An Empirical Study of the Naïve REINFORCE Algorithm for Predictive Maintenance

Rajesh Siraskar ([ORCID: 0000-0003-2341-8787](https://orcid.org/0000-0003-2341-8787))1,4, Satish Kumar1,2\*, Shruti Patil1,2, Arunkumar Bongale1, Ketan Kotecha1,2, Ambarish Kulkarni3

1\*Symbiosis Institute of Technology, Symbiosis International (Deemed University), Pune, 412115, Maharashtra, India.

2Symbiosis Centre for Applied Artificial Intelligence, Symbiosis International (Deemed University), Pune, 412115, Maharashtra, India.

3Swinburne University of Technology, Hawthorn VIC 3122, Australia.

4CTO Office, Birlasoft Ltd., Pune, 411057, Maharashtra, India.

\*Corresponding author(s). E-mail(s): [satish.kumar@sitpune.edu.in;](mailto:satish.kumar@sitpune.edu.in) Contributing authors: [rajesh.siraskar@gmail.com;](mailto:rajesh.siraskar@gmail.com)

**Abstract**

Reinforcement Learning (RL) is a biologically inspired, autonomous machine learning method. RL algorithms can help generate optimal predictive maintenance (PdM) policies for complex industrial systems. However, these algorithms are extremely sensitive to hyperparameter tuning and network architecture, and this is where automated RL frameworks (AutoRL) can offer a platform to encourage industrial practitioners to apply RL to their problems. AutoRL applied to PdM has yet to be studied. Aimed at practitioners unfamiliar with complex RL tuning, we undertake an empirical study to understand *untuned* RL algorithms for generating optimal tool replacement policies for milling machines.

We compare a naïve implementation of REINFORCE against the policies of industry-grade implementations of three advanced algorithms – Deep Q-Network (DQN), Advantage Actor-Critic (A2C), and Proximal Policy Optimization (PPO). Our broad goal was to study model performance under four scenarios: (1) simulated tool-wear data, (2) actual tool-wear data (benchmark IEEE*DataPort* PHM Society datasets), (3) univariate state with added noise levels and a random chance of break-down, and finally

(4) complex multivariate state. Across 15 environment variants, REINFORCE models demonstrated higher tool replacement precision 0.687, recall 0.629 and F1 0.609 against A2C (0.449/0.480/0.442), DQN (0.418/0.504/0.374) and PPO (0.472/0.316/0.345), while demonstrating lower variability. Comparing the best *auto-selected* model, over ten training rounds produced unusually wider performance gaps with the REINFORCE precision, recall and F1 at 0.884, 0.884, 0.873 against the best A2C (0.520/0.859/0.639), DQN (0.651/0.937/0.740), and PPO (0.558/0.643/0.580) models. For the REINFORCE, a basic hyperparameter sensitivity and interaction analysis is conducted to better understand the dynamics and present results for the hyperparameters learning rate, discount factor γand the network activation functions (ReLU and Tanh). Our study suggests that, in the untuned state, simpler algorithms like the REIN- FORCE perform reasonably well. For AutoRL frameworks, this research encourages seeking new design approaches to automatically identify optimum algorithm-hyperparameter combinations.

**Keywords:** Reinforcement Learning, predictive maintenance, REINFORCE

# Article Highlights

1. Contributes to the broader goal of AutoRL (Automated Reinforcement Learning) for Predictive Mainte-

nance

1. Research for practitioners with limited hyperparameter tuning expertise, thus focusing on practical indus-

try needs

1. The computational simple REINFORCE outperforms complex algorithms under default configurations

# Introduction

Milling machines are highly versatile, ubiquitous tools serving a variety of industries. A milling machine removes metal from the work piece by rotating and driving a cutting device into it. Abrasive forces cause tool wear, and optimal tool replacement reduces direct costs and optimizes the machines’ downtime. This is an important goal, considering that the global milling machine market is valued at USD 68.3 billion [[1](#_bookmark81)]. The cutting tool experiences multiple types of wear as it cuts through metal. Tool wear depends on several factors such as the cutting speed, force applied to the tool, lubrication and materials of the work piece and cutting tool.

Reinforcement learning (RL) is an artificial intelligence technique inspired by nature. Of all the foundational machine learning (ML) methods, RL is the closest method of learning as seen in animals and humans. Biological learning systems use the learning feedback loop, Fig. [1](#_bookmark0), that inspired core RL algorithms, [[2](#_bookmark82)].

An actor or “agent” interacts with an environment and learns via “trial-and-error”. It acts based on stimuli or feedback received from the environment after performing a certain action. Actions that help in achieving the learning goal receive a reward while actions that do not, are punished. Repeating this loop over thousands of episodes “reinforces” good actions thereby building a “policy” that is optimized for that goal. In the case of predictive maintenance for milling machines, the agent is the “planner” with a goal of learning an optimal tool replacement policy. The environment is technically anything other than the agent – sensors attached to the machine, job specifications, ambient conditions etc.

