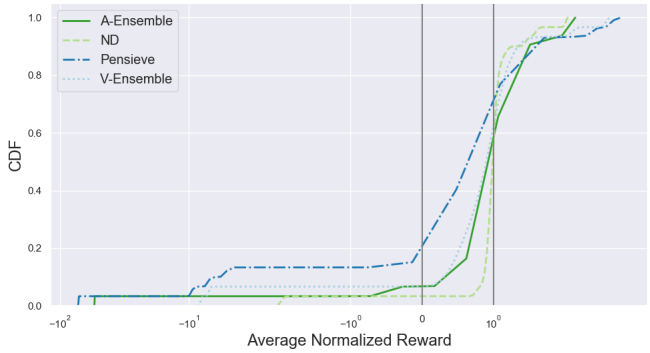


Figure 5: A CDF of performance for the different schemes across all 30 training-test combinations where test is OOD

of Pensieve (without safety assurance). Again, note the log scale outside [-1,1]. To provide more visibility into our results, Figure 5 presents a CDF of performance across our experiments for all 30 training-test dataset pairs.

Evidently, all three safety assurance schemes are better than vanilla Pensieve in terms of min, mean, and median values. Figure 4 shows that the choice between the different safety assurance schemes allows flexibility in trading off risk (min performance) for payoff (max/mean/median performance). How to balance the different factors is a choice to be made by the system designer. In the following, however, we seek general claims about the risk/payoff trade-off for each scheme.

Interestingly, **A-ensemble is essentially dominated by the other two safety assurance approaches**, with significantly lower min value, worse than Random (!) mean value, marginally better median value than the other two, and comparable (to ND) or worse (than V-ensemble) max performance. We conjecture that this is due to the fact that when training an ensemble of agents, though the training domain is the same for all, each might learn a *different* (good) policy. Consequently, variability between agent outputs might manifest *even on the training distribution*. Since we calibrate performance on training, this variability effectively leads to setting the threshold for defaulting for A-ensemble to be more tolerant to policy disagreement, and consequently failing to detect when an agent is operating OOD. This suggests that **due to the high variability in training, A-ensembles are inherently less reliable uncertainty estimators for OSAP.**

Deciding between ND and V-ensemble depends on the specific criterion; V-ensemble is better in terms of max performance, whereas ND is better in terms of the min and mean performance (the two are comparable in terms of median). Thus, **ND constitutes a safer choice, whereas V-ensemble can potentially provide higher performance gains.** To explain this, first note that **V-ensembles do not suffer from the variability of A-ensembles as they are trained with respect to *a single* agent's policy**. Second, **ND reflects the most conservative approach, as it defaults whenever OOD samples are detected, while V-ensemble may potentially yield higher rewards when choosing to stick with Pensieve on OOD samples when the value has high certainty.**

4 RELATED WORK

The mismatch between training and test distributions is the main challenge in *off-policy RL* [25]. Theoretical sample complexity guarantees were explored in [14, 16, 54–56], and recent work explores offline deep RL [13, 24, 41]. In these works, the mismatch is due to a change in policy, *not* in state transitions, as in our context. Perhaps surprisingly, generalization in RL has received relatively little attention [53], and recent studies focus on procedurally generated domains such as mazes [21, 51, 64] or platform games [31], and learning robust policies [18, 38]. None of these studies addresses the question of how to safely react to a change in the environment.

Classical methods for novelty detection include OC-SVM [44] and support vector data description [52]. Recent methods build on deep generative models [3, 11, 42, 48, 60]. *Curiosity-based exploration* in RL involves guiding the agent to unvisited states *during training* [35, 43]. Studies of curiosity-based exploration measure uncertainty in models of the environment [17, 37], in state observations [7, 10], and in the level of agreement between an ensemble of value functions or policies [33, 34, 47]. Our uncertainty estimation methods are inspired by this line of research, but strive to achieve a very different goal: *avoiding uncertainty at test time*. Consequently, our challenges and solutions are different.

