Antifragility via Online Learning and Monitoring: an IoT Case Study

Vincenzo Scotti[†], Diego Perez-Palacin[†,‡], Valerio Brauzi[*], Vincenzo Grassi[*], Raffaela Mirandola[†]
[†]*KASTEL, Karlsruhe Institute of Technology*, Karlsruhe, Germany
vincenzo.scotti@kit.edu, raffaela.mirandola@kit.edu
[‡]*Linnaeus University*, Växjö, Sweden
diego.perez@lnu.se
[*]*University of Roma Tor Vergata*, Roma, Italy
vlrbrz@gmail.com, vincenzo.grassi@uniroma2.it

***Abstract*—The introduction of antifragility as a design paradigm for self-adaptive systems is shifting the attention beyond resiliency, which focuses on the ability of systems to recover from failures, towards the ability of such systems to learn from these experiences. Ideally, the manager of an autonomous system should be able to exploit a learning framework and adapt itself alongside the system it is monitoring to deal with unforeseen circumstances. Machine learning, in general, and reinforcement learning, in particular, offer the tools and framework to learn online from experience by extracting models from the collected data. However, learning tools alone are not sufficient: understanding when learning is required is a complementary problem. In fact, learning continuously can turn out to be resource demanding and may lead to faulty models. In this context, monitoring approaches for online learning, a framework of machine learning designed to deal with evolving environments through statistical analysis by predicting when the monitored system has changed, comes in handy. With this paper we offer our view on the problem of implementing antifragility using online learning and monitoring approaches in conjunction with reinforcement learning. Moreover, to support the proposed methodology, we provide a demonstrative implementation applied to the manager of an IoT network and we compare it with alternative learning approaches to showcase its benefits.**

***Index Terms*—Antifragility, Reinforcement Learning, Online Learning, Monitoring, Drift Detection**

I. Introduction

Modern software-intensive systems require special care, for they introduce uncertainty due to their intrinsic complexity. In this context, the Self-Adaptive System (SAS) framework offers the tools to deal with the dynamic and, often, unpredictable nature of these systems. In fact, this framework allows for modifying policies and behavior according to system changes. Key concepts of this domain are *robustness*, that is the ability of a system to withstand disturbances and shocks without altering its behavior [1], *resilience*, that is the ability to recover from disruptive events [2], and –more recently– *antifragility*, that is the ability to improve by learning from the disruptive events [3] (we provide a more detailed characterization of antifragility in Section II-A). Hereafter, we focus on building SAS systems that offer the most modern and challenging concept of them, the antifragility.

Antifragility can be achieved using different methodologies and frameworks. These solutions are not necessarily to be considered as alternatives, but can be combined one with the other. Following the original description of antifragility from N. N. Taleb's book [3], antifragile systems should stick to certain design principles: simplicity, decentralization, redundancy, modularity and optionality. A common solution to the implementation of SAS is to exploit Reinforcement Learning (RL) to train an agent working as the manager in the SAS framework [4]. To make this approach antifragile, given the *online learning* nature of many RL algorithms, the agent undergoes a process called *lifelong learning* – often referred to as *passive* or *reactive*–, where the agent is continuously updated with recent observations, which include the possibly disruptive events we want the agent to adapt to [5]. Looking at more proactive approaches than the lifelong learning, instead, *chaos engineering* is a typical method to anticipate disruptive changes by injecting faults in the system and achieve antifragility by making the system preemptively robust [5], [6].

Focusing on approaches based on RL and lifelong learning [7], despite correctly implementing antifragility, they result in a *passive learning* solution. In this paper, we argue that these solutions applying RL by continuously updating an agent are suboptimal. In fact, they prescribe to update the data-driven model underlying the RL agent, ignoring the actual status of the system. While this correctly allows to incorporate knowledge about new disruptive events and make the agent react accordingly next time the same situation is presented –thus achieving antifragility–, by continuously updating the model they expose it to the risk of catastrophic forgetting (i.e., losing older knowledge embedded in the model). Moreover, lifelong learning prescribes to update the model even in the face of known circumstances, ignoring prior knowledge and consuming resources to learn even from these known scenarios; all of this without having a clear rationale behind the decision to update.

With this work, we propose our view on applying *active learning* approaches, as opposed to the passive ones, with RL to achieve antifragility and learn to react to unforeseen and unknown scenarios harming our system [5]. In fact, despite modelling completely unknown scenarios is impossible, it is possible to: (i) understand when a change has occurred

in the managed system, (ii) whether the change led to a known or, else, unknown scenario, and (iii) react accordingly. Where with "react accordingly" we mean either resorting to previous knowledge –specific to the new state– or learning how to behave in this new, unknown, state –taking care also of memorizing the new knowledge. By learning from the new unknown situations, storing the gained knowledge and recalling it when appropriate, we build our antifragility.

In Machine Learning (ML), approaching the problems of change detection and subsequent adaption is the goal of a framework called Online Learning and Monitoring (OL&M) [8]–[10]. In fact, OL&M proposes ways to combine techniques like online algorithms for RL to update the agent's model and approaches to detect changes in the monitored system and be able to react actively to such changes by adapting the agent. While OL&M accounts for both passive and active approaches, we decided to resort to active approaches because they allow us to react based on explicit and explainable change signals and because they allow us to explicitly re-use prior knowledge rather than blindly updating our agent's model.

Starting from the definition of antifragility and exploiting the techniques coming from OL&M, we show how an active learning and adaptation approach can achieve slightly better results than passive, lifelong, learning, without wasting resources, without ignoring prior knowledge and making interpretable decision. The main contribution of this paper towards active antifragility can be summarised as follows:

- we introduce of a methodology based on OL&M techniques to achieve antifragility in SAS;
- we apply our methodology to an Internet of Things (IoT) network to showcase its practical advantages;
- we provide a publicly available replication package, including the source code of our method and instructions to replicate our experiments[1].

We divide this paper into the following sections. In Section II, we provide background knowledge on the topics of antifragility and OL&M In Section III, we illustrate how the immune system inspired our method. In Section IV, we describe our methodology to exploit RL and OL&M techniques to achieve antifragility. In Section V, we outline the experimental settings to validate our methodology in light of our research questions. In Section VI, we report and comment on the results of our experiments. In Section VII, we report about related works about the application of RL to SAS and about antifragility. In Section VIII, we highlight and comment on the possible threats to our work. Finally, in Section IX, we summarize our work and we propose possible extensions.

II. Background

Hereafter, we provide the background knowledge concerning antifragility (see Section II-A) and OL&M (see Section II-B).