**State *St***

**RL Agent**

**Reward *Rt***

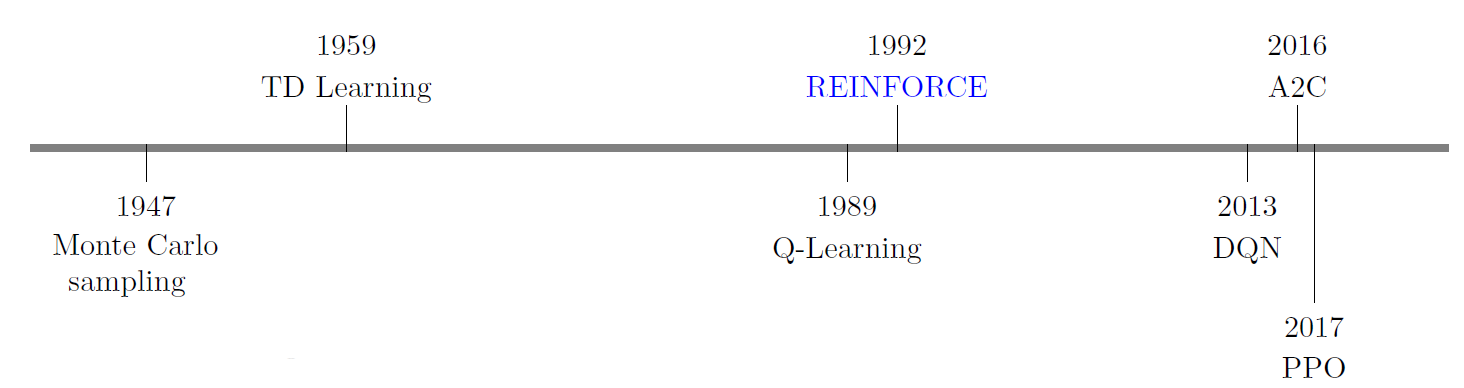
**Action *At***

**Environment**

**Fig. 1** Reinforcement Learning, [[2](#_bookmark82)]

AutoML offers automated machine learning (ML) platforms. Once the user feeds in data and sets an objective, they detect data types of features and response variables, select suitable ML techniques from a library, run techniques like grid-search on the hyperparameter space, train the models, select evaluation metrics, evaluate the models using techniques such as hold-out or cross-validation and finally select the best model. Such automated pipelines are often available for supervised and unsupervised ML fields. AutoML for RL, called AutoRL, is conceptually similar and intended for automating RL development. However RL is extremely sensitive to hyperparameter and network architectures and this prevents practical out-of-the-box use of RL, [[3](#_bookmark83), [4](#_bookmark84)]. Successful RL agents must therefore be manually trained by experts [[5](#_bookmark85)], which unfortunately

restricts the application of these learning methods across the wide industrial domain.



**Fig. 2** Time line of significant RL algorithms

Introduced in 1992, the REINFORCE algorithm [[6](#_bookmark86)] is an early policy-based RL algorithm, capable of handling both discrete and continuous observation and action spaces. In practice the REINFORCE algorithm is considered a “weak” algorithm and superseded by several algorithms developed since. Most notably the deep-neural network version of Q-Learning, the Deep Q-Network (DQN) [[7](#_bookmark87)], followed by Actor-Critic [[8](#_bookmark88)] and one of the most robust modern-day algorithms, the Proximal Policy Optimization (PPO) [[9](#_bookmark89)], Fig. [2](#_bookmark1). Our study takes a fresh look at REINFORCE, an otherwise neglected algorithm comparing it against three advanced algorithms, namely, DQN, Advantage Actor-Critic (A2C), and PPO. Secondly, while most RL studies are evaluated on OpenAI Gym environments, our experiments cover the PdM problem using a custom-built environment. In practice the milling tool is replaced after a set threshold. We use this “deterministic preventive maintenance” policy as the baseline for comparing the various policies. Our systematic evaluation, based on levels of environment difficulty, different bench-mark datasets and varying noise levels allow a broader comparison of the algorithms. Finally, we conduct statistical tests to ensure robust statistical-evidence based conclusion.

The main contributions of this research are:

1. Contributes to the broader goal of AutoRL for predictive maintenance.
2. Targeted toward industrial practitioners not accustomed to RL hyperparameter tuning.
3. Design and implement an RL environment for predictive maintenance of a milling machine.
4. Rigorous evaluation of four standard, untuned, RL algorithms.
5. Research focus on the computationally light REINFORCE algorithm.
6. Use of simple performance evaluation statistical measures and plots that industrial practitioners are normally used to.

The rest of the paper is structured as follows: In the next section we survey some related work and pro- vide the necessary technical background describing the algorithms studied in this research. Sec. [3](#_bookmark14) discusses implementation details of the REINFORCE algorithm and the PdM environment followed by the methodology adopted for training, testing and evaluation. Sec. [4](#_bookmark44) and [5](#_bookmark70) present and discuss, the results of experiments. Finally, we summarize and draw conclusions in Sec. [6](#_bookmark73).

# Related work and background

In this section we survey the intersection of past work related to RL for PdM, specifically milling machines and AutoRL.

## Literature Review

We reviewed related research to establish research gaps that we could address. Our work is meant to assist practitioners apply RL for predictive maintenance. It touches the dual fields of HPO (hyperparameter optimization) and the development of the AutoRL.

Application of RL for PdM has been surveyed by [[10](#_bookmark90)–[12](#_bookmark91)]. There are no research articles that cover

AutoRL for predictive maintenance. We therefore widened our scope and searched for AutoML for PdM

resulting in 21 articles, of which 14 were conference papers and 7 published in journals, Fig. [3](#_bookmark3), however the research coverage remained limited to supervised ML and excluded RL and AutoRL.

