Online Safety Assurance for Learning-Augmented Systems

Noga H. Rotman★, Michael Schapira★, Aviv Tamar✦

האוניברסיטה העברית בירושלים
THE HEBREW UNIVERSITY OF JERUSALEM

TECHNION
Israel Institute
of Technology

HotNets 2020

Deep Learning in Networking

- Deep learning (DL) is everywhere these days - even in computer networks!
  - Video streaming, traffic management, caching…
- These ML-based algorithms **do very well** when the **training** environment is **faithful** to the **operational** environment
- However, **real-world networks** may **differ greatly** from training environments, leading to a **drop in performance**
  - Even when training *in situ*
- How can we **reap the benefits** of ML **without** paying the costs for **bad generalization**?

Presenting OSAP

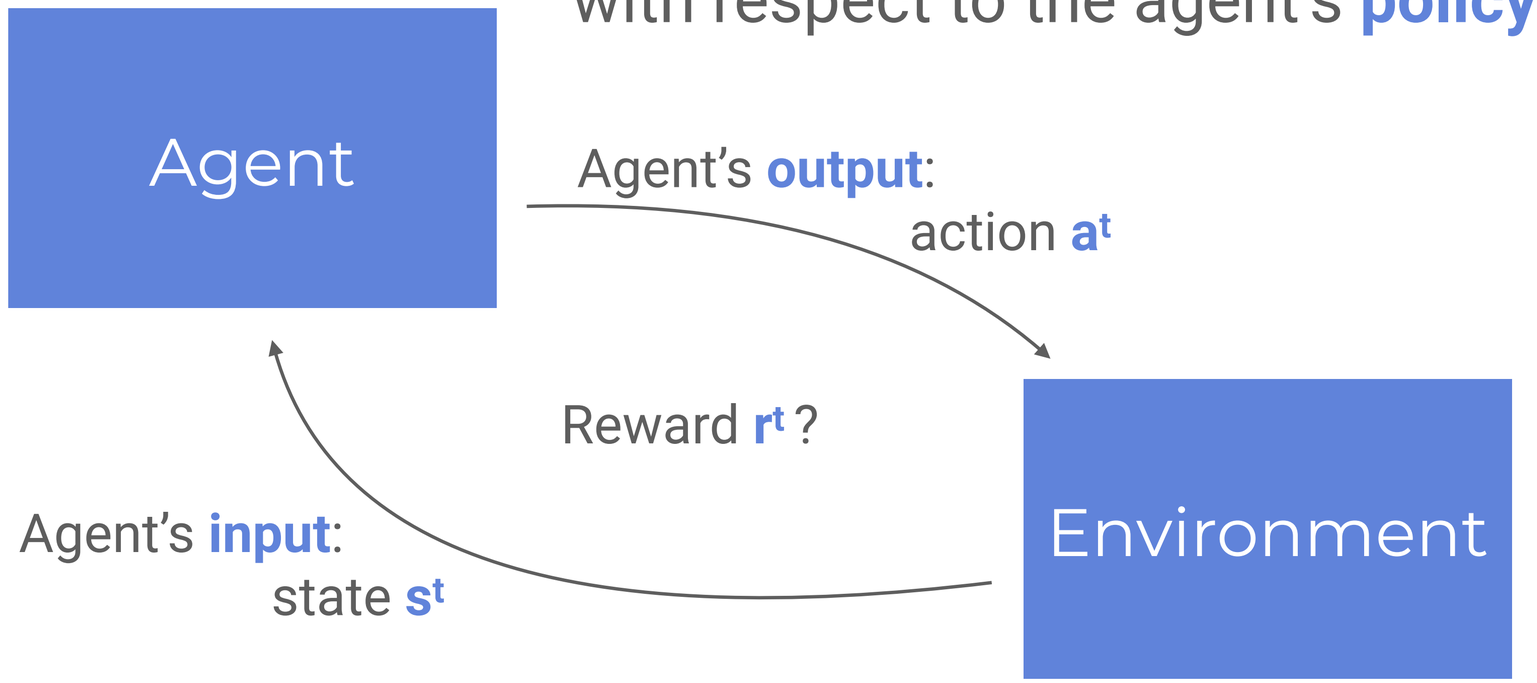
- We propose a **general methodology** for the **safe deployment** of DL-based systems
  - Building into the system the means to **detect**, in **real-time,** when the system encounters **scenarios** it was **not trained for**
- We term this challenge the **O**nline **S**afety **A**ssurance **P**roblem (OSAP)
- Online safety assurance can help by facilitating **switching** to a **reasonable**, default **non-DL** policy when the **system can’t be trusted** to make reliable decisions