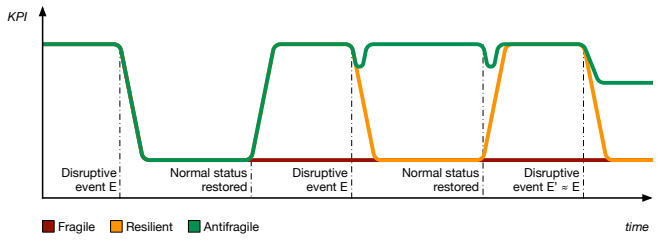

Fig. 1. Comparison of the expected behavior of fragile, resilient and antifragile systems in front of disruptive events

A. Antifragility

The notion of *antifragility* originates from N. N. Taleb's book [3], where the following often-cited definition is given: *"Antifragility is beyond resilience or robustness. The resilient resists shocks and stays the same; the antifragile gets better."*. An antifragile system then benefits from volatility, uncertainty, and change, adapting and enhancing its performance in response to such challenges. Although the antifragility framework emerged in the context of financial risk analysis, it has recently been applied across different domains, including biology, urban planning, software and Information and Communications Technology (ICT) systems.

In particular, in ICT antifragility is increasingly recognized as a desirable attribute for trustworthy, dependable systems operating in dynamic and uncertain environments. The concept has been formally integrated into the classic dependability taxonomy as a first-class attribute, highlighting the system's ability to take advantage of knowledge changes, such as new insights about threats, requirements, or the environment, and systematically improve its quality over time [11]. To get an intuition, Figure 1 provides an example of the expected behavior for Fragile, Resilient and Antifragile systems in front of unforeseen disruptive events, unforeseen known disruptive events and unforeseen partially known disruptive events.

Key characteristics for engineering antifragile systems include: (i) Continuous Learning and Adaptation – Antifragile systems monitor their environment and themselves, learning from unexpected events, errors, or attacks, and using this knowledge to improve future behavior; (ii) Change Triggering and Exploitation – Instead of avoiding all faults, antifragile systems may deliberately introduce controlled faults (e.g., via fault injection or chaos engineering) to uncover weaknesses and drive improvement; (iii) Digital Twins and Simulation –Leveraging digital twins allows safe experimentation and learning from simulated changes, supporting antifragile evolution without risking the live system [11].

B. Online learning and monitoring

OL&M is a ML framework designed to deal with *streaming* scenarios. In these scenarios, a (data-driven) model $f_\vartheta(x)$ – e.g., the RL agent serving as managing system– interacts with

[1]https://anonymous.4open.science/r/pysas-0251/README.md

an environment –e.g., the system to manage– that is *non-stationary* (i.e., the environment is *changing* or *adversarial*). The environment generates data at a given time stamp t according to a (possibly unknown) process $(x_t, y_t) \overset{\text{iid}}{\sim} \phi_t(x, y)$, which changes over time, and the model learns to predict $\hat{y}_t = f_\vartheta(x_t)$ to be as close as possible to the target variable y_t given the observation x_t. In these non-stationary settings, monitoring the interaction between the model and the environment is fundamental to react to the changes and adapt the model to the new process.

OL&M offers tools for (i) monitor the interaction between such model and the environment; (ii) learn or adapt the data-driven model. Monitoring the interaction is crucial to understand when a change has occurred (and thus when adaptation is needed). Hereafter we characterize the notions of concept and concept change (see Section II-B1 as well as the online learning approaches with specific attention to solutions for *non-stationary environments* (see Section II-B2).

1) Concept drift: In OL&M settings, we refer to the *stationary condition* of the process generating data in an environment with the term *concept*. Under a certain concept valid at time $t < T$, samples (x_t, y_t) all come from the same distribution $\phi_t(x, y) = \phi(x, y)$. Whenever at a time $t' > T$ the distribution $\phi_{t'}(x, y)$ changes so that $\phi_{t'}(x, y) = \phi'(x, y) \neq \phi(x, y)$, which is a possibility for non-stationary environments, we have a *concept drift* (or concept change) [12]. OL&M aims at identifying and reacting to these changes.

The joint distribution $\phi(x, y)$ generating the data can be decomposed as reported in Equation (1).

$$\underbrace{\phi(x,y)}_{\text{Joint distribution}} = \underbrace{\phi(y|x)}_{\text{Conditional distribution}} \cdot \underbrace{\phi(x)}_{\text{Marginal distribution}} \tag{1}$$

Concept drifts are characterized by *what* is changing and *how* the process changes over time [10]. *Real* drifts affect $\phi(y|x)$ and *virtual* drift affect $\phi(x)$. We are interested in monitoring real drifts, as the virtual ones do not affect our model.

Change (drift) detection techniques aim at understanding whether the current model is valid for current observations. It differs from *anomaly detection* (a closely related concept) because the latter aims at identifying samples not conforming to the model and the rest of the data. Typically, drift detection algorithms work using (i) a *statistic* that has a known response to normal data (e.g., average), (ii) decision rule to analyze the statistic (e.g., threshold).

Ideally, we would have a drift $\phi \to \phi'$ between two known distributions that techniques based on the likelihood analysis are the most suitable to detect. Most often, the distributions ϕ and ϕ' are unknown. In these settings it is common practice to monitor the error between the model prediction $\hat{y}_t$ and the feedback from the environment y_t.

2) Online learning: When using online learning approaches, we have a model doing predictions, possibly based on observations provided by the environment [9]. Typically the environment provides an additional reward or loss based on the difference between the predicted value and the target value without actually disclosing the target, making this framework particularly suitable for practical (real) scenarios where little information about the real process underlying the environment is known. In fact, in online learning there is no statistical characterization of the process being monitored and the is no starting information about the system.

Online learning can be seen as a meta approach. For example, some RL algorithms are for online learning (e.g., *Q-learning*) [13]. Usually, online learning algorithms aim at minimizing the *regret*: the difference between the loss (or reward) of the model we are learning and that of an oracle.

When moving to the specific case of learning online in non-stationary environments, we suppose to receive observations from a data stream for which our model produces a prediction' [8]. Based on the feedback generated from the environment (which is usually related to past predictions) we apply techniques to *adapt* the existing model. Literature identifies two alternative solutions: *active* adaptation and *passive* adaptation. The choice depend on the actual settings.

Active approaches combine the data-driven model we are learning and adapting with statistical tools to detect concept drift and pilot the adaptation exploiting recent data generated after the change. These approaches provide information on the drift occurrence, perform better than passive in stationary conditions and require adaptation only when a change is detected. Conversely, with passive approaches, the data-driven model undergoes continuous adaptation, deciding at each step which samples exploit for learning.

III. Motivation and Rationale

The proposed methodology is inspired by the human immune system, which is a classical example of an antifragile system [3]. In fact, its behavior concentrates on the following characteristics.