5 CONCLUSION AND DIRECTIONS FOR FUTURE RESEARCH

Learning-augmented systems have received much attention of late. However, releasing deep-learning-based systems in the wild requires guaranteeing that these attain acceptable performance even when their operational environment deviates from their training environment. We proposed detecting, in real time, when the decisions reached by a system are no longer unreliable, and defaulting to a safer alternative. Our initial investigation of such online safety assurance for ABR video streaming revealed that applying novelty detection methods or value-based uncertainty estimation for this purpose is promising, whereas using action-based uncertainty as a signal fares much worse. Natural directions for future research include the further investigation of which of the two online assurance schemes, novelty-detection-based and value-uncertainty-based, is more appropriate in different contexts, and exploring the implications for the performance of different thresholding strategies. Other intriguing research directions include extending our preliminary findings for ABR by considering other DL-based ABR systems (e.g., [61]) and default policies, and investigating online safety assurance when training is performed in situ [61]. In addition, the exploration of online safety assurance in other application domains opens exciting directions for future research.

ACKNOWLEDGMENTS

Michael Schapira is partly funded by the Israel Science Foundation (ISF) and an NSF-BSF grant (BSF-2019798). Aviv Tamar is partly funded by the Israel Science Foundation (ISF-759/19) and the Open Philanthropy Project Fund, an advised fund of Silicon Valley Community Foundation.

REFERENCES

[1] Dash industry form. http://mediapm.edgesuite.net/dash/public/nightly/samples/dash-if-reference-player/index.html, 2016. Accessed: 2020.

[2] Z. Akhtar, Y. S. Nam, R. Govindan, S. Rao, J. Chen, E. Katz-Bassett, B. Ribeiro, J. Zhan, and H. Zhang. Oboe: auto-tuning video abr algorithms to network conditions. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, pages 44–58, 2018.

[3] J. An and S. Cho. Variational autoencoder based anomaly detection using reconstruction probability. *Special Lecture on IE*, 2(1), 2015.

[4] M. Arjovsky, L. Bottou, I. Gulrajani, and D. Lopez-Paz. Invariant risk minimization. *arXiv preprint arXiv:1907.02893*, 2019.

[5] K. J. Åström and B. Wittenmark. *Adaptive control*. Courier Corporation, 2013.

[6] T. Auld, A. W. Moore, and S. F. Gull. Bayesian neural networks for internet traffic classification. *IEEE Transactions on neural networks*, 18(1):223–239, 2007.

[7] M. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in neural information processing systems*, pages 1471–1479, 2016.

[8] S. Ben-David, J. Blitzer, K. Crammer, and F. Pereira. Analysis of representations for domain adaptation. In *Advances in neural information processing systems*, pages 137–144, 2007.

[9] D. P. Bertsekas. *Dynamic programming and optimal control*, volume 1. Athena scientific Belmont, MA, 1995.

[10] Y. Burda, H. Edwards, A. Storkey, and O. Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

[11] R. Chalapathy and S. Chawla. Deep learning for anomaly detection: A survey. *CoRR*, abs/1901.03407, 2019.

[12] L. Chen, J. Lingys, K. Chen, and F. Liu. Auto: Scaling deep reinforcement learning for datacenter-scale automatic traffic optimization. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, pages 191–205, 2018.

[13] S. Fujimoto, D. Meger, and D. Precup. Off-policy deep reinforcement learning without exploration. *arXiv preprint arXiv:1812.02900*, 2018.

[14] M. Ghavamzadeh, M. Petrik, and Y. Chow. Safe policy improvement by minimizing robust baseline regret. In *Advances in Neural Information Processing Systems*, pages 2298–2306, 2016.

[15] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. On calibration of modern neural networks. *arXiv preprint arXiv:1706.04599*, 2017.

[16] J. P. Hanna, P. Stone, and S. Niekum. Bootstrapping with models: Confidence intervals for off-policy evaluation. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[17] R. Houthooft, X. Chen, Y. Duan, J. Schulman, F. De Turck, and P. Abbeel. Vime: Variational information maximizing exploration. In *Advances in Neural Information Processing Systems*, pages 1109–1117, 2016.