Experimental comparison of RL algorithms, has been limited to using standard benchmark OpenAI Gym environments: [[13](#_bookmark92)] documents experimental evaluation of four policy-gradient and actor-critic algorithms PPO, Soft Actor Critic (SAC), Deep Deterministic Policy Gradient (DDPG) and A2C using the Pendulum, Mountain Car, Bipedal Walker, Lunar Landing and Atari 2600 game environments. [[14](#_bookmark93)] evaluate DQN, DoubleDQN, A2C, REINFORCE and PPO using Cartpole, Space Invaders and the Lunar Lander. [[15](#_bookmark94), [16](#_bookmark95)] are significant contributions toward analyzing *real-world* challenges. They performed empirical studies on

Research Articles: AutoML for PdM

8

6

6

5 5

4

3

2

1 1

0

2018 2019 2020 2021 2022 2023 2024 2025

**Fig. 3** Research articles related to AutoML and PdM. None were found on AutoRL for PdM.

Cartpole, humanoid and walker environments[2](#_bookmark4) by embedding them with real-world design concepts and then evaluating REINFORCE, Trust Region Policy Optimization (TRPO) and DDPG.

[[15](#_bookmark94), [17](#_bookmark96)] evaluate RL for continuous control. [[17](#_bookmark96)] study DDPG, ACKTR, TRPO and PPO, using, like we do, the OpenAI RL implementations, on complex MuJoCo[3](#_bookmark5) environments. [[18](#_bookmark97)] is one experimental evaluation we found covering a *real-world* application where DQN, A2C and PPO are applied for choosing the operational radio frequency (RF) mode for a multi-function RF system and go on to recommend PPO as the best.

None of the AutoRL articles covered PdM, we therefore study them for their contribution toward cre-

ating AutoRL frameworks in general. [[5](#_bookmark85)] is a detailed technical assessment and survey of the AutoRL field.

They provide a deep dive into the techniques and methods to automate the RL training pipeline, outline the

implied challenges and the open research questions that remain. They observe that it is the methods that

can *dynamically adapt hyperparameter* configurations during the training cycle that will generate the best

models; however their study concluded that, despite substantial research, it is not yet clear which hyper-

parameters must be optimized dynamically and which statically. A second important conclusion was the

acknowledgment that relation of hyperparameters to environments was as yet undetermined. They align with

[[19](#_bookmark98)], that hyperparameter optimization will vary and be *specific to an environment.*

[[3](#_bookmark83)] are the first to propose a *tabular AutoRL benchmark* for studying the hyperparameters of RL algo-

rithms. They have considered hyperparameter search spaces for five RL methods - PPO, DDPG, A2C, SAC

and TD3, across 22 OpenGym, environments. They studied four hyperparameters of which only two were

common across all five algorithms – the learning rate and discount factor γ. They did not address any

network design hyperparameter.

The main contribution of [[4](#_bookmark84)] is the proposition of a *generalized AutoRL framework* split into three

main components – MDP (Markov Decision Process) modeling, algorithm selection and hyperparameter

optimization.

2From the Real-World Reinforcement Learning (RWRL) Suite

3[MuJoCo](https://mujoco.org/): ‘Multi-Joint dynamics with Contact’ e.g. HalfCheetah, Hopper, Walker and Swimmer

All AutoRL related work [[3](#_bookmark83)–[5](#_bookmark85), [19](#_bookmark98)], point unanimously toward three research gaps: First, unlike AutoML

pipelines, AutoRL as a usable framework has not yet been developed, second there is no ideal hyperparame-

ter configuration setup that is clearly optimal for all or even a set of related environments, and third domain

specific work is required for AutoRL to be effective in solving real world problems.

The work by [[5](#_bookmark85)] mention that as a completely closed-loop system, in RL, each of its component influences

the other. This therefore poses a challenge for automatic tuning as any interim system evaluation (during

training) is highly likely to be stochastic due to the interaction of various noise sources (e.g. policy, environ-

ment). They concluded that no AutoRL approach has been developed to offer the best combination of an

algorithm itself and then its configuration.

The literature survey helped identify the following open areas and helpful pointers for our research:

1. **Predictive maintenance** is an unexplored domain. Build a predictive maintenance RL environment and

train an agent for taking maintenance actions.

1. The **REINFORCE** algorithm has not been studied and evaluated within a PdM environment.
2. A comparison of **untuned** RL methods may enhance our understanding of how best to create automated

techniques to address the interaction between algorithm and hyperparameter settings.

1. A good selection of **common hyperparameters** across the algorithms are the learning rate and discount

factor γ.

1. None of the AutoRL comparative studies covered network architecture elements - consider covering at

least one important **network hyperparameter**.

Our work focuses on RL. Supervised machine learning can be used to solve such problems i.e. detect when

the fault part needs replacement. As a final part of this existing literature study, we briefly table related ML

work in Table [1](#_bookmark6) that we can compare to, later in the Discussion Sec. [5](#_bookmark70). We see that Support Vector Machine

(SVM), a classification technique, is the most popular method. Research by [[20](#_bookmark99)–[22](#_bookmark101)], is related to finding

*regions* of the fault signal, using *attention* based ML techniques from the the vision and text fields. Finally,

Research by [[22](#_bookmark101)], covers the same PdM problem and datasets we use in this research, and applies a very

light and classical ML technique, the Nadaraya–Watson kernel regression, as the attention mechanism, [[23](#_bookmark102)].