The immune system identifies *antigens* in the body. An antigen is any substance that can trigger an immune response. Different types of immune cells carry out the identification of antigens in the body, the *T cell* and *B cell lymphocytes*, and the Antigen Presenting Cell, which process antigens and present them to the T cell. When an antigen is found, two possibilities arise: the antigen is new to the body or already known.

The immune system triggers a *primary response* when the antigen is unknown to the body. The primary response may take several days to resolve [14]. Beyond creating antibodies and cells that face the antigen, it also creates *memory cells* (long-living memory B and T cells), leading to *immunological memory*. In turn, when the antigen is already known, the activation of the memory cells that were generated in the previous encounter with the antigen enables a rapid response. This mechanism is called *secondary response*.

Besides the primary and secondary responses, there are other parallelisms between the immune system and antifragility worth taking into account. In fact, in 2022, a global outbreak of *Mpox* began [15] caused by monkeypox virus clade II. Historical *smallpox* vaccination during childhood, which ended 40 years ago in Europe, still showed some

protection effectiveness [16]. From the antifragile systems engineering viewpoint, this opens a third possibility beyond the identification of *known* and *unknown* situations; the identification of a similar one for which there is *partial* knowledge and the system improvement based on such previous knowledge. In our methodology, we implement this aspect with specific training initializations.

IV. Methodology

To architect an immune system-inspired antifragile framework, we resort to the conceptualization and terminology of antifragility defined for ICT systems in [11]. At the same time, following ML terminology, we introduce statistics-related concepts necessary to describe our view on antifragility.

We define our methodology for antifragility in the following settings. The system's *operational context* encompasses the real-world processes and process changes and their corresponding impact on system behavior. The system *status* refers to the observation generated by the currently active process and the system's response(s) to that observation. The system's *knowledge* represents its current understanding of both the operational context and its status, this includes information about known processes, their effects or occurrences, and the system's behavioral adaptations in response.

The *requirements* specify that the system must be capable of identifying changes and, when appropriate, adjusting its behaviour accordingly. Modifications to the system's requirements themselves are considered beyond the scope of this work. The identification of a process change for which the system lacks complete prior knowledge constitutes a *knowledge update*.

Algorithm 1 describes how the OL&M tools should be used to drive the antifragile behavior. Every time a process change is detected, the new process ϕ' is compared to the currently known processes. It is the *knowledge change* given by the introduction of a new unknown process to drive the antifragile behavior. When the process is unknown the RL agent managing the system starts learning again to incorporate in a new policy the knowledge about this new process. Conversely, when the process changes to a known one, previous knowledge is restored. The key to our contribution is to use OL&M techniques to actively look for changes and trigger the adaptation accordingly.

According to the immune system antifragility described in Section III, the antrifragile behaviour triggers when a process changes $\phi \rightarrow \phi'$. If the detected process is already known ($\phi' \in K$), the system can handle this anomaly without needing to accommodate new knowledge about it, but simply resorting to the corresponding policy π'; as in the immune system *secondary response*. Conversely, when the process is not known, the agent may use only partial knowledge about the new process (if there is a similar process $\phi'' \sim \phi'$) or no knowledge at all. In the former case, the policy[2] of a similar known process is used to initialize π' before training, similar to the immune system reaction to partially known antigens. In the latter case, we simply initialize a new random policy, as it happens in the *primary response* for creating new antibodies and memory cells. Once the training process end, the new policy is associated to the new process, updating the system knowledge. The trivial case where no process change is detected is handled simply by leaving the current policy π unchanged.

[2]In this context, with the term policy we refer to any function predicting the actions to take for a given status of the system.

Algorithm 1 Antifragility OL&M loop

```
K = {}                                          ▷ Process-control mapping
φ ← random_φ()                                  ▷ Initialise current process
π ← random_π()                                  ▷ Initialise current policy
while true do                                   ▷ Loop as long as system is running
    if φ → φ' then                              ▷ Change detection
        if φ' ∈ K then                          ▷ Known process
            π' ← K[φ']                          ▷ Select known policy
        else
            if ∃ φ'' ∈ K | φ' ≈ φ'' then       ▷ Similar process
                π' ← clone(K[φ''])              ▷ Transfer learning
            else                                ▷ Unknown process
                π' ← random_π()                 ▷ Random initialization
            end if
            π' ← fit(π', φ')                    ▷ Learn policy with DT and RL
            K[φ'] ← π'                          ▷ Register new knowledge
        end if
        φ ← φ'                                  ▷ Update current process
        π ← π'                                  ▷ Update current policy
    end if
end while
```

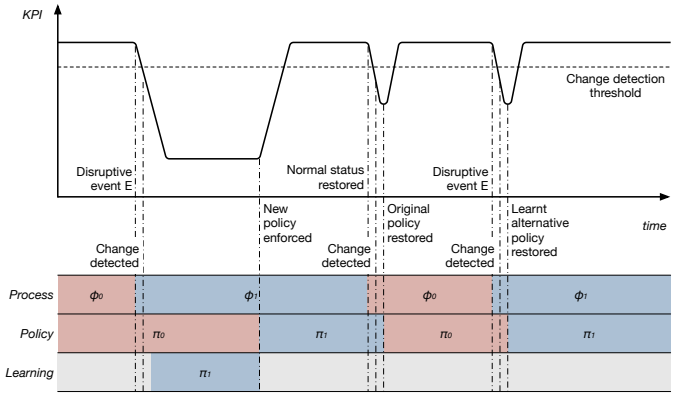

Fig. 2. Online learning and monitoring effect on KPI

Implementing the methodology described in Algorithm 1 would result in observing a behavior like that of Figure 2. While the first process change from ϕ_0 to ϕ_1 results in a failure of the known policy, leading to a significant drop in the Key Performance Indicator (KPI), the change detection system recognizes a change has occurred. Since no other known process is available, the current policy π_0 is maintained while the training of a new agent using policy π_1 is issued.

Once the agent is trained, the manager switches to π_1 and restores the KPI to an acceptable value.

At this point, when the process changes back from ϕ_1 to ϕ_0, given the changed policy we may experience a drop in performance, which is solved as soon as the previous policy π_0 is restored. At this point, a resilient system would have been simply restored to its initial status, without learning anything, while a system using a passive learning approach might have issued more than one training session (depending on the adopted schedule).

Similarly to the last step, when process ϕ_1 occurs again, the corresponding policy π_1 is restored as soon as the KPI analysis detects the change and the known process is recognized. In this case, a resilient system would have ended up in the same failed state because of the shock, while a passive system would have ignored prior knowledge and might have issued another training (again, depending on the schedule).