[18] S. Huang, N. Papernot, I. Goodfellow, Y. Duan, and P. Abbeel. Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284*, 2017.

[19] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson. A buffer-based approach to rate adaptation: Evidence from a large video streaming service. In *Proceedings of the 2014 ACM conference on SIGCOMM*, pages 187–198, 2014.

[20] N. Jay, N. Rotman, B. Godfrey, M. Schapira, and A. Tamar. A deep reinforcement learning perspective on internet congestion control. In *International Conference on Machine Learning*, pages 3050–3059, 2019.

[21] A. Juliani, A. Khalifa, V.-P. Berges, J. Harper, E. Teng, H. Henry, A. Crespi, J. Togelius, and D. Lange. Obstacle tower: A generalization challenge in vision, control, and planning. *arXiv preprint arXiv:1902.01378*, 2019.

[22] V. Kirilin, A. Sundarrajan, S. Gorinsky, and R. K. Sitaraman. Rl-cache: Learning-based cache admission for content delivery. *IEEE Journal on Selected Areas in Communications*, 2020.

[23] S. Kullback and R. A. Leibler. On information and sufficiency. *The annals of mathematical statistics*, 22(1):79–86, 1951.

[24] A. Kumar, J. Fu, M. Soh, G. Tucker, and S. Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. In *Advances in Neural Information Processing Systems*, pages 11761–11771, 2019.

[25] S. Levine, A. Kumar, G. Tucker, and J. Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

[26] E. Liang, H. Zhu, X. Jin, and I. Stoica. Neural packet classification. In *Proceedings of the ACM Special Interest Group on Data Communication*, pages 256–269. 2019.

[27] H. Mao, R. Netravali, and M. Alizadeh. Neural adaptive video streaming with pensieve. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, pages 197–210, 2017.

[28] H. Mao, M. Schwarzkopf, S. B. Venkatakrishnan, Z. Meng, and M. Alizadeh. Learning scheduling algorithms for data processing clusters. In *Proceedings of the ACM Special Interest Group on Data Communication*, pages 270–288. 2019.

[29] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016.

[30] R. Netravali, A. Sivaraman, S. Das, A. Goyal, K. Winstein, J. Mickens, and H. Balakrishnan. Mahimahi: Accurate record-and-replay for http. In *2015 USENIX Annual Technical Conference (USENIX ATC 15)*, pages 417–429, 2015.

[31] A. Nichol, V. Pfau, C. Hesse, O. Klimov, and J. Schulman. Gotta learn fast: A new benchmark for generalization in rl. *arXiv preprint arXiv:1804.03720*, 2018.

[32] X. Nie, Y. Zhao, Z. Li, G. Chen, K. Sui, J. Zhang, Z. Ye, and D. Pei. Dynamic tcp initial windows and congestion control schemes through reinforcement learning. *IEEE Journal on Selected Areas in Communications*, 37(6):1231–1247, 2019.

[33] I. Osband, J. Aslanides, and A. Cassirer. Randomized prior functions for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 8617–8629, 2018.

[34] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy. Deep exploration via bootstrapped dqn. In *Advances in neural information processing systems*, pages 4026–4034, 2016.

[35] P.-Y. Oudeyer, F. Kaplan, and V. V. Hafner. Intrinsic motivation systems for autonomous mental development. *IEEE transactions on evolutionary computation*, 11(2):265–286, 2007.

[36] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[37] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 16–17, 2017.

[38] A. Pattanaik, Z. Tang, S. Liu, G. Bommannan, and G. Chowdhary. Robust deep reinforcement learning with adversarial attacks. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 2040–2042. International Foundation for Autonomous Agents and Multiagent Systems, 2018.

[39] C. E. Rasmussen. Gaussian processes in machine learning. In *Summer School on Machine Learning*, pages 63–71. Springer, 2003.

[40] H. Riiser, P. Vigmostad, C. Griwodz, and P. Halvorsen. Commute path bandwidth traces from 3g networks: analysis and applications. In *Proceedings of the 4th ACM Multimedia Systems Conference*, pages 114–118, 2013.