Safety Assurance as a General Strategy

- Safety assurance is **fundamental** to computer and networked systems **in general**
  - A **default non-DL policy** typically **exists** - the current solution
  - Systems needs to operate in **environments too diverse** to be represented by their training data
- To the best of our knowledge, **we are the first** to tackle this issue

Sequential Decision Making (MDP)

Goal: Optimizing the **discounted expected return** $\mathbb{E}^{\pi}\left[\sum_{t=0}^{\infty}\gamma^{t}r(s_t,a_t)\right]$ with respect to the agent's **policy** π



Many algorithms employ a **value function** to estimate the **expected discounted return** at state s^t

OSAP From an ML Viewpoint

OSAP deals with a **largely unexplored** scenario in ML literature:

> What happens when a policy learned for one MDP is applied to another MDP?

Why is this the case?

- **Training** in a **simulator / emulator**, which **fails to capture real-world** dynamics
- **Changing** network **conditions**, e.g. different cell towers
- **New factors**: routing changes, network failures…

What To Measure?

Uncertainty in agent's **input**:
Identifying when the input differs from the training data, .i.e. the agent wasn't trained on these conditions

- U_S - w.r.t. the environment's **states**
- This is a standard *unsupervised* ML task termed *novelty detection* (ND)

Uncertainty in agent's **output**:
Even if the input changes, perhaps the agent learned something that is still relevant

- U_π - w.r.t. the **policy**
- U_V - w.r.t. the **value function**
- There is **no standard way** to evaluate uncertainty w.r.t. the output
- We compare the **outputs** of **ensembles**
  - several policies / value functions

How To Measure?

U_S as **novelty detection**

- OC-SVM learns a function that outputs +1 in a small region capturing most of the data points, and −1 elsewhere

Note: U_π and U_V output a continuous value - so we need to set a bar above which a sample is deemed dissimilar

U_π as **agent ensembles**

- If each agent is certain of its policy, the **output** of **several agents** should be **similar**
- Similarity: single value calculated using the Kullback-Leibler divergence

U_V as **value-function ensembles**

- If there are multiple but **different good policies**, different agents **actions might differ**, but the **values** would be **similar**
- Similarity: the sum of distances between these values and the average value

When To Default?

- To **avoid switching** to the default policy **prematurely**:
  - We consider a *sequence* of ***k* datapoints**
  - Defaulting only if **uncertainty** is **detected *l* consecutive times**
- Setting the thresholds involves configuring k, l, and also, for $\mathbf{U}_{\pi}$ and $\mathbf{U}_{\mathbf{V}}$, the bar above which the action/value are considered uncertain
- There is a **tension** between **optimizing performance** when the training and test environments are similar, and **controlling** the **possible damage** otherwise

Therefore, the choice of **threshold** is **flexible**, and should **reflect** the **system's desired balance** between performance and risk

Case Study: Adaptive Video Streaming (ABR)

- Each video split into chunks
- Each stored in different discrete bitrates
  - e.g. 240P, 480P, 720P (HD)...

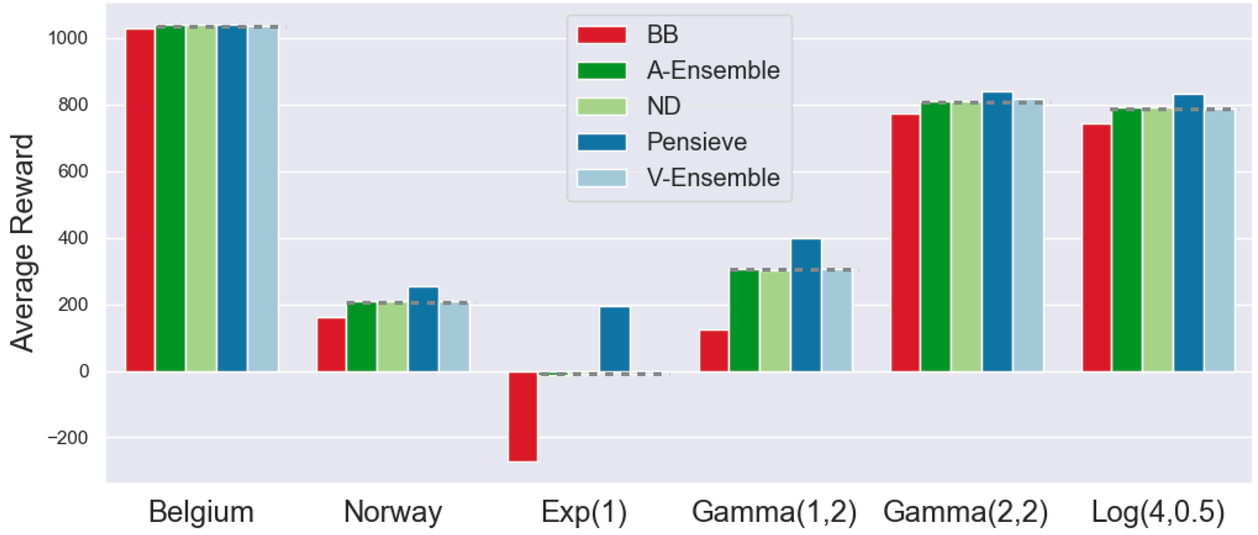
- The ABR algorithm needs to **choose** the next **bitrate** *r*
  - **Undershoot** - **bad resolution**
  - **Overshoot** - suffer **rebuffering**