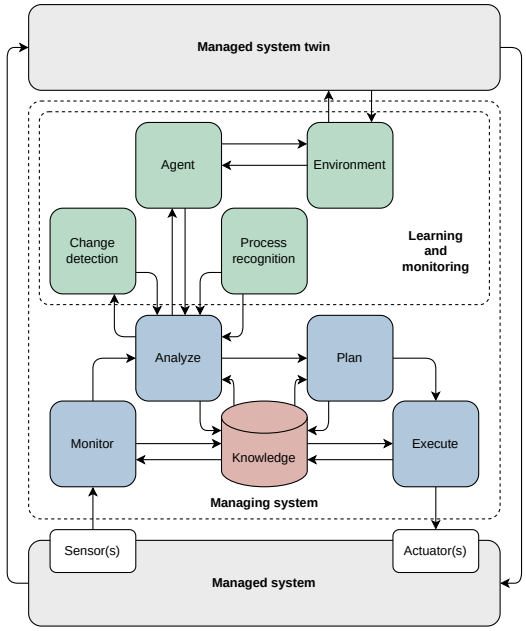

Fig. 3. MAPE-K manager combined with OL&M to achieve antifragility

To implement our methodology, we propose to extend the typical *MAPE-K* components [17] to interact with the OL&M tools necessary to antifragility. We identify the functional modules in Figure 3, where we reuse the terminology *managing system* and *managed system* SAS [18]. A digital twin of the managed system complements the architecture by providing a safe simulation and training environment for the RL agent used to manage the system. The twin is synchronized with the real system every time the parameters of the agent need to be updated.

The *monitor* of the MAPE-K loop implements the *rapid response* of the immune system, collecting observations of the system state for both the real system and the system twin. Here the twin is used as a system model to verify if the current observation is compliant with the expected one. Given the

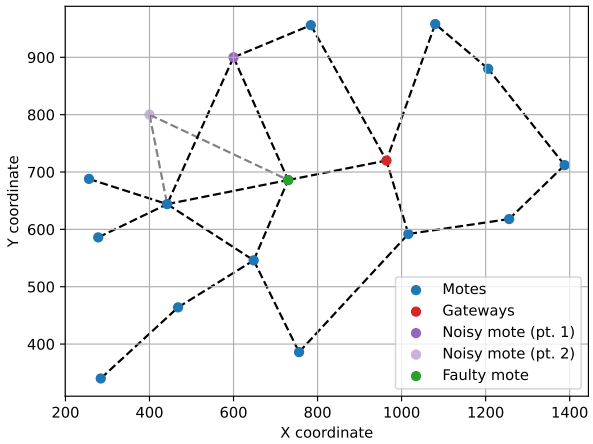

Fig. 4. DeltaIoT network topology used in the experiments

current observations, the *analyzer* of the MAPE-K loop queries the *change detection* module from the OL&M components to verify if a process change has occurred. If change is detected the current observation is processed together with the past ones by the *process recognition* module to search for known processes equivalent to the current one. If there is no known process that matches the current *primary response* of the immune system is triggered: the twin is synchronized with the new system status and the RL agent is trained simulating experience using the new process. After training, the knowledge about this new process is registered in the main *knowledge* of the MAPE-K manager (this is the *immunological memory*). Conversely, if there was no process change or if the process changed to a known one, the *secondary response* of the immune system intervenes, possibly fetching knowledge about the current process in the form of agent parameters to replace the current ones.

In our implementation we also account for similarity between processes and partial knowledge re-use. In fact, to replicate an effect similar to that of the smallpox vaccine on the Mpox virus, we consider fitting an agent on data coming from a new process that is similar to a known one, using parameters from the known process as initialization. Similarity is assessed as equivalence from the process recognition module by the analyzer.

V. Experiments

In this section we report the experimental setup (see Section V-A), the implementation details (see Section V-B), and the research questions we followed (see Section V-C) to validate our methodology.

A. Setup

We conducted our experiments using the *DeltaIoT* exemplar [19], [20]. The exemplar contains the code to operate remotely the original network it was inspired from, or simulate that same network. In all our experiments, we interact only with the network simulator. The original network system,

visualized in Figure 4, is composed of 14 motes and one gateway linked one to the other; the role of the managing system is to operate on the links to reduce energy consumption while ensuring all packets are eventually delivered.

For the scope of this paper we considered two use cases: (i) noisy mote injection and (ii) faulty mote injection. In the former use case we simulate the appearance of a new mote connected to three of the existing motes in the network and generating high traffic (purple mote in Figure 4), then we detach the mote from the network restoring the original topology, and, finally, we re-introduce that same mote with slightly different position and connections. The idea of this use case is to simulate an attempt to flood the IoT network. In the latter use case we simulate the disappearance of one of the original network motes (green mote in Figure 4), choosing one the central motes with multiple connections, and then we re-introduce the mote after some time restoring the original topology. The idea of this use case is to simulate a fault that causes a spike in the packet loss.

We evaluate the behavior of managers implementing different strategies for the adaptation. We propose three RL-based managers adopting the following learning strategies:

1) *active learning* (building an antifragile system using OL&M – our proposed solution);
2) *passive learning* (building an antifragile system using lifelong learning, like in [7]);
3) *static learning* (building a resilient system).

Additionally, we use as baselines the following:

1) a manager acting only as an *observer* (thus taking no action);
2) the original *expert-defined* manager coming with the DelataIoT exemplars.

All managers were allowed to operate only on links of the original network topology by 1) changing the frequency a packet is forwarded on one of the links departing from a given mote (increase, decrease, leave unchanged). 2) changing the power over the link (increase, decrease, leave unchanged).

We compare the results of the different managers looking at the following Quality of Service (QoS) metrics:

- cumulative energy consumption of all motes (the lower the better);
- average packet loss (the lower the better);
- network fairness index (the higher the better) [21].

We computed the fairness index starting from the measures of residual battery and queue size of the network motes, we used the variance of those metrics at each step as the weight in the index computation.

B. Implementation

We implemented and trained the managing system as a RL agent using *Q-learning* [13], [22] with the Q-network (i.e., the neural network underlying the agent) being a collection of linear models. At inference time, we use the values predicted by the Q-network for each action to build our policy and select the actions to take over the system. For each controllable link, we created two linear models, one predicting a vector with the output of the action-value function obtained (we compute the values for increasing, decreasing or leaving as is the forwarding frequency over a link) and the other predicting the output of the action-value function for the same actions (increase, decrease and leave unchanged) applied to the power of that link. The total action value function is computed as the sum of the elements if the output vectors of the single neural networks corresponding to the selected actions. Given the linear nature of the model, we encode the status of the DeltaIoT network by concatenating the features describing motes, gateways and links as well as the global network features is a single vector (we zeroed out the features of missing elements and we ignored those of elements introduced in the network after training). We take care of standardizing and normalizing the features. At this stage, we selected linear models rather than complex deep neural network models because our focus was on the active approach to antifragility instead of technical RL details.

