Addressing Tactic Volatility in Self-Adaptive Systems Using Evolved Recurrent Neural Networks and Uncertainty Reduction Tactics

Aizaz Ul Haq
Department of Software Engineering
Rochester Institute of Technology
Rochester NY, USA
au7149@rit.edu

Niranjana Deshpande
Department of Software Engineering
Rochester Institute of Technology
Rochester NY, USA
nd7896@rit.edu

AbdElRahman ElSaid
University of Puerto Rico
Mayagüez, PR 00680 USA
abdelrahman.elsaid@upr.edu

Travis Desell
Department of Software Engineering
Rochester Institute of Technology
Rochester NY, USA
tjdvse@rit.edu

Daniel E. Krutz
Department of Software Engineering
Rochester Institute of Technology
Rochester NY, USA
dxkvse@rit.edu

ABSTRACT

Self-adaptive systems frequently use *tactics* to perform adaptations. Tactic examples include the implementation of additional security measures when an intrusion is detected, or activating a cooling mechanism when temperature thresholds are surpassed. *Tactic volatility* occurs in real-world systems and is defined as variable behavior in the attributes of a tactic, such as its latency or cost. A system's inability to effectively account for tactic volatility adversely impacts its efficiency and resiliency against the dynamics of real-world environments. To enable systems' efficiency against tactic volatility, we propose a *Tactic Volatility Aware* (TVA-E) process utilizing *evolved Recurrent Neural Networks* (eRNN) to provide accurate tactic predictions. TVA-E is also the first known process to take advantage of *uncertainty reduction tactics* to provide additional information to the decision-making process and reduce uncertainty. TVA-E easily integrates into popular adaptation processes enabling it to immediately benefit a large number of existing self-adaptive systems. Simulations using 52,106 tactic records demonstrate that: I) eRNN is an effective prediction mechanism, II) TVA-E represents an improvement over existing state-of-the-art processes in accounting for tactic volatility, and III) Uncertainty reduction tactics are beneficial in accounting for tactic volatility. The developed dataset and tool can be found at https://tacticvolatility.github.io/

CCS CONCEPTS

• **Theory of computation → Semi-supervised learning**; • **Computing methodologies → Planning under uncertainty**; **Semi-supervised learning settings**; **Genetic algorithms**.

GECCO '22, July 9–13, 2022, Boston, MA, USA

ACM ISBN 978-1-4503-9237-2/22/07…$15.00
https://doi.org/10.1145/3512290.3528745

KEYWORDS

Adaptive Systems, Uncertainty, Deep Learning, Recurrent Neural Networks

ACM Reference Format:
Aizaz Ul Haq, Niranjana Deshpande, AbdElRahman ElSaid, Travis Desell, and Daniel E. Krutz. 2022. Addressing Tactic Volatility in Self-Adaptive Systems Using Evolved Recurrent Neural Networks and Uncertainty Reduction Tactics. In *Genetic and Evolutionary Computation Conference (GECCO '22), July 9–13, 2022, Boston, MA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3512290.3528745

1 INTRODUCTION

Self-adaptive systems utilize *adaptation tactics* to respond to events and accomplish system goals [4, 27]. Example tactics include a web farm provisioning an additional virtual machine (VM) when the workload reaches a specific threshold, or reducing non-essential functionality on an autonomous Unmanned Aerial Vehicle (UAV) when battery levels are low. Many tactics will have attributes in the form of latency and cost. Tactic latency is the amount of time from when a tactic is invoked until its effect on the system is realized [29, 30]. Tactic cost is the resources necessary to complete the execution of the tactic, where 'cost' could be in terms of energy, computation cycles, monetary values or other resources [33].

Tactic volatility frequently has a significant impact on the system's decision-making process, as it can directly impact if and when a tactic is selected and executed [27, 30]. For example, tactic latency volatility may lead to scenarios where tactics are begun too early or too late [27, 29]. Tactic cost volatility may result in a system implementing a tactic that is more expensive compared with a tantamount, but less expensive alternative.

Problematically, the vast majority of state-of-the-art decision-making processes do not account for tactic volatility and merely assume that tactics have static, unchanging attributes [27]. This inhibits the system's ability to account for real-world variability in its decision-making process and has been shown to adversely impact the system's effectiveness, resiliency and even security [18, 33, 39]. Accounting for tactic volatility is challenging due to the diverse and variable environments that many systems operate within.

We propose a *Tactic Volatility Aware* process with *evolved Recurrent Neural Networks* (eRNN) (TVA-E) to address the limitations of current systems in properly accounting for tactic volatility [33]. The system first predicts anticipated tactic attributes using *evolved Recurrent Neural Networks* (eRNNs). These predicted tactic attributes are then incorporated into the system's existing decision-making process, providing it with more accurate information. *Uncertainty reduction tactics* (URT) [28] are also integrated into the system's adaptation control loop. Uncertainty reduction tactics are actions that reduce uncertainty, and improve the quality and quantity of knowledge used in the decision-making process. The novelty of this work, compared other studies [27, 33], is that it is the first known effort to: I) Evaluate the use of a neuroevolution (NE) algorithm for accounting for tactic volatility, and II) Evaluate the use of Uncertainty Reduction Tactics for enabling a system to better account for tactic volatility.

This work advances the state-of-the-art enabling self-adaptive processes to better account for tactic volatility by implementing uncertainty reduction tactics and evaluate their ability to assist in accounting for tactic volatility. This work also broadens the field of machine learning applications to address tactic volatility analysis and mitigation. Moreover, using Neural Architecture Search (NAS) methods to optimize neural structures and parameters has been well applied to feed-forward ANNs for tasks involving static inputs, including convolutional variants [26, 40, 42], for the automated design of recurrent neural networks [8, 9, 35], for robotic controllers and game playing [10, 19, 25, 36–38], and for evolving RNNs for time series data prediction has not been well studied [41]. This work advances these efforts by applying evolving eRNN as a prediction mechanism, and investigating its benefits in accounting for tactic volatility.

Our evaluation addresses the following research questions: **RQ1:** *Is eRNN effective for predicting tactic volatility?* We demonstrate effectiveness of eRNN for predicting tactic volatility. **RQ2:** *Does eRNN effectively transfer the tactic volatility predictions to effective decisions compared with other baselines?* eRNN effectively leads to more effective system actions when compared with existing techniques. **RQ3:** *Are Uncertainty Reduction Tactics effective in helping to predict tactic volatility?* We demonstrate the foundational benefits of uncertainty reduction tactics in self-adaptive systems, specifically helping to make more accurate predictions regarding tactic latency and cost.