## Technical Background

### Key concepts of RL

A task is a goal we set for our agent to learn. In our case the agent must learn to optimally predict the replacement of the tool. Frequent tool replacements increase down-time while delaying it results in inferior

**Table 1** ML based solutions for similar predictive maintenance problem.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |
| ML technique | Time series | Research article | Accuracy | Domain - use-case |
| Support Vector Machine | Time domain, multivariate | [Niu et al.](#_bookmark103) [[24](#_bookmark103)] | 0.967 | Milling tool-wear condition monitoring |
| k-Nearest Neighbour | Time-domain with wavelet transforms | [Yang et al.](#_bookmark104) [[25](#_bookmark104)] | 0.948 |  |
| Support Vector Machine | Time-domain with wavelet transforms | [Lei et al.](#_bookmark105) [[26](#_bookmark105)] | 0.908 |  |
| Support Vector Machine | Time and frequency domain | [Zhou et al.](#_bookmark106) [[27](#_bookmark106)] | 0.908 |  |
| Support Vector Machine | Time domain, multivariate | [Ou et al.](#_bookmark107) [[28](#_bookmark107)] | 0.918 |  |
| Naive Bayes and k-Star | Time domain, multivariate | [Madhusudana et al.](#_bookmark108) [[29](#_bookmark108)] | 0.969 |  |
| Tree family classifiers | Time domain, multivariate | [Patange et al.](#_bookmark109) [[30](#_bookmark109)] | 0.890 |  |
| Calibration model (reference signal based) | Time domain, multivariate | [Liu et al.](#_bookmark110) [[31](#_bookmark110)] | 0.900 |  |
| CNN with Attention Knowledge Distillation | Time domain, multivariate | [Fu et al.](#_bookmark99) [[20](#_bookmark99)] | 0.972 | Bearings: Classification of faults |
| CNN based LeNet | Time domain, multivariate | [Pandey et al.](#_bookmark100) [[21](#_bookmark100)] | 0.953 |  |
| Nadaraya-Watson Attention mechanism | Time domain, univariate | [Siraskar et al.](#_bookmark101) [[22](#_bookmark101)] | 0.984 | Milling tool-wear fault detection. |
| Nadaraya-Watson Attention mechanism | Time domain, multivariate | [Siraskar et al.](#_bookmark101) [[22](#_bookmark101)] | 0.834 | (Performance in *R*2) |

work piece quality. In Fig. [1](#_bookmark0) the agent interacts with the environment by performing an action (a ∈ A), which then alters the state of the environment to one of many states (s ∈ S). The resulting state is determined by state-transition probabilities (P) as governed by Markov Decision Process (MDP) theory. The new state provides feedback via a reward (r ∈ R). Higher positive rewards “reinforce” good behavior. Performing this over thousands of episodes with the objective of maximizing the total rewards R, enables the agent to develop a policy π which is essentially a mapping of the optimal action to perform given a certain state.

A **value function** computes how good a state or an action is by predicting future rewards, also known as a "return" facilitates discounting i.e. applying less weight to future rewards. Value functions can be represented by **state-value** of a state , as the expected return: ; or an **action-value** function of a state-action pair as . With this brief overview of RL, we look at the core ideas of the four algorithms we experimented with.

### Deep Q-Network (DQN)

Deep Q-Network [[7](#_bookmark87)] significantly improved the earliest RL algorithm, Q-learning, by introducing neural net- works to learn policies for high-dimension environments with two novel strategies to significantly stabilize

learning – an “experience replay buffer” and a target network that was frozen and only periodically updated. Equation ([1](#_bookmark7)) shows the DQN loss function where D is the replay memory, sampled using a uniform distribution parameterized with , helps compute the Q values and represents parameters of the frozen target Q-network. The objective then, is to minimize this loss.

### Advantage Actor Critic (A2C)

A2C is a variant of Asynchronous Advantage Actor Critic (A3C) [[8](#_bookmark88)], and uses multiple computation workers to avoid the use of a replay buffer. A2C is a policy-gradient actor-critic algorithm. Policy-gradient algorithms strive to model and optimize the policy directly. Actor-critic structures consist of two networks – a critic that updates function parameters w of the value function (i.e. either or ; and an actor that updates the policy parameters for , following the direction computed by critic. Actors therefore learn the parameterized policy π*θ* using the policy-gradient as shown in ([2](#_bookmark8)).

Where the advantage function measures how good or bad the action is w.r.t. policy's average, for a particular state, using (3).

### Proximal Policy Optimization (PPO)

[[9](#_bookmark89)] formulated PPO which is often considered as the most robust of the RL algorithms. PPO is a policy- gradient method based on TRPO (Trust Region Policy Optimization) by [[32](#_bookmark111)], where the core idea is the use of a trust region to improve training stability by avoiding updates to parameters that vastly change the policy at a particular time step. TRPO ensures this by using a divergence constraint on the magnitude of policy update. If ([4](#_bookmark10)) represents the ratio of probabilities between policies of previous and current iteration, then the objective function of TRPO is given by ([5](#_bookmark11)), where represents the estimated advantage function.

PPO extends TRPO by additionally imposing a regional constraint. It prevents large updates by forcing the ratio to stay within a small interval , around 1.0, by use of a hyperparameter .