Each time a process change is issued (including the first time at the start of the monitoring period), we train the RL agents using the Q-learning algorithm for 20 epochs and for 200 iterations per epoch. The Q-networks are trained to maximize the reward $r = 1 - e$, where e is the cumulative energy consumption normalized by its maximum value. At the start of each epoch, we collect experience using the current Q-network from an entire day of simulation of the DeltaIoT network (we reset the simulation before each training step). We store the collected experience in a *replay buffer* queue with the capacity of four days of simulation of the DeltaIoT network. We train the network using *Adam* as optimizer [23], with a learning rate $\eta = 10^{-3}$, a maximum gradient norm of 1, and a discount factor $\gamma = {}^{19}/_{20}$ that we decrease linearly to ${}^{1}/_{20}$. We stop the training if there is no improvement in the loss above 10^{-1} for more than 10 training steps (we use a warm-up period of 100 steps at the beginning of training before checking for the improvement). The change detection process for our active adaptation strategy, part of the antifragile OL&M-based methodology we propose, was implemented by monitoring the absolute difference between the QoS metrics measured by the managed network and the same values measured by applying the same actions to the digital twin. In practice, the twin serves as a predictive model of the actual DeltaIoT network, and the prediction error serves as KPI to understand when to issue a change detection in the process generating the system observations and trigger the adaptation. We use the Early Drift Detection Method (EDDM) monitoring scheme on the error to check for process changes [24]. We set the change detection threshold to $\gamma_{CD} = 3\sigma$, with σ being the standard deviation of the expected prediction error), and we issue for a change whenever the Exponentially Weighted Moving Average (EWMA) of the error ($\alpha = 10^{-1}$) grows above γ_{CD}. Moreover, we use a secondary threshold ${}^{\gamma_{CD}}/_{2}$ to issue a warning for change detection. In case a change is detected, all samples between the start of the latest warning state and the detection moment are considered samples coming from the new process.

The process recognition, another fundamental module of our antifragile OL&M-based methodology, was implemented by looking at the similarity between states. Whenever a change detection is issued, all the previous states are vectorized (standardization and normalization are done with the statistics of the current process), and the vectors with the states attributed to the same process are averaged into cluster centroids (each cluster represents a process). To attribute the current state to one of the previous processes, we compute the cosine similarity between the vector of the current state and that of the centroids. If the highest cosine similarity is $\geq {}^{2}/_{3}$, we consider the corresponding process the same of the current and we restore the corresponding Q-network –and, thus, the corresponding policy–; in this way, we manage to re-use previous knowledge as is. If the highest cosine similarity is $\geq {}^{1}/_{2}$, we consider the processes similar and we apply transfer learning from the corresponding Q-network before starting the training, re-using the partial knowledge available. If no centroid matches the similarity threshold, we simply train a new agent from scratch and update our knowledge at the end.

The antifragile manager realizing the passive learning strategy (lifelong learning) was implemented simply issuing an agent update every four hours (which correspond to 16 simulation steps of the DeltaIoT network). Each time we applied transfer learning we restarted the training from the most recent weights of the Q-network. The resilient manager realizing the static learning strategy was implemented training the agent only once at the beginning of the simulation.

All RL agents were trained cloning the current state of the simulation (motes, links and gateways) and resetting the components to an initial state. Moreover, to simulate a more realistic scenario, rather than reading directly the mote activation probabilities, we resorted to the values estimated in the last n simulation steps (we set $n = 8$; looking empirically at the running averages of the motes' activations is an exploratory analysis of the network). As a consequence, we delayed the training in case of change detection by at least n steps after the first warning and we waited a warm-up period of n steps before the training of any of the RL agents the first time. We use the same observations to fit the standardization and normalization transformations we apply to the input features and to the reward we want to maximize. We preferred this approach rather than doing some offline preprocessing to showcase how the proposed solutions can work completely online, despite some delays to collect sufficient data.

C. Research questions

With our experiments we aim at answering to the following research questions.

Research Question 1 (RQ1)

How good are the RL agents learning to operate the network from the training with the twin?

We compare the QoS metrics of the three agents with those of the baselines to understand (i) whether the agents are actually optimising the objective (i.e., if they are doing at least better than the base manager that is simply monitoring the network), and (ii) whether the agent are learning human expert-level control over the network (i.e., if they are doing better than the expert defined manager).

Research Question 2 (RQ2)

How do the three RL agents strategies (active, passive and static) compare one to the other?

If agents have been implemented correctly, the two antifragile agents (those implementing the active and passive strategy) should consistently get better QoS metrics than the resilient one (that implementing the static strategy). Moreover, the agent using the active strategy should be obtaining better figures than the one using the passive strategy in correspondence of the transition between one process and the other.

Research Question 3 (RQ3)

What is the impact of transfer learning on the RL agents using the active and passive strategies?

Ideally transfer learning should (i) reduce the downtime by helping to converge faster and (ii) improve performance by re-using the knowledge gained coming from a similar process. However, we have no guarantees on the benefits on transfer learning in practice as well as no guarantees that catastrophic forgetting may occur [25]. We compare the convergence time and the QoS metrics of the agents using active and passive strategies ablating transfer learning to understand its impact.

VI. Results

We report the main results from the two use cases in Figure 5. The main QoS metric we analyzed was the energy consumption (i.e., the objective the RL agents had to minimize). However, as shown by the other metrics, energy consumption alone is not representative of performances. In fact, for example, its reduction causes increase the packet loss.

To answer RQ1, we can see that all managing systems undergoing training with RL are obtaining better values of energy consumption than the observer managing system (which does not act on the network) in both uses cases. This implies that the agents, despite the simple linear model, are correctly learning to approach to task. However, the expert-designed managing system performs better on energy consumption (the target metric to optimize) and the network fairness index than all the considered agents, hinting that the problem may be more complex than what the linear neural network underlying the agents can capture. Nevertheless the policies learned by the agents result in a lower packet loss (as expected since the energy consumption is higher).