2 PROBLEM DEFINITION

The attributes of a tactic are frequently primary inputs into the system's *utility* calculation and overall decision-making process [27, 29]. The attributes of a tactic will frequently experience *tactic volatility*, which is any rapid or unpredictable change that exists within the attributes of a tactic [33]. Due to their prevalence [27, 33], the two forms of volatility addressed in our work are *latency* and *cost*.

2.1 Tactic Latency Volatility

Tactic latency is defined as the amount of time from when a tactic is initiated until its effect is produced [27, 29]. Examples of tactic latency include the amount of time required to fully activate a VM, or the necessary time to perform a system operation such as transferring a file over a network. Research is beginning to demonstrate the benefits of accounting for tactic latency in the self-adaptive decision-making process [6, 27, 29]. Unfortunately, many state-of-the-art adaptation processes still consider tactic latency to be a static value, one that does not change [33]. However, real-world systems will frequently encounter scenarios that experience tactic latency volatility. Accounting for tactic latency volatility is imperative for several reasons including:

(1) **Determine the most appropriate tactic(s):** The anticipated latency of a tactic will likely be a significant determining factor for choosing the most appropriate tactic(s) that should be implemented by the system [27].
(2) **Augment a slower tactic with a faster one:** Sometimes a system will use a faster, but less optimal tactic to supplement a more optimal, but slower tactic [27]. This will enable the system to begin to realize at least some of the benefits from an adaptation decision earlier, while it is waiting for the slower tactic to be ready. When determining when and if to augment a slower tactic with a faster one, understanding tactic volatility is essential for knowing when and which tactic(s) to use.
(3) **Understanding when to start a tactic:** A robust proactive system will determine when a tactic is required, and optimally begin the execution of the tactic with a sufficient amount of time enabling it to be ready when needed. A system that is unable to account for tactic latency volatility will likely begin tactics too early (incurring additional costs) or too late (not available when needed) [27, 33].
(4) **Incompatible tactics, and tactics that must be run in unison or succession:** Certain tactics may not be run whenever another incompatible tactic is being executed. Contrarily, there may be cases when two tactics must be executed in unison. Properly anticipating tactic latency is imperative for both of these scenarios. If multiple tactics are incompatible, understanding when a tactic is expected to begin and conclude is imperative for planning the execution of conflicting tactics. If multiple tactics must be executed in unison and at least one of these tactics contains latency, then planning must be done to anticipate when to begin the execution of the tactics so they are both properly executed together. Tactics may also need to be run in succession of one another.

2.2 Tactic Cost Volatility

The definition of tactic cost will widely vary and is likely to be domain-specific. Some examples of tactic cost include the energy, computations, or monetary value necessary to complete the execution of a tactic. Estimating tactic cost will likely be a primary concern for a self-adaptive system, especially if there are defined resource limitations or the system simply has the goal of achieving the highest reward at the lowest cost [27].

(1) **Cost may be a determining factor when selecting between multiple tactic options:** A tactic's cost may be the decisive factor for determining the most appropriate tactic that a system should select. Therefore, to determine the most appropriate tactic option(s), an accurate cost estimation is imperative.

(2) **The anticipated cost may impact the system's ability to perform subsequent or concurrent tactics:** Systems frequently have a finite amount of resources. Therefore, it is imperative to accurately predict the cost of an action to: I) Determine if there are enough resources to perform the examined action, II) Select tactics to accrue the highest amount of possible utility.
(3) **Cost may exceed reward:** Systems will frequently determine the expected cost of performing an operation and, in conjunction with the reward, determine if the operation is worth performing. If the cost exceeds the reward, then it may not be optimal for the system to execute the tactic.

2.3 Uncertainty Reduction Tactics

Tactics are frequently used by self-adaptive systems to respond to events and accomplish system goals [23, 27]. *Uncertainty reduction tactics* (URT) are tactics that are specifically designed to reduce a system's uncertainty, and improve the quality and quantity of knowledge used in the decision-making process [28]. Example uncertainty reduction tactics include probing a sensor that has not recently been used to determine its response time and to ensure that it is still active/available. Depending on the domain, numerous forms of data can be attained from uncertainty reduction tactics, some of which include information regarding the availability, response time and reliability of a resource. Uncertainty reduction tactics differ from conventional tactics in that the only objective of an uncertainty reduction tactic is to gather data and reduce uncertainty. An advantage of uncertainty reduction tactics is that they can provide additional meta data regarding an operation, typically without necessitating the cost/risk of executing the operation itself. While various uncertainty reduction tactics have been proposed, to our knowledge no existing works have implemented or evaluated uncertainty reduction tactics. We hypothesize that uncertainty reduction tactics can augment predictive processes such as eRNN to help to make more accurate tactic-based predictions and therefore positively impact the system's decision-making process.

2.4 Evolved Recurrent Neural Networks

Recurrent neural networks (RNNs) generally outperform classical statistical methods, *e.g.,* those from the auto-regressive integrated moving average (ARIMA) family of models, on data that is highly non-linear, acyclic and not seasonal. Further, classical statistical methods are not well suited to time series forecasting which incorporates multiple correlated time series of input data. This makes the use of RNNs a much stronger choice for providing predictions for self-adaptive systems.

Hand-crafting effective and efficient structures for RNNs is a difficult, expensive, and time-consuming process. Instead of simply evaluating standard RNN architectures, neuro-evolution can be used to select, connect, and combine potential architectural components – yielding a more rigorous and comprehensive search over available architectures, automating the design and training process.

For this study, the Evolutionary eXploration of Augmenting Memory Models (EXAMM) algorithm [32] was selected for the neuro-evolution process for several reasons. First, EXAMM progressively grows larger ANNs starting from a minimal seed architecture, in a manner similar to the popular Neuro-Evolution of Augmenting Toplogies (NEAT) algorithm [40] that makes it tend to generate smaller and more efficient architectures. Furthermore, in contrast to NEAT, EXAMM utilizes higher-order node-level mutation operations, Lamarckian weight initialization (or the reuse of parental weights), and backpropagation through time (BPTT) to conduct local search, the combination of which has been shown to speed up both ANN training as well as the overall evolutionary process [12]. EXAMM also operates with an easily-extensible suite of memory cells, not found in other strategies, including LSTM, GRU, MGU, UGRNN, Δ-RNN cells and, more importantly, has the natural ability to evolve deep recurrent connections over large, variable time lags. Other research has also demonstrated that EXAMM has been shown to more quickly and reliably evolve RNNs in parallel than training traditional layered RNNs sequentially [14, 16, 17].