### REINFORCE

The REINFORCE algorithm, invented by [[6](#_bookmark86)], directly learns a policy to produce action probabilities from states. True to the fundamental RL concept, actions that cause favorable states, are positively reinforced thereby increasing their probability of occurrence, while those resulting in unfavorable states are penalized. Equation ([7](#_bookmark12)) is the objective function the agent maximizes and is defined as the expected return over many trajectories sampled from the policy. Comparing this to the objective functions of the more advanced algorithms above, we see just how very simple it indeed is.

REINFORCE uses policy gradient ([8](#_bookmark13)) to update the action probabilities.

# Methodology

We undertook this systematic empirical after the preliminary encouraging results of REINFORCE. Our

study broadly follows the exploratory research methodology outlined in [[33](#_bookmark112)–[35](#_bookmark113)], and is based on the original

*empirical cycle* proposed by Adriaan D De Groot, [[36](#_bookmark114)]. Exploratory research approach is often used to arrive

at a broad hypothesis [[37](#_bookmark115)]. We built an experimentation set-up, collected data, systematically analyzed it

and tabled our observations, finally employing abductive reasoning to present some reasons for the behavior

we observed.

## Implementation details

RL requires an environment to function. We built a custom “milling” environment for the agent to learn a policy for tool replacement. Fig. [4](#_bookmark15) shows the stepped approach used to generate the fifteen environments,

**Table 2** Our research method steps, mapped to the standard empirical cycle of [[36](#_bookmark114)]

**Empirical**

**cycle phase**

**Our process step**

**Reference**

Observation Preliminary experiments demonstrated good results for the REINFORCE on the PdM problem.

Induction Formulate a high-level hypothesis based on the preliminary observations. Use model performance metrics - precision recall and F1-score.

Deduction Design and build an experimentation set-up. Design a set of experiments to test the hypothesis.

*H*0: *µRF − µAA* = 0*. Ha*: *µRF − µAA >* 0

Implementation Sec. [4.6](#_bookmark59)

Testing Conduct the experiments and collect data. Experiment results (Sec.

[4](#_bookmark44)), Sensitivity analysis

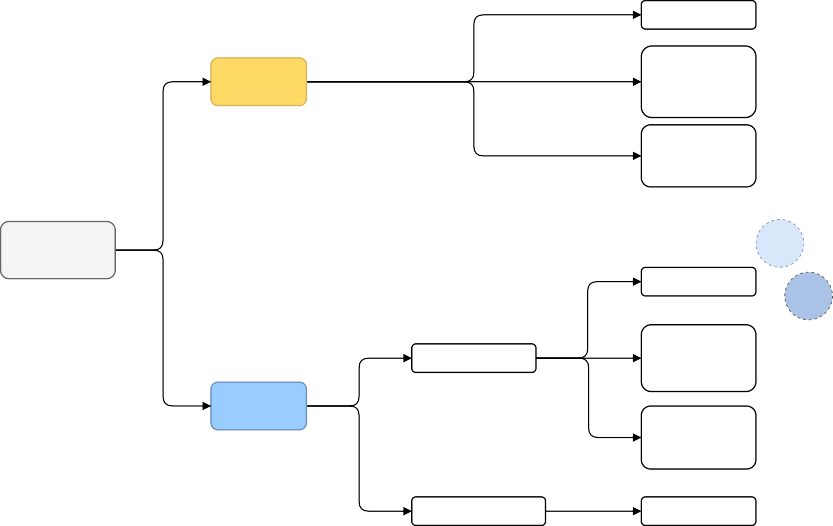
(Sec. [4.7](#_bookmark64))

Evaluation Evaluate and present the test results. Present the interpretations based on abductive reasoning.

Results and discussion sections [4](#_bookmark44), [5](#_bookmark70)

of varying complexity. We first describe the simulation-based environment followed by the real-data based family, describing the child-variants along the way.

Increasing difficulty level



**Dasic**

No noise

**Simulated**

Low noise and low chance of breakdown

High noise and high chance of breakdown

**Environments**

**PHM C-01**

No noise

Uni-variate state

**PHM C-04**

Low noise and **PHM**

low chance of **C-06**

breakdown

**Real data**

High noise and high chance of breakdown

Multi-variate state

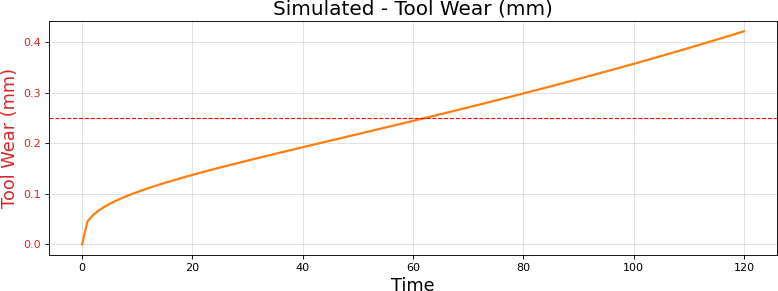
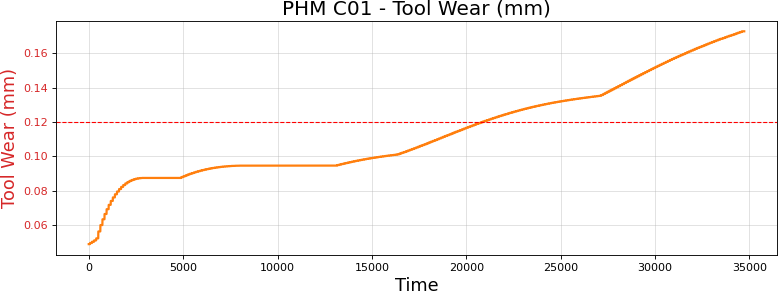
No noise