Concerning RQ2, the active antifragile strategy performs better, on average, than both the passive strategy and the resilient (static) one in terms of energy consumption. The difference is more evident in the second use case. This result is in line with our expectations that continuously updating

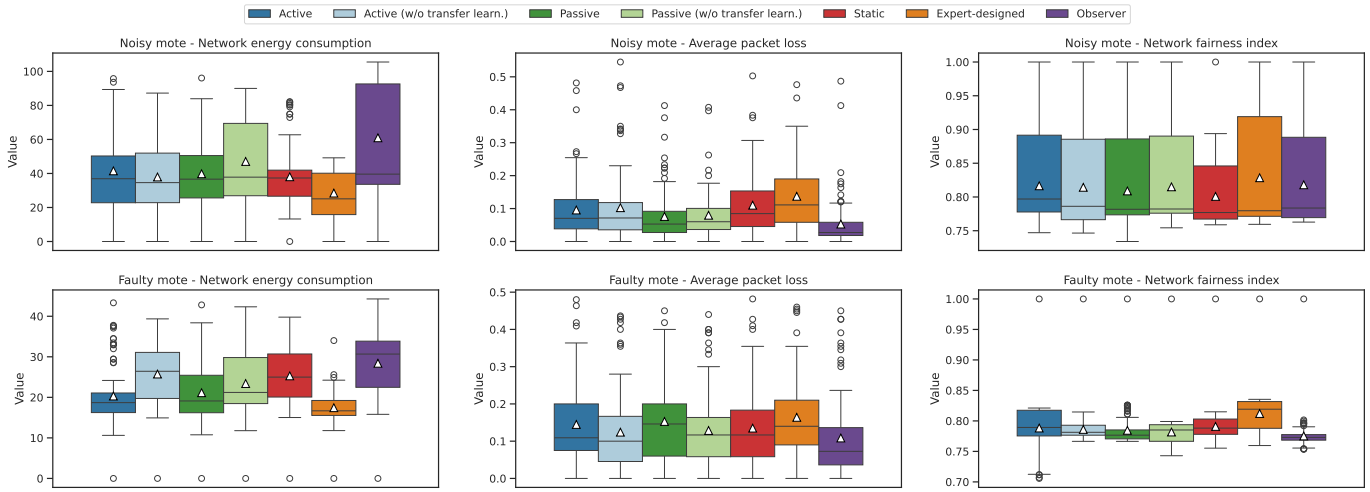

Fig. 5. Distribution of measured QoS (the white triangle indicates the average value)

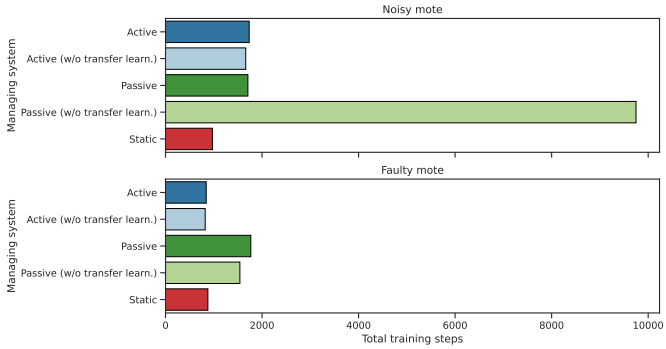

Fig. 6. Total training steps of RL agents across issued training sessions

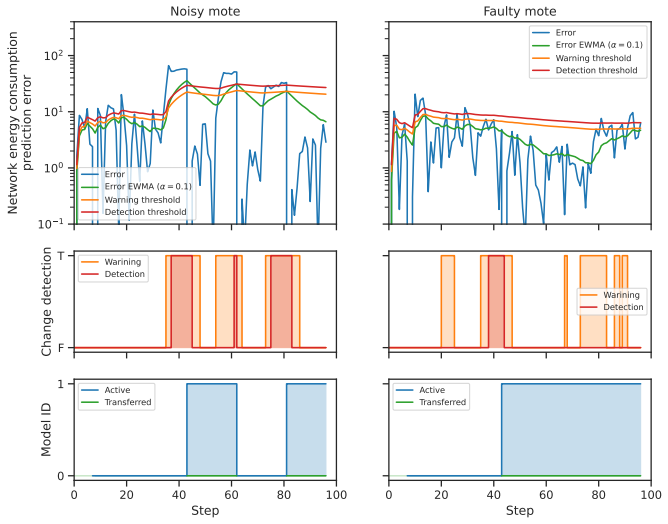

Fig. 7. Change detection process visualization

the model do not necessarily yield to better performance of the RL (even despite longer training sessions). However, results changes depending on the use case for the packet loss and the network fairness. Moreover, between the two adjustment strategies, from Figure 6 we can see that the passive one spends more time training and, thus, consuming more resources (as we were arguing for this paper) without practical benefits on the monitored QoSimproving the results.

About RQ3, we have contrasting results. In fact, no variant of the active and passive strategies with or without transfer learning performs clearly better than the other. Moreover, concerning the training steps and the convergence, which we report in Figure 6, we can see that transfer learning does not have necessarily a positive impact and that, when using a passive adjustment strategy, the models underlying the RL agents are initialization dependent. In fact, the agent using the static strategy, which trains only one at the beginning like all other agents, is not the one with the lowest number of training steps. This is a hyperparameter tuning problem, which is currently out of the scope of this work.

Finally, we report some additional considerations about the change detection mechanism based on Figure 7. Abrupt changes introducing an high level of noise like in the first use case are much more evident in the error analysis, yet, even in those cases we may miss to detect one of the changes. Moreover, the similarity metric and the choice of status representation had a strong impact on the process recognition and subsequently the choice of transferred model. Managing the change or similarity thresholds and the EWMA smoothing parameter α introduces a tradeoff between false positive and false negative in change detection and process recognition. Same applies to choice of the monitored metrics. In fact the return of the noisy mote in the first use case and the failure of the mote in the second use case were not issued looking at the error on energy consumption, but from one of the other metrics. Hyperparameters, like thresholds, should be tuned offline using techniques like cross validation to ensure the best performance possible.

Research questions summary

RQ1 RL agents correctly learn the task, but the linear model is not sufficient for expert-level performance;
RQ2 The active strategy achieve better results than the passive and static strategies on the target metric;
RQ3 Transfer affects training time and performance in different ways.

VII. Related works

Since the publication of Taleb's book on Antifragility, there has been a growing interest in applying this concept to ICT and software systems. A thorough survey presented in [26] explores both antifragility and resilience, highlighting various methods for managing disruptions within business ecosystems. This work includes a classification of disruption sources and drivers, along with a compilation of strategies for addressing disruptions and examples of engineered systems that demonstrate effective approaches to enhancing resilience and antifragility. In the context of ICT systems, [27] examines the notion of antifragility and provides multiple instances where modern ICT systems remain vulnerable to downtime and other significant disruptions. The book further discusses principles such as modularity, redundancy, and diversity-among others-that could facilitate the development of antifragile systems. A recent application of the antifragility principles and strategies to cloud native applications is illustrated in [6] Similar principles are discussed in [28], with a particular focus on software engineering and software architectures. Additionally, [29], [30] emphasize the importance of adopting a new mindset when designing antifragile systems, advocating for the integration of adaptability from the earliest stages of system conception rather than reacting to change after the fact.