2.5 Motivating Example

A cloud-based multi-tier web application with a self-adaptive component will serve as the motivating example. The system's goal is to maximize utility while minimizing cost, where the application is comprised of Virtual Machine (VM) servers that process incoming requests. Each VM instance has a defined cost and VMs can be added or removed from the resource pool as user demand dictates. Optional content (*e.g.,* advertisements can be reduced using the 'dimmer' feature to efficiently provide content while encountering variable workloads. This example is based on work by Moreno [27].

The target response time (T) and utility (U) is calculated (Equation 1) as defined by the *Service Level Agreement* (SLA). Penalties are incurred if the target response time is not met, while rewards are accrued for meeting the target average response time against the measurement interval. The average response rate is a, the average response time is r, the maximum request is k and the length of each interval is defined as τ. A dimmer (d) reduces provided content as necessary and cost (C) is proportional to the number of active VMs. In this example, the reward of the optional content (R_O) produces a higher utility than mandatory content (R_M).

$$U = \begin{cases} (\tau a(dR_O + (1-d)R_M)/C & r \leq T \\ (\tau \min(0, a-k)R_O)/C & r > T \end{cases} \quad (1)$$

Two tactics can be used to account for increases in user traffic: I) Add an additional VM server, or II) Use the *dimmer* to reduce the proportion of responses that include the optional content. The tactic of adding a new VM server can take several minutes, while reducing optional content has negligible latency.

Tactic latency volatility If the system anticipates that the response time threshold will be surpassed in the immediate future, then it could proactively start the tactic of adding a VM server to keep the response time under the defined threshold. If the system overestimates latency, then it could incur superfluous additional costs since the VMs would be active longer than necessary. If the system determines that the defined response time threshold is going to be surpassed before a new VM server can be added, then an appropriate action may be to use the faster tactic of reducing optional content while the new VM server is being added. Inaccurate tactic latency predictions can result in scenarios where the system executes a tactic too early or too late, or even selects the improper

tactic for the encountered scenario. Therefore, *accounting for tactic latency volatility is a paramount concern.*

Tactic cost volatility Accounting for tactic cost volatility is important in this scenario since cost is an integral piece of the utility calculation. If cost is defined to be lower than what is being actually being regularly encountered, then this could result in scenarios where optional (*O*) content is shown too frequently. Conversely, if cost is defined to be higher than is being routinely encountered, then this could result in inaccurate utility calculations and scenarios where optional (*O*) content is shown too infrequently (leading to less reward). *A volatility aware solution that enables the system to more accurately predict cost would enable the system to make decisions that lead to more optimal outcomes.*

Uncertainty Reduction Tactics *Uncertainty reduction tactics* [28] are system actions that are specifically designed to reduce a system's uncertainty, and improve the quality and quantity of knowledge used in the decision making process. Example uncertainty reduction tactics include probing a sensor that has not recently been used or pinging a remote resource to ensure that it is still active. Uncertainty reduction tactics can provide valuable insight into predicting the latency and cost of executing a tactic such as activating an additional VM. While various uncertainty reduction tactics have been proposed, to our knowledge no existing works have implemented or evaluated uncertainty reduction tactics in the autonomous decision making process. *Uncertainty reduction tactics may provide a lightweight, supportive input mechanism into making more accurate predictions regarding tactic volatility.*

3 PROPOSED TACTIC VOLATILITY AWARE PROCESS

Most self-adaptive processes used for autonomous intelligent decision making employ a form of an *adaptation control loop* [20]. An adaption loop cycles through all possible adaptation options at defined intervals, selecting the *adaptation strategy(s)* that will return the highest utility. An adaptation strategy is a predefined composition of adaptation tactics (benefit) [27]. Using the motivating example in Section 2.5, during each adaptation control loop the system selects the adaptation strategy that is expected to return the highest utility. As described in Section 2, predicting the volatility of the tactic is likely a crucial component of the decision-making process due to the impact that these predictions can have on the system's effectiveness, resiliency and ability to complete system and mission critical operations [33]. Our TVA-E process does not represent a new adaptation decision-making process, but provides more accurate information to existing adaptation processes.

An overview of our TVA-E process is shown in Algorithm 1. TVA-E integrates into the system's existing adaptation control loop (L2), iterating through all existing adaptation options (L3). Predictions regarding the attributes of the tactic (*e.g.,* latency, cost, etc.) are performed using existing time-series data (L4). These predictions are then incorporated into the system's utility equation to determine the 'goodness' of each adaptation option (L5). This anticipated utility then serves as a primary input to the system's decision-making process (L6). For example, depending on the system it may select the strategy with the highest utility (L7). After the system executes

Algorithm 1 Integration of TVA-E into Adaptation Loop

Input: Time series data
Output: Adaptation decision
1: —Existing adaptation control loop—
2: **procedure** Adaptation Determination
3: **for each** Adaptation Option **do**
4: $\hat{TacticVariables} \leftarrow prediction(TimeSeriesData)$
5: $\hat{predictedUtility} \leftarrow UtilityCalc(\hat{TacticVariables})$
6: $adaptationToExecute \leftarrow getmax(predictedUtility)$
7: $executeMaxAdaptation(adaptationToExecute)$;
8: $TimeSeriesData \leftarrow ObservedTacticAttributes()$
9: $TimeSeriesData \leftarrow UncertaintyReductionTactic()$
10: —Existing adaptation control loop—

the tactic(s), the observed tactic attributes are recorded as time-series data (L8). Information recorded from the execution of the uncertainty reduction tactic is also stored as time-series data for future predictions (L9).

We chose eRNN for our TVA-E process due to its demonstrated ability to make accurate and reliable predictions using limited amounts of data in comparison with alternative options [14–17]. However, a myriad of alternative prediction techniques could be easily incorporated into our process to account for tactic volatility.

Our proposed TVA-E process will easily integrate into popular adaptation processes such as Proactive Latency-Aware (PLA) [27] and MAPE-K [11], which is the most influential reference control model for autonomic and self-adaptive systems [13]. Integration into existing adaption processes is supported by using existing data to perform tactic predictions, and then providing the improved tactic predictions to the system's existing decision-making process. For example, the proposed TVA-E process will integrate into the *Analyze* component of MAPE-K, providing enhanced tactic values to the *Plan* component. This makes TVA-E highly applicable to a large number of existing self-adaptive processes and systems ranging from simple cyber-physical systems to large UAVs.