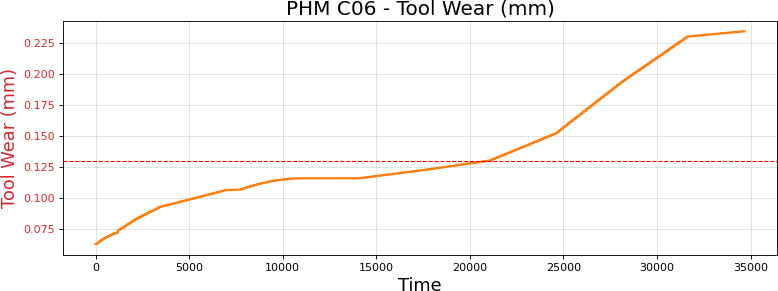
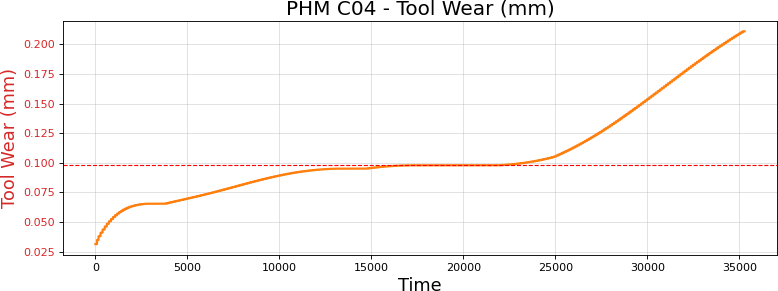
**Fig. 4** The fifteen different environments used for evaluation.

### Simulating tool wear

[[38](#_bookmark116)] provides a parameterized power-exponential function for modeling tool wear ([9](#_bookmark16)), where V B represents the flank wear in mm.

(a) Simulated wear data (b) PHM C01 wear data



(c) PHM C04 wear data (d) PHM C06 wear data

**Fig. 5** Tool wear data. Red dotted line indicates the wear threshold beyond which tool is replaced.

We used the parameter values provided in the paper a = 0.08257, b1 = 0.3342 and b2 = 0.03147 to simulate 120 data points. Fig. [5(a)](#_bookmark17) shows the tool wear simulated using ([9](#_bookmark16)), with the red dotted-line indicating the wear threshold beyond which tool is replaced. This provided the mechanism to simulate the basic variant of tool-wear based **univariate state**. We then added two further levels of increased complexity using noise and a chance of breakdown, giving us three distinct environments.

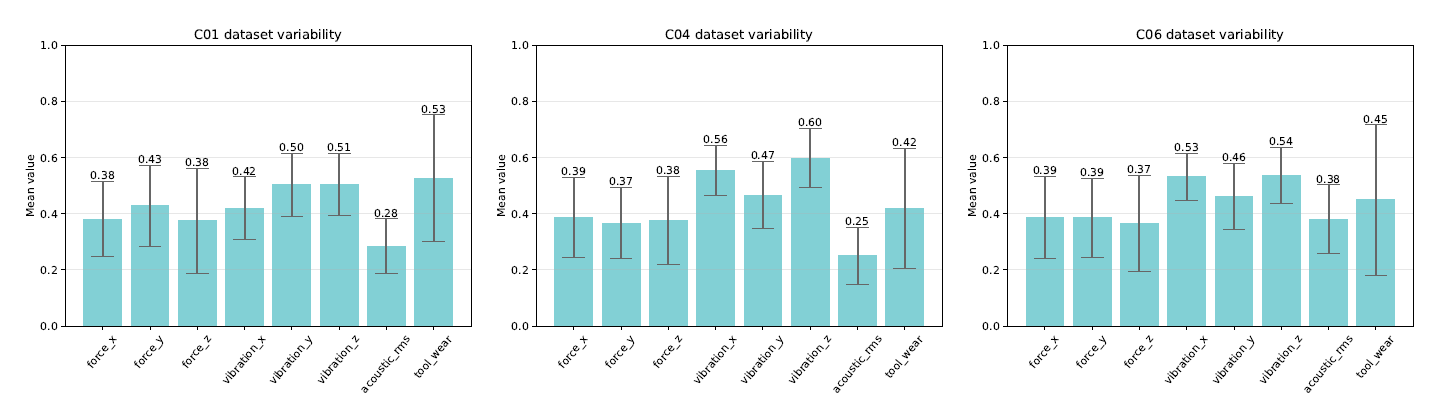
### Actual tool wear data

The IEEE*DataPort* hosts the “2010 PHM Society” tool-wear data obtained from a high-speed CNC milling machine, [[39](#_bookmark117)]. C01, C04 and C06 datasets are suggested as benchmarks to be used for machine learning research and were the ones we used[5](#_bookmark22). The data is from seven sensors – dynamometer measuring forces in X, Y and Z dimensions, measured in N; accelerometer measuring vibration in X, Y and Z dimensions, measured in g and finally acoustic emission data as AE-RMS, in V. A separate file contains tool wear data in mm. Figures [5(b)](#_bookmark18), [5(c)](#_bookmark19) and [5(d)](#_bookmark20) show the tool wear for the three datasets. We use real data to create two state designs, an univariate state consisting of only the tool wear and a **multivariate** state designed using all the additional seven sensor values. Just as we did for the simulated case, the complexity of the univariate state is increased using two levels of noise and break-down parameters. The multivariate state is complex in itself. Fig. [6](#_bookmark23) shows a comparison of the dataset features along with the intrinsic variability. We selected these

datasets for this inherent variability and this shows up as variation in performance across the datasets, as

5In the article, we often refer to these datasets as “PHM-” data.