[41] E. Sarafian, A. Tamar, and S. Kraus. Safe policy learning from observations. *arXiv preprint arXiv:1805.07805*, 2018.

[42] T. Schlegl, P. Seeböck, S. M. Waldstein, U. Schmidt-Erfurth, and G. Langs. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In *International Conference on Information Processing in Medical Imaging*, pages 146–157. Springer, 2017.

[43] J. Schmidhuber. Formal theory of creativity, fun, and intrinsic motivation (1990–2010). *IEEE Transactions on Autonomous Mental Development*, 2(3):230–247, 2010.

[44] B. Schölkopf, J. C. Platt, J. C. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Comput.*, 13(7):1443–1471, July 2001.

[45] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897, 2015.

[46] G. A. Seber and A. J. Lee. *Linear regression analysis*, volume 329. John Wiley & Sons, 2012.

[47] R. Sekar, O. Rybkin, K. Daniilidis, P. Abbeel, D. Hafner, and D. Pathak. Planning to explore via self-supervised world models. *arXiv preprint arXiv:2005.05960*, 2020.

[48] Y. Song and Z. Ou. Learning neural random fields with inclusive auxiliary generators. *ArXiv*, abs/1806.00271, 2018.

[49] Y. Sun, X. Yin, J. Jiang, V. Sekar, F. Lin, N. Wang, T. Liu, and B. Sinopoli. Cs2p: Improving video bitrate selection and adaptation with data-driven throughput prediction. In *Proceedings of the 2016 ACM SIGCOMM Conference*, pages 272–285, 2016.

[50] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[51] A. Tamar, Y. Wu, G. Thomas, S. Levine, and P. Abbeel. Value iteration networks. In *Advances in Neural Information Processing Systems*, pages 2154–2162, 2016.

[52] D. M. J. Tax and R. P. W. Duin. Support vector data description. *Mach. Learn.*, 54(1):45–66, Jan. 2004.

[53] M. E. Taylor and P. Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(Jul):1633–1685, 2009.

[54] P. Thomas and E. Brunskill. Data-efficient off-policy policy evaluation for reinforcement learning. In *International Conference on Machine Learning*, pages 2139–2148, 2016.

[55] P. Thomas, G. Theocharous, and M. Ghavamzadeh. High confidence policy improvement. In *International Conference on Machine Learning*, pages 2380–2388, 2015.

[56] P. S. Thomas, G. Theocharous, and M. Ghavamzadeh. High-confidence off-policy evaluation. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[57] A. Valadarsky, M. Schapira, D. Shahaf, and A. Tamar. Learning to route. In *Proceedings of the 16th ACM Workshop on Hot Topics in Networks*, pages 185–191, 2017.

[58] J. van der Hooft, S. Petrangeli, T. Wauters, R. Huysegems, P. R. Alface, T. Bostoen, and F. De Turck. HTTP/2-Based Adaptive Streaming of HEVC Video Over 4G/LTE Networks. *IEEE Communications Letters*, 20(11):2177–2180, 2016.

[59] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. Jarrod Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. Carey, İ. Polat, Y. Feng, E. W. Moore, J. Vand erPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and S. . . Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[60] W. Wang, A. Wang, A. Tamar, X. Chen, and P. Abbeel. Safer classification by synthesis. *CoRR*, abs/1711.08534, 2017.

[61] F. Y. Yan, H. Ayers, C. Zhu, S. Fouladi, J. Hong, K. Zhang, P. Levis, and K. Winstein. Learning in situ: a randomized experiment in video streaming. In *17th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 20)*, pages 495–511, 2020.

[62] F. Y. Yan, J. Ma, G. Hill, D. Raghavan, R. S. Wahby, P. Levis, and K. Winstein. Pantheon: the training ground for internet congestion-control research. *Measurement at http://pantheon. stanford. edu/result/1622*, 2018.

[63] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli. A control-theoretic approach for dynamic adaptive video streaming over http. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, pages 325–338, 2015.

[64] C. Zhang, O. Vinyals, R. Munos, and S. Bengio. A study on overfitting in deep reinforcement learning. *arXiv preprint arXiv:1804.06893*, 2018.