4 TACTIC SIMULATION TOOL AND DATASET

An additional contribution of this work is the creation of the *CELIA* (taCtic EvaLuator sImulAtor) tool. The objective of CELIA's simulation component is to regularly collect an updated version of a file from the remote location, with the goal of maximizing utility. CELIA uses the utility equation: $\hat{U} = (\frac{Reward}{(\hat{Latency}+\hat{Cost})})$ to determine if the expected reward in relation to the expected latency and cost of the operation warrants the file collection and extraction operations, or if the system should 'pass' and wait for another decision-making iteration. The expected reward is compared against a pre-determined threshold value according to a system's Service Level Agreement (SLA). When determining if the system should perform tactic operations, the simulation component takes two files as input: I) The predicted values for each tactic attribute (latency, cost), and II) The observed tactic attributes which serve as the ground truth. CELIA iterates through each adaptation decision-making cycle first using the predicted tactic attributes, and then the actual values (the ground truth). CELIA provides several output values, such as the absolute difference of the observed (ground truth) vs. expected utility and if the adaptation decision was correct.

CELIA emulates a simple intelligent self-adaptive system that is tasked with downloading and processing a file from multiple remote locations. It repeatedly performs the following: I) download a compressed file from remote servers, emulating the tactic of performing a communication or file transmission operation; II) extract the file's contents, emulating the tactic of performing a simple file I/O operation; III) perform a `grep` of the extracted contents; IV) compress the extracted files contents, providing an additional tactic example of a file I/O operation; V) delete the file thus providing an additional example of a file i/o tactic; and VI) A simple uncertainty reduction tactic that gathers augmenting information regarding the availability and response time of the remote resource. The data created by CELIA is important since existing resources such as 'The Internet Traffic Archive'[2] do not contain both latency and cost values for performed actions. Our created data, source code and Docker image is provided on the project website [1].

For this work, the 'download' tactic perpetually downloads the Apache installation file[1] from three real-world hosting servers located around the world (USA, Switzerland, Canada). Servers hosting Apache were chosen due to their prevalence and that they were anticipated to experience real-world volatility. Variability and uncertainty are inherently included in these operations since they are being conducted using 'live' servers located across the world, providing impacts such as periodic latency and communication challenges just as encountered in any real-world operation. Tactic latency is the amount of time necessary to perform each operation, while tactic cost is average CPU resources used for the operation. Overall, 52,106 tactic operations were used in our evaluation.

We chose to evaluate the ability of the examined methods using this tactic as it contained the most real-world volatility, emulating the actions that a file transfer operation would resemble in a self-adaptive system. This tactic consistently experienced real-world volatility due to numerous factors such as network and server variability. Additional reasons for focusing on this tactic were that: I) The latency and cost variability provided by this server was sufficient to provide a proper evaluation, and II) To simplify the analysis output/examination.

5 EXPERIMENTAL DESIGN

To determine the effectiveness of eRNN in relation to alternative options, we evaluated the following prediction mechanisms against our eRNN solution: AutoRegressive Integrated Moving Average (ARIMA), Support Vector Regressor with a Radial Basis Function kernel (SVR-RBF), one layer Multi-layer Perception (MLP, *e.g.,* feed forward neural networks) and Long Short-term Memory (LSTM) recurrent neural networks. We compared our eRNN-based technique against these techniques [3] for a variety of reasons. ARIMA was used in a similar work [33], while LSTM RNNs are well known architecture for forecasting multivariate time-series data because of their gated memory cells which utilize information from previous time steps [34]. MLP are basic feed forward neural networks that consist of only feed forward fully connected hidden layers, therefore serving as a baseline neural network model. We also included SVR as it is a sophisticated model which has been shown to perform well on time series data prediction [24, 31]. Each model was provided a total of 52,106 records (36,472 records for training, 7,819 for testing and 7,815 for validation).

For our experiments, we used ARIMA (1, 1, 0), with a differencing order of d = 1 as the time-series data was non-stationary. Autocorrelation plots were used to determine autoregressive (p=1) and moving terms (q=0). Support vector regression (SVR) was performed with a radial basis function (RBF) kernel and a linear kernel. The regularization parameters and hyper-parameters of the kernel function were set to the default values of 1.0 and 0.1. The MLP model was fully connected with one hidden layer of 100 nodes with the output layer predicting cost and latency. The LSTM model was fully connected, consisting of a single hidden layer with 1,000 LSTM nodes followed by the output layer.

The neural networks and EXAMM processes trained the neural networks using the training data and utilized the testing data to optimize hyperparameters and determine when to stop the training process. The resulting eRNNs were quite small, with the largest network having only 44 total nodes and 590 weights. The validation data was excluded from the evolutionary process and only used at the conclusion to determine the final performance and make sure the trained networks were generalizable.

After demonstrating the benefits of eRNN compared to other evaluated machine learning options, we then assessed the benefits of uncertainty reduction tactics used in conjunction with our eRNN-based process. We evaluated URTs with eRNN only since eRNN was found to be the most effective prediction mechanism that was able to utilize URTs. ARIMA is a univariate model and is unable to leverage information from other sensors (URT # 1). This evaluation was accomplished by implementing two uncertainty reduction tactics that were chosen due to their real-world applicability [43] and discussion in a previous work [28]. The two URTs used in our evaluation were:

URT #1: Reducing uncertainty due to model drift - The models used may progressively become inconsistent with the system's environment due to various forms of internal and external volatility. This uncertainty can be addressed by incorporating additional sensors or data gathering operations [28]. In our evaluation, we emulate this uncertainty reduction operation by regularly 'pinging' the remote server to attain information about its availability and response time. This information is then incorporated into the system's tactic prediction process.

URT #2: Changing the sampling rate of a parameter - The mean and variance of the observations can generate a confidence interval for the monitored parameter value. Adapting the sampling rate is one method of controlling the width, and uncertainty of the interval [43]. We emulate such an uncertainty reduction operation by incorporating every *n*th observation into our prediction process. The *n*th value would be determined according to system specifications. For this experiment, we chose *n* as 5, 10, 20. These values were chosen to mimic loss of information that would occur by changing sampling rate with respect to the size of our dataset. Values less than 5 did not represent a significant reduction in sample size due to the magnitude of our data. We also chose a sampling rate of 10 and 20 by doubling the previous sampling rate. Our data didn't allow for values beyond 20 because the amount of data was not sufficient for our experiments.

[1] https://downloads.apache.org/httpd/httpd-2.4.43.tar.gz

URT #2 is a flexible framework that allows us to adjust the proper URT strategy to specific learning tasks. In this experiment, we conduct extensive cross-validation and choose n=5 for latency prediction and n=20 for cost prediction for optimal prediction improvement. The advantage of such flexibility may not be significant in this study since the utility is only determined by two factors (latency and cost). However, if we consider a more complex simulation environment where the utility depends on hundreds or thousands of factors, we can use customized sampling rates to minimize the uncertainty in each factor input resulting in a considerable impact on the utility score.