Evaluation Framework

- We used **Pensieve**, a DL-based ABR algorithm as our **learned policy**
- We **implemented** the **three** proposed **safety assurance schemes** as variants of Pensieve
- As the **default policy** we chose a **simple-yet-effective** ABR strategy called Buffer-Based (**BB**)
- We evaluated using both **synthetic** and **real-world** datasets

Performance With Safety Assurance in-Distribution

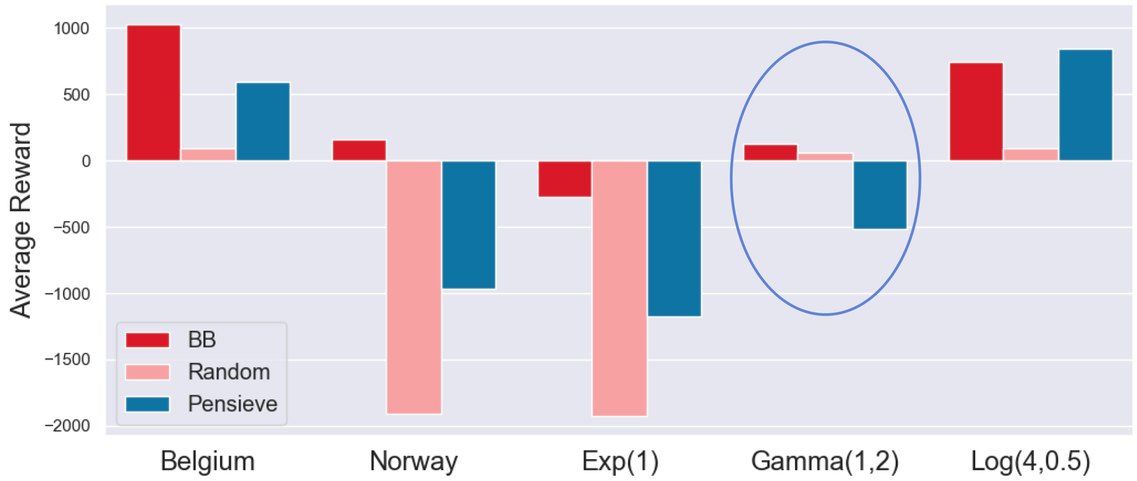
- **Pensieve outperforms BB** in-distribution, meaning, **when the training and test datasets** come from the **same distribution**
- Our **safety assurance schemes** perform **better** than **BB**
- But not as good as Pensieve
  - a **necessary cost** needed to accomplish **safety**