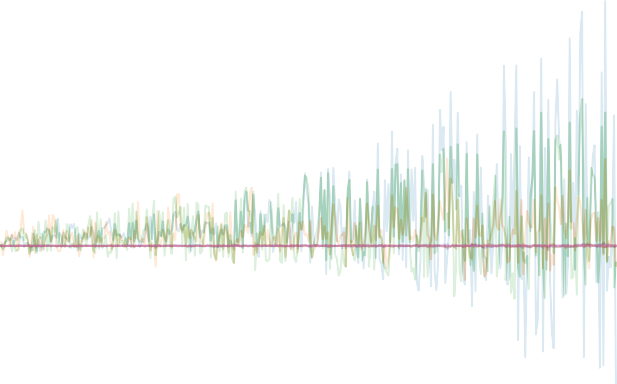
we shall see in Sec. [4](#_bookmark44) later. An indicative plot of PHM-C06 feature data, Fig. [7](#_bookmark24), demonstrates the inherent feature complexity and we therefore use the natural (without noise) form.



1. PHM-C01 dataset (b) PHM-C04 dataset (c) PHM-C06 dataset

**Fig. 6** Feature variability across datasets. Values are normalized to visualize relative variability.

125



force\_x force\_y force\_z vibration\_x vibration\_y vibration\_z

acoustic\_emission\_rms tool\_wear

100

75

50

25

0

25

50

75

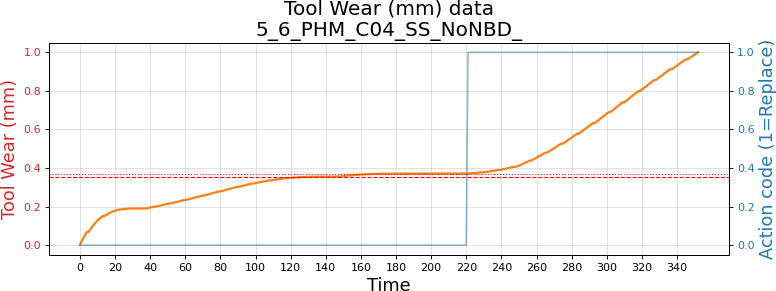
0 5000 10000 15000 20000 25000 30000 35000

**Fig. 7** Indicative plot of PHM-C06 multivariate data, showcasing inherent feature complexity. Y-axis indicate magnitude of sensor values. X-axis is number of data-points.

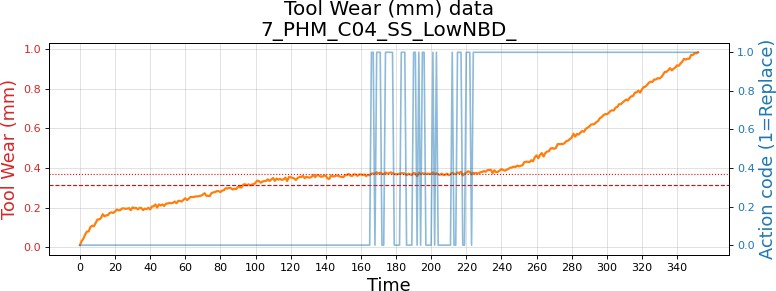
In practice, state variables are often normalized to improve stability and convergence. Both the simulated and real data was normalized using min-max scaling such that the tool wear and other state features, . We will see next how this allows adding white noise of similar magnitudes across different PHM datasets.

### Adding noise and chance of break-down

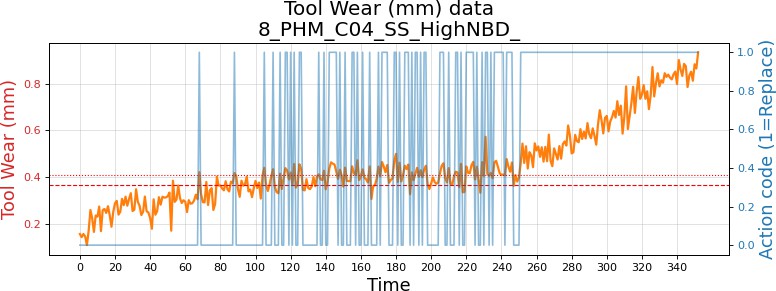
Study of noise in RL settings is of prime importance, as suggested by [[19](#_bookmark98)]. Fig. [8](#_bookmark28) shows the effect of adding two levels of noise; a “low” level by adding Gaussian noise of order −3, (0.0, 0.001] and “high” of order



* 1. No noise



* 1. Low noise



* 1. High noise

**Fig. 8** PHM C04 tool wear data (normalized) and the effect of noise. Blue line is the replacement action decision.

−2, (0.0, 0.01]. Since the tool wear is less than 0.24 mm, this adds significant perturbations as seen in Fig. [8(b)](#_bookmark26) and [8(c)](#_bookmark27). The noise affects the tool replacement decision (solid blue line) around the replacement threshold (dotted red line). The *human* preventive maintenance policy replaces the tool if the wear exceeds the threshold and this decision boundary oscillates due to the noise. One can see that in the case of no noise (Fig. [8(a)](#_bookmark25)), the decision boundary is clean.