The following evaluation criteria was used in our analysis:

(1) **Ability to accurately predict tactic attributes:** The predicted attributes of a tactic frequently have a significant impact on the system's decision-making process, as it can directly impact if and when a tactic is selected and executed [27, 30]. Therefore, it is important for a system to accurately predict the attributes of a tactic to ensure effective, efficient and resilient functionality [33]. In our evaluation, we compared the predicted tactic latency and cost values against the observed ground truth values.
(2) **Expected vs achieved utility:** We evaluated the system's ability to achieve the predicted amount of utility while operating in dynamic environments with variable data. Accurately anticipating the achieved utility is imperative for a system's decision-making process for choosing the most appropriate tactic(s) and adaptation strategies for the encountered scenario [21, 27, 29, 30]. In our evaluation, we compared the predicted utility vs. the observed utility (ground truth).
(3) **Ability to make correct decisions:** The ability to make proper decisions is paramount for self-adaptive systems, therefore we recorded when the system: *a)* Performed an action when the system should not have, *b)* Performed an action when the system should have, *c)* Didn't perform an action when it should not have, and *d)* Didn't perform an action when it should have.
Our evaluation compared the predicted recommended system action vs the action that the system should have taken using observed values (ground truth). We evaluated the impact of each prediction mechanism on a system's decision-making process both quantitatively and qualitatively.

6 EXPERIMENTAL RESULTS

RQ1: Is eRNN effective for predicting tactic volatility? This research demonstrates the capabilities of eRNN in both the general prediction process, and more specifically in benefiting systems in accounting for tactic volatility. For tactic latency and cost, we first use the mean squared error (MSE) to evaluate the predicted result, as it is a common loss function for regression tasks. However, MSE is known to be sensitive to the scale of the data and the outliers. Therefore, we also reported the Mean absolute error (MAE) as an alternative error measurement.The two measurements are given by: $MSE = \frac{1}{N}\sum_{i=1}^{N}(t_i - \hat{t}_i)^2$, and $MAE = \frac{1}{N}\sum_{i=1}^{N}|t_i - \hat{t}_i|$. , where N is the total number of test cases, t is the ground truth value (*e.g.,* latency or cost) and $\hat{t}_i$ is the predicted value. ARIMA utilizes univariate data to make predictions, therefore using data from uncertainty reduction tactics in addition to cost or latency data is not possible (and is therefore not reported). We ran each model on the three different download servers representing various tactic options (Section 4). We report the averaged performance measurements of the three servers ($\overline{MSE}$, $\overline{MAE}$) in Table 1.

Table 1: eRNN's capabilities for predicting tactic latency.

URT tactic	Model	$\overline{MSE} \times 10^{-3}$	$\overline{MAE} \times 10^{-1}$
no URT	**eRNN**	**0.61**	**0.16**
	ARIMA	0.69	0.17
	LSTM	0.83	0.16
	MLP	0.86	0.16
	SVR_linear	8.87	0.35
	SVR_rbf	9.0	0.36
URT_Combine (Sample rate=5)	**eRNN**	**0.57(7%↑)**	**1.15(6%↑)**

- Table 1 demonstrate that eRNN achieves the best performance for predicting latency and cost. The uncertainty reduction tactic (URT) doesn't help eRNN to improve the predictive power in terms of MSE, but does help with MAE.
- Figure 2 demonstrates that the predicted latency has a large number of outliers in the observed values (ground truth). To better evaluate the model in such a case, we further report the MAE for each compared model as it is less sensitive to outliers. A small percentage of outliers will likely have little impact on the functionality of the system from a more aggregate level (*e.g.,* it doesn't matter if the prediction that leads to an non-optimal action is only slightly wrong or very wrong, the incorrect action was still taken). In RQ #2, we further justify our claim that the higher MAE does not prevent eRNN from making effective decisions.
- The additional uncertainty reduction tactic information has a significant impact on eRNN for cost prediction. The MSE is reduced by approximately half when uncertainty reduction tactic information trains the eRNN.
- The additional uncertainty reduction tactic information for model training serves as regularization. Figure 1 and Figure 2 demonstrate that without uncertainty reduction tactic information, eRNN tends to provide more predictions that are close to the outliers (which are likely to be nothing more but systematic noises). Such an overfitting issue is well addressed with the presence of uncertainty reduction tactic information.
- eRNN provides significantly more efficient networks. The MLP models had 900 and 1,100 trainable parameters (without and with URT #1), the LSTM models had 19,000 and 21,000 trainable parameters (without and with URT #1), and the eRNN models had 485 and 590 weights (without and with URT #1). EXAMM was able to find RNNs that not only had better performance, but were orders of magnitude smaller in terms of trainable connections – highlighting the inefficiency of using traditional fixed architectures. These evolved networks have the potential to more easily be embedded in systems with computational and power limitations, which is common in adaptive tactical scenarios [5, 22].

Outcome: *Our findings demonstrate that eRNN is effective at both making predictions, and specifically at predicting tactic cost and latency volatility.*

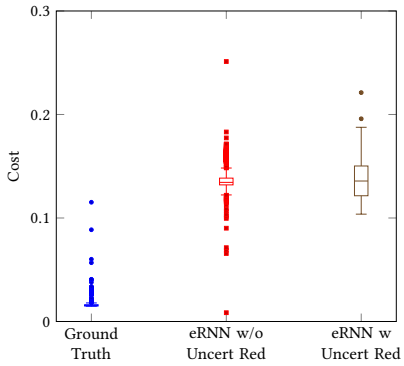

Figure 1: Predicted tactic cost (on server1)

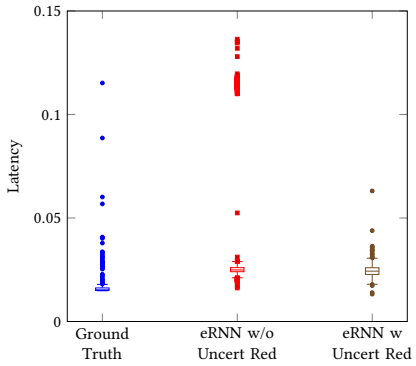

Figure 2: Predicted tactic latency (on server1)

Table 2: Evaluation of utility prediction demonstrating the how predicted utility deviates from the true utility.