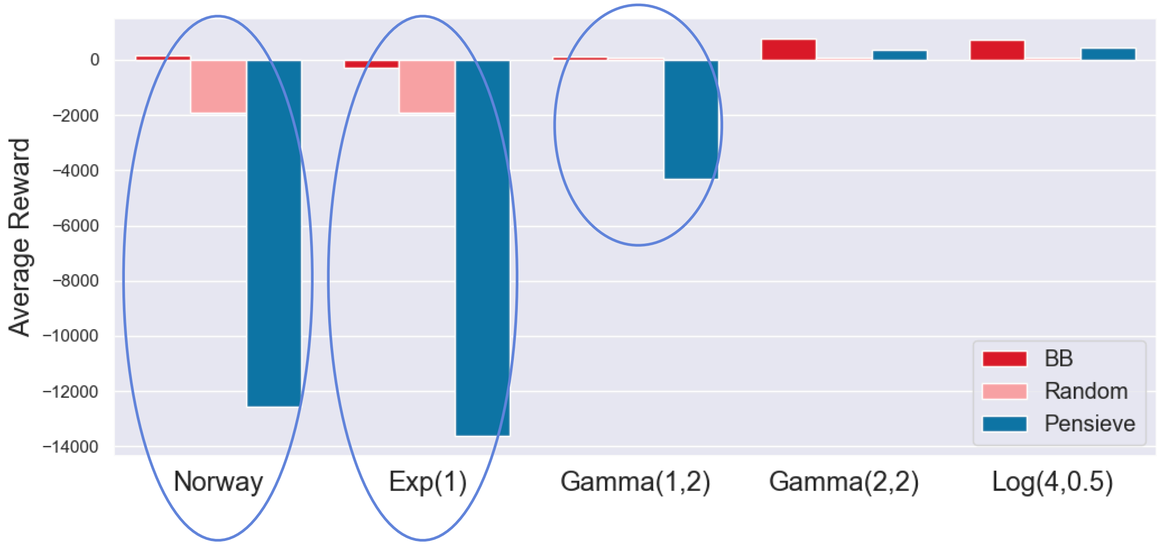
U_π and U_V are calibrated to match the performance of U_S, to allow for a fair comparison

Pensieve Is Dominated by BB When OOD

Agent trained on Gamma(2,2)



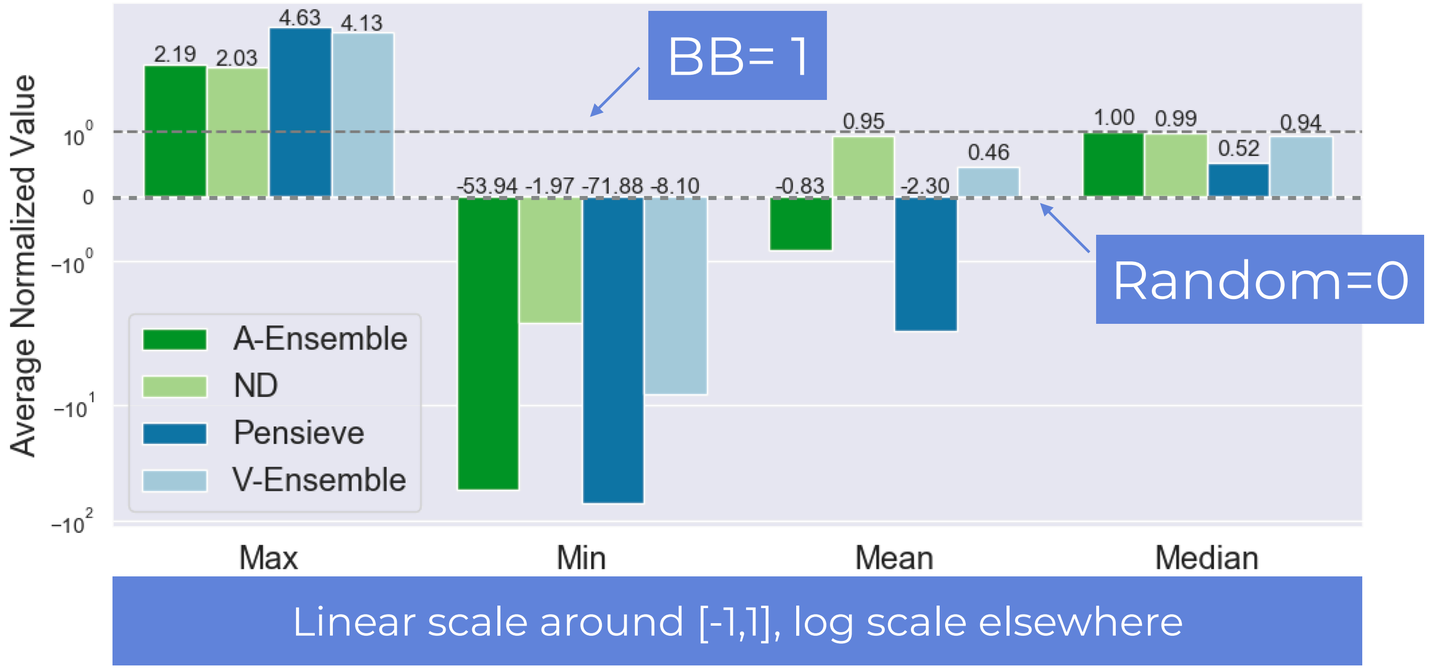
Agent trained on Belgium



Pensive does not generalize well, so when tested on **distributions different from the training distribution** (OOD), it may **performs poorly**, sometimes **worse** than **random**!