Recent research in the SAS community has explored the concept of adding an upper layer to existing managing systems to introduce flexibility and advanced learning capabilities. Work in [7] proposes a lifelong self-adaptive architecture where this layer detects and manages new knowledge by updating learning models and adaptation plans in the underlying system. Similarly, [31] describes an upper layer with evaluation, learning, and verification components designed to adjust the adaptation logic of lower layers when new insights arise. This theme of self-improvement through learning is further echoed in [32]. Several frameworks, including those in [33], and [34], incorporate a MAPE-K feedback loop at this upper layer to dynamically refine the adaptation strategies of the managing system. Recursive design principles are also explored in [35], which introduces a "self-self management" layer that mirrors autonomic components, later enhanced in [36] and [37] with machine learning for real-time decision-making. Together, these approaches emphasize a shift toward systems that actively adapt by embedding learning and exploration at multiple layers.

More recently, the paper [38] proposes a framework that combines lifelong planning and dynamic knowledge distillation to optimise self-adaptation in configurable systems under time-varying workloads. It leverages past configurations only when beneficial to accelerate adaptation while mitigating misleading information With this work, we share the ideas of trying to identify similarities with past "states", and leveraging any such similarity to speed up the discovery of a suitable configuration and of using a digital twin to accelerate the identification of a new configuration.

Finally, concerning OL&M and antifragility, the idea of resorting to online learning techniques has already been considered [5]. In particular, modelling the system as a non-stationary environment where to apply online learning techniques appears to be the agreed approach. Moreover, the theoretical analysis about regret in suggests that the antifragility can benefit from proactive approaches, a direction we are willing to explore in the future, rather than passive (reactive) ones. However, to the best of our knowledge, no implementation of the OL&M framework has been developed for SAS, which instead we are proposing with this paper.

VIII. Threats to validity

For practical reasons, we conducted our experiments by simulating the evaluation subjects rather than using real systems. Using one simulation platform is a potential threat to external validity. To mitigate this issue, we consider a case study derived from the existing literature, which is open source and can be reused by other researchers for comparison. Additionally, to avoid the risk of obtaining results by chance, we considered two different evaluation scenarios to understand the generality of the results we report.

To mitigate construct validity concerns, we compared our method against established baselines, ensuring that the evaluation is grounded in meaningful and relevant metrics. Additionally, concerning the ML side, we conducted the experiments using manually tuned hyperparameters, resorting to typical values for the RL agent training (e.g., we used the default learning rate for the optimizer) and for the change detection and process recognition thresholds where possible. While properly implementing a ML pipeline would require tuning these hyperparameters by searching for the best configuration via offline preprocessing, we consider this process out of the scope of the paper, which, instead, focuses on the methodology. For the same reason, we avoid adopting deep learning approaches and we stick to linear models, which still deliver satisfying results at this point in our work. Nevertheless, we consider increasing the complexity of the neural network as a future extension when focusing on other aspects of this active adaptation strategy.

Further details about the simulation platform and the evaluation subjects are available in our replication package. Additional studies using more simulators and study subjects from other domains would increase the generality of our results.

IX. Conclusion

With this paper, we presented our view on implementing antifragility in SAS using the OL&M framework. Inspired by the human immune system, our solution adopts an active

approach to the problem of adapting an RL agent. Differently from existing lifelong learning approaches, which implement adaptation in a passive way by continuously updating the model underlying the RL agent, our solution avoids wasting resources in adapting when not needed by storing and reusing new knowledge from unforeseen, possibly disruptive, events. We experimented by comparing our methodology with alternative strategies to support the validity of our approach. Results of the experiments show that an active approach can achieve better results than static baseline approaches and slightly better results than passive approaches, but with lower resources consumption.

Starting from the methodology and the initial results described in this paper, different research directions can be explored. From the *methodological* perspective, for example, we intend to investigate how proactivity combines with our approach. From the *technical* perspective, we would like to understand better the role of the model underlying the agent, looking at more complex neural networks. Finally, given that our framework is expected to be agnostic of the managed system, we will extend the evaluation to other use cases and exemplars to understand the versatility of our methodology.

Acknowledgment

This work was supported by funding from the pilot program Core Informatics at KIT (KiKIT) of the Helmholtz Association (HGF) and supported by the German Research Foundation (DFG) - SFB 1608 - 501798263 and KASTEL Security Research Labs, Karlsruhe.

References

[1] J.-C. Laprie *et al.*, "From dependability to resilience," in *38th IEEE/IFIP Int. Conf. On dependable systems and networks*, 2008, pp. G8–G9.

[2] A. Avizienis, J. Laprie, B. Randell, and C. E. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Trans. Dependable Secur. Comput.*, vol. 1, no. 1, pp. 11–33, 2004. [Online]. Available: https://doi.org/10.1109/TDSC.2004.2

[3] N. N. Taleb, *Antifragile: Things That Gain from Disorder*. New York: Random House, 2012.

[4] O. Gheibi, D. Weyns, and F. Quin, "Applying machine learning in self-adaptive systems: A systematic literature review," *ACM Trans. Auton. Adapt. Syst.*, vol. 15, no. 3, pp. 9:1–9:37, 2020. [Online]. Available: https://doi.org/10.1145/3469440

[5] M. Jin, "Preparing for black swans: The antifragility imperative for machine learning," *CoRR*, vol. abs/2405.11397, 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2405.11397

[6] J. S. Botros, L. F. Al-Qora'n, and A. Al-Said Ahmad, "Towards antifragility of cloud systems: An adaptive chaos driven framework," *Information and Software Technology*, vol. 174, p. 107519, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950584924001241

[7] O. Gheibi and D. Weyns, "Lifelong self-adaptation: Self-adaptation meets lifelong machine learning," in *SEAMS 2022, Pittsburgh, PA, USA, May 22-24, 2022*, B. R. Schmerl, M. Maggio, and J. Cámara, Eds. ACM/IEEE, 2022, pp. 1–12. [Online]. Available: https://doi.org/10.1145/3524844.3528052

[8] G. Ditzler, M. Roveri, C. Alippi, and R. Polikar, "Learning in nonstationary environments: A survey," *IEEE Comput. Intell. Mag.*, vol. 10, no. 4, pp. 12–25, 2015. [Online]. Available: https://doi.org/10.1109/MCI.2015.2471196

[9] N. Cesa-Bianchi and G. Lugosi, *Prediction, learning, and games*. Cambridge University Press, 2006. [Online]. Available: https://doi.org/10.1017/CBO9780511546921

[10] D. M. V. Sato, S. C. D. Freitas, J. P. Barddal, and E. E. Scalabrin, "A survey on concept drift in process mining," *ACM Comput. Surv.*, vol. 54, no. 9, pp. 189:1–189:38, 2022. [Online]. Available: https://doi.org/10.1145/3472752

[11] V. Grassi, R. Mirandola, and D. Perez-Palacin, "A conceptual and architectural characterization of antifragile systems," *J. Syst. Softw.*, vol. 213, p. 112051, 2024. [Online]. Available: https://doi.org/10.1016/j.jss.2024.112051

[12] M. Basseville, I. V. Nikiforov *et al.*, *Detection of abrupt changes: theory and application*. Prentice hall Englewood Cliffs, 1993, vol. 104.