Model	$\overline{MAPE}$
eRNN	21%
ARIMA	13%
LSTM	16%
MLP	14%
SVR_linear	57%
SVR_rbf	57%
eRNN+URT_Combine (Sample rate=5)	16%

RQ2: Does eRNN effectively transfer the tactic volatility predictions to effective decisions compared with other baselines? Unlike the tactic cost and latency whose values are generally distributed around 1, the calculated utility's value is not well scaled (range from 0 to 16,441.9). As a result, we use the scale-independent measurement, mean absolute percentage error (MAPE), instead of MSE to evaluate the utility prediction. The formula for MAPE is given by: $MAPE = \frac{1}{N}\sum_{i=1}^{N}|\frac{t_i-\hat{t}_i}{t_i}|$.

In Table 2 the MAPE exceeding 100% indicates that the model is likely to heavily overestimate the utility especially when the true utility is small (close to zero). This is usually considered a significant prediction failure. In self-adaptive systems, the large prediction failure has a high risk of leading the incorrect adaptation decisions. We found that eRNN performs similarly to state-of-art methods such as ARIMA, LSTM and MLP, and that its performance can be further boosted with URT operations.

Accurately anticipating utility is imperative for systems to choose the most appropriate tactic(s) and adaptation strategies for the encountered scenario [21, 27, 29, 30]. The benefits can be further improved by leveraging the uncertainty reduction data during the training phase, specifically for eRNN for which the URT produces a 24% improvement.

The final decision-making process can be seen as a binary classification task. Thus, we can evaluate the system as a binary classifier. We begin by defining several terms. The true positive (TP) represents the number of correct 'update' decisions made by the system. The true negative (TN) is the correct 'not update' decisions made by the system. The false positive (FP) is the number of incorrect 'update' decisions and the false negative is the number of incorrect 'not update' decisions. We can then evaluate the classification accuracy of the system output. This accuracy calculation reveals the proportion of the correct decisions in the system output. For the wrong decisions, we use false positive rate (FPR) and false negative rate (FNR) for evaluation where $FPR = \frac{FP}{TP+FP}$ is the frequency of wrong 'update' and $FNR = \frac{FN}{FN+TN}$ is the frequency of wrong 'not update'. Using the values from Table 3, we can conclude:

- The uncertainty reduction information helps improve the accuracy by 13% in our eRNN-based TVA-E) technique, and achieves the highest accuracy.
- The eRNN-based process has the lowest FPR. This is consistent with the previous observation on its small MAE of the utility prediction. Since our eRNN-based system rarely overestimates the small utility values, it is less likely to produce a false update tactic action.
- The eRNN-based system has moderate FPR and FNR, but results in low prediction accuracy compared with other baselines. We observe that URTs manage to rebalance the two types of mistakes (*i.e.,* FPR and FNR) so that the overall benefit (*i.e.,* the accuracy) can be maximized. In the rebalanced decision making procedure, URT chooses to tolerate the increase of FNR for this type of mistake will only have a minor negative impact if the amount of the utility missed by incorrect 'not update' decisions are small. We can use utility loss to quantify such negative impact. The utility loss is the sum of the ground truth utility scores for all incorrect 'not update' decisions. Intuitively, the utility loss measures the amount of utility a system could have gained with the correct 'update' decisions. eRNN also significantly reduced the utility loss compared to other strategies, obtaining results two orders of magnitude better than the next best strategy (MLP).
- The high FNR rate of eRNN based system has a minor negative impact on the final decision-making. The false-negative decisions made by eRNN based system have the least utility loss compared with other methods.
- Our eRNN-based TVA-E process can leverage uncertainty reduction information to reduce the utility loss to 0.1%, meaning that UCT helps eRNN to minimize the expected loss during the decision making process. Similar to utility loss, we can use utility

gain to quantify the positive impacts made by correct decisions. The utility gain is the sum of the true utility scores for all correct 'update' decisions.

Table 3: Comparison of the decisions made by the simulated self-adaptive systems using evaluated prediction mechanisms (threshold=1000).

Model	$\overline{FPR}$	$\overline{FNR}$	$\overline{U\_gain}$	$\overline{U\_loss}$	$\overline{Acc}$
eRNN	15%	7%	2.10E+07	1.40E+05	88%
ARIMA	6.6%	5.6%	2.1E+07	9.3E04	94%
LSTM	7%	11%	2.00E+07	1.90E+05	91%
MLP	9.20%	6%	2.00E+07	9.00E+04	92%
SVR_linear	0%	54%	2.00E+07	1.80E+06	59%
SVR_rbf	0%	56%	2.00E+07	1.90E+06	58%
eRNN + URT_Combine (Sample rate=5)	0.53%	17%	2.30E+06	3.40E+02	99%

Our general observations are that: *a)* eRNN is the best candidate process for latency and cost predictions, as the model performance is both accurate and robust. *b)* The strength of eRNN cost and latency prediction can be well transformed to utility prediction. *c)* Our eRNN-based process makes the most correct decisions and also maximizes the utility gain and minimize the utility loss during the decision-making. *d)* The uncertainty reduction information plays the role of regularizing the model prediction/risk control/reducing the uncertainty of the model output. *e)* Uncertainty reduction information can reduce utility loss.

Outcome: *Our analysis demonstrates that eRNN is the best method for predicting volatility compared with existing prediction techniques. We also observe that uncertainty reduction tactics provide useful information to help make more accurate decisions.*

RQ3: Are Uncertainty Reduction Tactics effective in helping to predict tactic volatility? While the potential benefits of uncertainty reduction tactics have been previously discussed [28], there have been no known efforts to demonstrate or evaluate their potential benefits in simulated self-adaptive systems. As demonstrated in RQ1 & RQ2, we found that uncertainty reduction tactics were beneficial in helping eRNN to provide better tactic latency and cost predictions.

- The effectiveness of URT is model specific. While all the models exhibit improvements with URT added, eRNN best leverages the uncertainty information and improve its MSE by 50% (on average, other methods with UTC provide less than 10% improvement).
- Additional uncertainty reduction tactic information for model training serves as regularization. Figure 1 and Figure 2 demonstrate that without uncertainty reduction tactic information, eRNN tends to provide more predictions that are close to the outliers. Overfitting issues are well addressed with the use of uncertainty reduction tactic information.

Outcome: *This work demonstrates the potential benefits of uncertainty reduction tactics, specifically in helping to make more accurate predictions regarding tactic volatility.*

7 DISCUSSION

Self-adaptive systems will be increasingly expected to efficiently, effectively, and resiliently perform in volatile environments. This work further demonstrates the need for self-adaptive systems to account for the volatility of tactics (tactic volatility). Demonstrated benefits of accounting for tactic volatility include reduced uncertainty and an increased probability of the system making optimal decisions.