Contrasting the Three Schemes When OOD

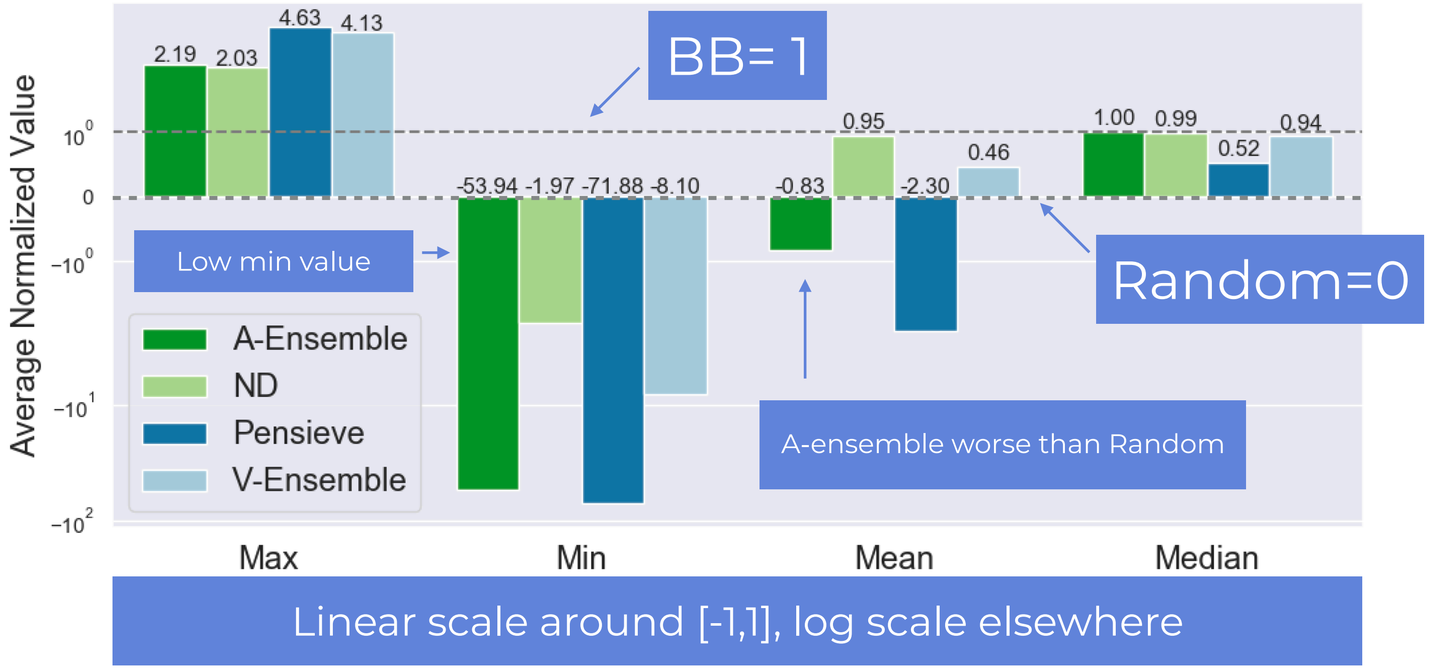
- **All three** safety assurance **schemes** are **better** than Pensieve in terms of **min**, **mean**, and **median** values
- **ND** is a **safer choice**
  - Defaulting if the agent was not trained on the input
- **V-ensemble** can **potentially** provide **higher performance**
  - **Utilizing** on the **knowledge** gained during the training even when **OOD**



Remember, the threshold is flexible, so can be chosen to obtain safer/riskier results

A-Ensembles Vs. V-Ensembles

- Interestingly, **A-ensemble is** essentially **dominated** by the **other** two **safety assurance schemes**
  - Note the significantly lower min value and worse than Random (!) mean value
- We conjecture that this is because **each agent** might **learn** a **different** (good) policy, causing the **high variability** between agent outputs **even** on the **training data**



Conclusion and Future Research

- We proposed **detecting**, in **real time**, when the decisions reached by a system are **no longer reliable**, and **defaulting** to a safer alternative
- Initial results show that incorporating **online safety assurance** maintains **good performance** when **in-distribution**, while also being **safe OOD**

What's next?

- Extending results to **other** DL-based **ABR systems** and **default policies**
- Investigating OSAP when **training** is performed **in situ**
- Exploring online safety assurance in **other application domain**

Thank You!

Questions?

Full text: https://doi.org/10.1145/3422604.3425940