[13] R. S. Sutton and A. G. Barto, *Reinforcement learning - an introduction*, ser. Adaptive computation and machine learning. MIT Press, 1998.

[14] R. W. Melvold and R. P. Sticca, "Basic and tumor immunology: A review," *Surgical Oncology Clinics of North America*, vol. 16, no. 4, pp. 711–735, 2007, tumor Immunology for the Practicing Surgeon. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S105532070700097X

[15] World Health Organization, "Mpox outbreak," https://www.who.int/emergencies/situations/mpox-outbreak, accessed: 2025-02-06.

[16] C. Soledad, F. Silvia, K. Anders *et al.*, "Effectiveness of historical smallpox vaccination against mpox clade ii in men in denmark, france, the netherlands and spain, 2022," *Euro Surveill.*, vol. 29, no. 34, 2024.

[17] J. Kephart and D. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, 2003.

[18] D. Weyns, *An Introduction to Self-adaptive Systems: A Contemporary Software Engineering Perspective*, ser. IEEE Press. Wiley, 2020.

[19] M. U. Iftikhar, G. S. Ramachandran, P. Bollansée, D. Weyns, and D. Hughes, "Deltaiot: A self-adaptive internet of things exemplar," in *12th IEEE/ACM International Symposium on Software Engineering for Adaptive and Self-Managing Systems, SEAMS@ICSE 2017, Buenos Aires, Argentina, May 22-23, 2017*. IEEE Computer Society, 2017, pp. 76–82. [Online]. Available: https://doi.org/10.1109/SEAMS.2017.21

[20] ——, "Deltaiot: A real world exemplar for self-adaptive internet of things (artifact)," *Dagstuhl Artifacts Ser.*, vol. 3, no. 1, pp. 04:1–04:2, 2017. [Online]. Available: https://doi.org/10.4230/DARTS.3.1.4

[21] M. D'Angelo, M. Caporuscio, V. Grassi, and R. Mirandola, "Decentralized learning for self-adaptive qos-aware service assembly," *Future Gener. Comput. Syst.*, vol. 108, pp. 210–227, 2020. [Online]. Available: https://doi.org/10.1016/j.future.2020.02.027

[22] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. A. Riedmiller, A. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nat.*, vol. 518, no. 7540, pp. 529–533, 2015. [Online]. Available: https://doi.org/10.1038/nature14236

[23] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1412.6980

[24] G. J. Ross, N. M. Adams, D. K. Tasoulis, and D. J. Hand, "Exponentially weighted moving average charts for detecting concept drift," *Pattern Recognit. Lett.*, vol. 33, no. 2, pp. 191–198, 2012. [Online]. Available: https://doi.org/10.1016/j.patrec.2011.08.019

[25] R. M. French, "Catastrophic forgetting in connectionist networks," *Trends in cognitive sciences*, vol. 3, no. 4, pp. 128–135, 1999.

[26] J. Ramezani and L. M. Camarinha-Matos, "Approaches for resilience and antifragility in collaborative business ecosystems," *Technological Forecasting and Social Change*, vol. 151, p. 119846, 2020.

[27] K. Hole, *Anti-fragile ICT Systems*, ser. Simula SpringerBriefs on Computing. Springer International Publishing, 2016.

[28] D. Russo and P. Ciancarini, "Towards antifragile software architectures," *Procedia Computer Science*, vol. 109, pp. 929–934, 12 2017.

[29] H. de Bruijn, A. Größler, and N. Videira, "Antifragility as a design criterion for modelling dynamic systems," *Systems Research and Behavioral Science*, vol. 37, no. 1, pp. 23–37, 2020. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/sres.2574

[30] A. Gorgeon, "Anti-fragile information systems," in *Proceedings of the International Conference on Information Systems - Exploring the Information Frontier, ICIS 2015, Fort Worth, Texas, USA, December 13-16, 2015*, T. A. Carte, A. Heinzl, and C. Urquhart, Eds. Association for Information Systems, 2015. [Online]. Available: http://aisel.aisnet.org/icis2015/proceedings/BreakoutIdeas/6

[31] V. Klös, T. Göthel, and S. Glesner, "Comprehensible and dependable self-learning self-adaptive systems," *Journal of Systems Architecture*, vol. 85-86, pp. 28–42, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1383762117304472

[32] M. Baruwal Chhetri, A. Uzunov, B. Vo, S. Nepal, and R. Kowalczyk, "Self-improving autonomic systems for antifragile cyber defence: Challenges and opportunities," in *2019 IEEE International Conference on Autonomic Computing (ICAC)*, 2019, pp. 18–23.

[33] F. M. Roth, C. Krupitzer, and C. Becker, "Runtime evolution of the adaptation logic in self-adaptive systems," in *2015 IEEE International Conference on Autonomic Computing*, 2015, pp. 141–142.

[34] E. Zavala, X. Franch, J. Marco, and C. Berger, "Hafloop: An architecture for supporting highly adaptive feedback loops in self-adaptive systems," *Future Gener. Comput. Syst.*, vol. 105, no. C, p. 607–630, apr 2020. [Online]. Available: https://doi.org/10.1016/j.future.2019.12.026

[35] S. Bouchenak, F. Boyer, B. Claudel, N. De Palma, O. Gruber, and S. Sicard, "From autonomic to self-self behaviors: The jade experience," vol. 6, no. 4, 2011.

[36] M. Baruwal Chhetri, A. Uzunov, Q. B. Vo, R. Kowalczyk, M. Docking, H. Luong, I. Rajapakse, and S. Nepal, "Aware - towards distributed self-management for resilient cyber systems," in *2018 23rd International Conference on Engineering of Complex Computer Systems (ICECCS)*, 2018, pp. 185–188.

[37] A. V. Uzunov, M. Brennan, M. B. Chhetri, Q. B. Vo, R. Kowalczyk, and J. Wondoh, "Aware2-mm: A meta-model for goal-driven, contract-mediated, team-centric autonomous middleware frameworks for antifragility," in *2021 28th Asia-Pacific Software Engineering Conference (APSEC)*, 2021, pp. 547–552.

[38] Y. Ye, T. Chen, and M. Li, "Distilled lifelong self-adaptation for configurable systems," 2025. [Online]. Available: https://arxiv.org/abs/2501.00840