This work demonstrates the benefits of eRNN in predicting tactic volatility, and also the general capabilities of our eRNN-based TVA-E process as well. These demonstrated benefits exhibit the capabilities of eRNN and provides the foundation for its application into other areas of machine learning. Thus, the findings of this work advance both research in self-adaptive systems and in a multitude of other areas of applied machine learning.

While the benefits of uncertainty reduction tactics have been discussed from a theoretical perspective [28], this is the first effort to evaluate their benefit in a simulated system. This creates the foundation for future work, such as I) The examination of uncertainty reduction tactics in additional simulated and physical environments and settings, II) Their further evaluation, III) Creation of additional uncertainty reduction tactics in a multitude of areas and purposes. The demonstrated benefits of the supplementary information provided to eRNN by uncertainty reduction tactics demonstrates the benefits of this additional information to this model. Uncertainty is widely recognized as being detrimental [7, 28], and this work demonstrates the ability of uncertainty reduction tactics to diminish the amount of uncertainty encountered by the system.

The provided dataset and simulation tool [1] will assist other researchers and practitioners in creating and evaluating their own tactic volatility aware processes. The necessity of such processes will continue to increase as the recognized need for systems to effectively function in volatile environments correspondingly grows.

8 CONCLUSION AND FUTURE WORK

Like most uncertainty reduction tactics, this work did not account for cost or risk associated with real-world uncertainty, since these aspects are negligible by design. We recognize that the specific uncertainty reduction operations used within a system will be entirely domain specific and therefore it is impossible for any work to assume that it could implement and evaluate every form of uncertainty reduction operation. Future work will further investigate uncertainty reduction tactics, and EXAMM shall be used for continuous online evolution as new information is received to offer online accurate predictions.

We propose a new *Tactic Volatility Aware* process with *evolved Recurrent Neural Networks* (TVA-E). TVA-E is able to easily integrate into popular adaptation processes, enabling it to readily and positively impact a large number of self-adaptive systems. Our simulations using 52,106 records demonstrate that: I) TVA-E can effectively account for tactic volatility, II) eRNN demonstrates its effectiveness compared with other leading machine learning alternatives, III) Uncertainty reduction tactics can be beneficial in accounting for tactic volatility. Complete results, evaluation software and other information is publicly available on the project website: https://tacticvolatility.github.io/

REFERENCES

[1] 2022. CELIA(taCtic EvaLuator sImulAtor) Tool. https://tacticvolatility.github.io/. https://tacticvolatility.github.io/

[2] 2022. The Internet Traffic Archive. http://ita.ee.lbl.gov/. http://ita.ee.lbl.gov/

[3] Nesreen K Ahmed, Amir F Atiya, Neamat El Gayar, and Hisham El-Shishiny. 2010. An empirical comparison of machine learning models for time series forecasting. *Econometric Reviews* 29, 5-6 (2010), 594–621.

[4] Konstantinos Angelopoulos, Alessandro V Papadopoulos, Vítor E Silva Souza, and John Mylopoulos. 2016. Model predictive control for software systems with CobRA. In *2016 IEEE/ACM 11th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS)*. IEEE, 35–46.

[5] Peter E Bailey, David K Lowenthal, Vignesh Ravi, Barry Rountree, Martin Schulz, and Bronis R De Supinski. 2014. Adaptive configuration selection for power-constrained heterogeneous systems. In *2014 43rd International Conference on Parallel Processing*. IEEE, 371–380.

[6] Javier Cámara, Gabriel A Moreno, and David Garlan. 2014. Stochastic game analysis and latency awareness for proactive self-adaptation. In *Proceedings of the 9th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*. ACM, 155–164.

[7] Javier Cámara, Wenxin Peng, David Garlan, and Bradley Schmerl. 2017. Reasoning about Sensing Uncertainty in Decision-Making for Self-Adaptation. In *International Conference on Software Engineering and Formal Methods*. Springer, 523–540.

[8] Andrés Camero, Jamal Toutouh, and Enrique Alba. 2018. Low-cost recurrent neural network expected performance evaluation. *arXiv preprint arXiv:1805.07159* (2018).

[9] Andrés Camero, Jamal Toutouh, and Enrique Alba. 2019. A Specialized Evolutionary Strategy Using Mean Absolute Error Random Sampling to Design Recurrent Neural Networks. *arXiv preprint arXiv:1909.02425* (2019).

[10] Luigi Cardamone, Daniele Loiacono, and Pier Luca Lanzi. 2009. On-line neuroevolution applied to the open racing car simulator. In *2009 IEEE Congress on Evolutionary Computation*. IEEE, 2622–2629.

[11] Rogério De Lemos, Holger Giese, Hausi A Müller, and Mary Shaw. 2010. Software Engineering for Self-Adaptive Systems II. (2010).

[12] Travis Desell. 2018. Accelerating the evolution of convolutional neural networks with node-level mutations and epigenetic weight initialization. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. ACM, 157–158.

[13] Ibrahim Elgendi, Md Farhad Hossain, Abbas Jamalipour, and Kumudu S Munasinghe. 2019. Protecting cyber physical systems using a learned MAPE-K model. *IEEE Access* 7 (2019), 90954–90963.

[14] AbdElRahman ElSaid, Steven Benson, Shuchita Patwardhan, David Stadem, and Desell Travis. 2019. Evolving Recurrent Neural Networks for Time Series Data Prediction of Coal Plant Parameters. In *The 22nd International Conference on the Applications of Evolutionary Computation*. Leipzig, Germany.

[15] AbdElRahman ElSaid, Joshua Karnas, Zimeng Lyu, Daniel Krutz, Alexander G Ororbia, and Travis Desell. 2020. Neuro-Evolutionary Transfer Learning Through Structural Adaptation. In *International Conference on the Applications of Evolutionary Computation (Part of EvoStar)*. Springer, 610–625.

[16] AbdElRahman ElSaid, Joshua Karns, Alexander Ororbia II, Daniel Krutz, Zimeng Lyu, and Travis Desell. 2020. Neuroevolutionary Transfer Learning of Deep Recurrent Neural Networks through Network-Aware Adaptation. arXiv:2006.02655 [cs.NE]

[17] AbdElRahman ElSaid, Joshua Karns, Zimeng Lyu, Daniel Krutz, Alexander Ororbia, and Travis Desell. 2020. Improving Neuroevolutionary Transfer Learning of Deep Recurrent Neural Networks through Network-Aware Adaptation. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference* (Cancún, Mexico) *(GECCO '20)*. Association for Computing Machinery, New York, NY, USA, 315–323. https://doi.org/10.1145/3377930.3390193 Best paper nominee.

[18] Aidin Ferdowsi, Samad Ali, Walid Saad, and Narayan B Mandayam. 2019. Cyber-physical security and safety of autonomous connected vehicles: Optimal control meets multi-armed bandit learning. *IEEE Transactions on Communications* 67, 10 (2019), 7228–7244.

[19] Marco Galassi, Nicola Capodieci, Giacomo Cabri, and Letizia Leonardi. 2016. Evolutionary strategies for novelty-based online neuroevolution in swarm robotics. In *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 002026–002032.

[20] David Garlan, S-W Cheng, A-C Huang, Bradley Schmerl, and Peter Steenkiste. 2004. Rainbow: Architecture-based self-adaptation with reusable infrastructure. *Computer* 37, 10 (2004), 46–54.

[21] Thomas J. Glazier, Bradley Schmerl, Javier Cámara, and David Garlan. 2017. *Utility Theory for Self-Adaptive Systems*. Technical Report CMU-ISR-17-119. Carnegie Mellon University Institute for Software Research.

[22] Matt Knudson and Kagan Tumer. 2011. Adaptive navigation for autonomous robots. *Robotics and Autonomous Systems* 59, 6 (2011), 410–420.

[23] J. Kramer and J. Magee. 2007. Self-Managed Systems: an Architectural Challenge. (May 2007), 259–268. https://doi.org/10.1109/FOSE.2007.19

[24] Kunhui Lin, Qiang Lin, Changle Zhou, and Junfeng Yao. 2007. Time series prediction based on linear regression and SVR. In *Third International Conference on Natural Computation (ICNC 2007)*, Vol. 1. IEEE, 688–691.

[25] Jan Hendrik Metzen, Frank Kirchner, Mark Edgington, and Yohannes Kassahun. 2008. Towards efficient online reinforcement learning using neuroevolution. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation*. 1425–1426.

[26] Risto Miikkulainen, Jason Liang, Elliot Meyerson, Aditya Rawal, Dan Fink, Olivier Francon, Bala Raju, Hormoz Shahrzad, Arshak Navruzyan, Nigel Duffy, and Babak Hodjat. 2017. Evolving Deep Neural Networks. *arXiv preprint arXiv:1703.00548* (2017).

[27] Gabriel A Moreno. 2017. *Adaptation Timing in SelfAdaptive Systems*. Ph. D. Dissertation. Carnegie Mellon University.

[28] Gabriel A. Moreno, Javier Cámara, David Garlan, and Mark Klein. 2018. Uncertainty Reduction in Self-Adaptive Systems. In *Proceedings of the 13th International Conference on Software Engineering for Adaptive and Self-Managing Systems* (Gothenburg, Sweden) *(SEAMS '18)*. Association for Computing Machinery, New York, NY, USA, 51–57.

[29] Gabriel A. Moreno, Javier Cámara, David Garlan, and Bradley Schmerl. 2018. Flexible and Efficient Decision-Making for Proactive Latency-Aware Self-Adaptation. *ACM Trans. Auton. Adapt. Syst.* 13, 1, Article 3 (April 2018), 36 pages. https://doi.org/10.1145/3149180

[30] Gabriel A. Moreno, Alessandro V. Papadopoulos, Konstantinos Angelopoulos, Javier Cámara, and Bradley Schmerl. 2017. Comparing Model-based Predictive Approaches to Self-adaptation: CobRA and PLA. In *Proceedings of the 12th International Symposium on Software Engineering for Adaptive and Self-Managing Systems* (Buenos Aires, Argentina) *(SEAMS '17)*. IEEE Press, Piscataway, NJ, USA, 42–53. https://doi.org/10.1109/SEAMS.2017.2

[31] K-R Müller, Alexander J Smola, Gunnar Rätsch, Bernhard Schölkopf, Jens Kohlmorgen, and Vladimir Vapnik. 1997. Predicting time series with support vector machines. In *International Conference on Artificial Neural Networks*. Springer, 999–1004.

[32] Alexander Ororbia, AbdElRahman ElSaid, and Travis Desell. 2019. Investigating Recurrent Neural Network Memory Structures Using Neuro-evolution. In *Proceedings of the Genetic and Evolutionary Computation Conference* (Prague, Czech Republic) *(GECCO '19)*. ACM, New York, NY, USA, 446–455. https://doi.org/10.1145/3321707.3321795

[33] J. Palmerino, Q. Yu, T. Desell, and D. Krutz. 2019. Improving the Decision-Making Process of Self-Adaptive Systems by Accounting for Tactic Volatility. In *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. 949–961. https://doi.org/10.1109/ASE.2019.00092

[34] Gábor Petneházi. 2019. Recurrent Neural Networks for Time Series Forecasting. arXiv:1901.00069 [cs.LG]

[35] Aditya Rawal and Risto Miikkulainen. 2016. Evolving deep LSTM-based memory networks using an information maximization objective. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*. ACM, 501–508.

[36] Sebastian Risi and Julian Togelius. 2015. Neuroevolution in games: State of the art and open challenges. *IEEE Transactions on Computational Intelligence and AI in Games* 9, 1 (2015), 25–41.

[37] Fernando Silva, Luís Correia, and Anders Lyhne Christensen. 2013. Dynamics of neuronal models in online neuroevolution of robotic controllers. In *Portuguese Conference on Artificial Intelligence*. Springer, 90–101.

[38] Fernando Silva, Paulo Urbano, Luís Correia, and Anders Lyhne Christensen. 2015. odNEAT: An algorithm for decentralised online evolution of robotic controllers. *Evolutionary Computation* 23, 3 (2015), 421–449.

[39] Matthew David Smith and Marie-Elisabeth Paté-Cornell. 2017. Cyber Risk Analysis for a Smart Grid: How Smart is Smart Enough? A Multi-Armed Bandit Approach.. In *SG-CRC*. 37–56.

[40] Kenneth Stanley and Risto Miikkulainen. 2002. Evolving neural networks through augmenting topologies. *Evolutionary computation* 10, 2 (2002), 99–127.

[41] Kenneth O Stanley, Jeff Clune, Joel Lehman, and Risto Miikkulainen. 2019. Designing neural networks through neuroevolution. *Nature Machine Intelligence* 1, 1 (2019), 24–35.

[42] Kenneth O Stanley, David B D'Ambrosio, and Jason Gauci. 2009. A hypercube-based encoding for evolving large-scale neural networks. *Artificial life* 15, 2 (2009), 185–212.

[43] Niels LM Van Adrichem, Christian Doerr, and Fernando A Kuipers. 2014. Opennetmon: Network monitoring in openflow software-defined networks. In *2014 IEEE Network Operations and Management Symposium (NOMS)*. IEEE, 1–8.