Break down occurs due to excessive tool use and can often occur randomly. In conjunction with Guassian noise this complexity is added for the univariate state based environments. For the low-noise variant we add a low 5% chance of break down and for the high noise variant we add a higher chance of 10%. The “milling” episode is terminated if a probability, sampled from a uniform distribution is less than this “chance” threshold.

**Table 3** List of the fifteen environments and their categorization.

|  |  |  |
| --- | --- | --- |
| Environment variant | Noise factor | Breakdown chance |
| **Simulated** |  |  |
| 1 Simulated - No noise | None | None |
| 2 Simulated - Low noise | 1e-3 | 0.05 |
| 3 Simulated - High noise | 1e-2 | 0.10 |
| **Real data – simple univariate** |  |  |
| 4 PHM C01 SS (simple, univariate) - No noise | None | None |
| 5 PHM C01 SS (simple, univariate) - Low noise | 1e-3 | 0.05 |
| 6 PHM C01 SS (simple, univariate) - High noise | 1e-2 | 0.10 |
| 7 PHM C04 SS (simple, univariate) - No noise | None | None |
| 8 PHM C04 SS (simple, univariate) - Low noise | 1e-3 | 0.05 |
| 9 PHM C04 SS (simple, univariate) - High noise | 1e-2 | 0.10 |
| 10 PHM C06 SS (simple, univariate) - No noise | None | None |
| 11 PHM C06 SS (simple, univariate) - Low noise | 1e-3 | 0.05 |
| 12 PHM C06 SS (simple, univariate - High noise | 1e-2 | 0.10 |
| **Real data – complex multivariate** |  |  |
| 13 PHM C01 MS (complex, multivariate) - No noise | None | None |
| 14 PHM C04 MS (complex, multivariate) - No noise | None | None |
| 15 PHM C06 MS (complex, multivariate) - No noise | None | None |

Table [3](#_bookmark29) summarizes the 15 environment variants and their three logical groups: (1) Simulated 1-3 (2) Real data – simple univariate environment (4-12) and Real data – complex multivariate (13-15).

### Tool wear as a Markov Decision Processes (MDP)

Formulating our problem to be solved by RL requires us to assume that the wear process satisfies the Markov property – which implies that the transition of tool wear to another state is dependent only on the current state and not on any previous states. MDPs are defined by the tuple . We will define the elements of state space , action space and reward function , next.

### State and environment elements

There are two basic state definitions across the 15 environment variants, the “simple univariate” and the “complex multivariate”.

The elements of simple univariate state vector are , where is the current tool wear. As part of the environment other elements that are sensed by the agent are [], where is the wear threshold, is the noise factor and is one of [0, 1e-3, 1e-2]s the chance of tool breakdown and is one of [0, 0.05, 0.10]. The complex multivariate state where, as mentioned in Sec. [3.1](#_bookmark21), () represents the force along the 3 axes, similarly represents the vibration, the acoustic emission.

### Actions

The action space is binary, A ∈ [0, 1], where 0 (CONTINUE) represents continuation of milling operation and 1 (REPLACE TOOL) represents the action of replacing the tool.

### Environment feedback

“Feedback” generated by the action an agent takes, is the central mechanism by which agents learn. This is implemented via the step() function in the environment code and is outlined in Algorithm [1](#_bookmark31). At every time step an action A*t* is taken and the resulting state is evaluated for terminating conditions or assigning a reward and continuing.

### Reward function

In the reward function ([10](#_bookmark30)) there are three distinct phases where a reward (or penalty) is offered as feedback. and are constants that determine the magnitude of reward. When the current state of wear is less than the threshold we have a desirable condition, and we award the agent a positive reward. The formulation ([10](#_bookmark30)) allows a higher reward to be collected the closer it is to threshold so as to maximize tool usage, but not allowing it to cross it. If it does, the agent is penalized (negative reward) by a magnitude of , and once again the farther away it is from the threshold i.e. a highly deteriorated tool the larger the penalty. To avoid this “bad” state, the agent must learn to replace the tool; represented by the third condition. Tool replacement implies a direct cost (that of the tool) and a much higher and significant downtime “cost”. To ensure the agent does not learn to replace unnecessarily, we “penalize” it. It is important to note that the last condition in ([10](#_bookmark30)) is an “incremental addition”, the first two conditions are mutually exclusive and evaluated first, followed by the last condition which is incrementally added on whatever reward is collected in the previous condition. The agent then tries to balance these three conditions, such that it maximizes its total return, over time.

Of the final two elements of the MDP quintuple, represents the probability transition matrix and is

usually not known and we will therefore use *model-free* RL techniques to learn that from “experiences”; enables the agent to learn long-term impact of its actions i.e. what is the long term impact of replacing the tool now or that of delaying the replacement, is set to 0.99 to facilitate this farsightedness.

**Algorithm 1** Agent class step() method – Reward handling mechanism

1: **procedure** Step class method: Reward handling

**Class attributes:** Wear-threshold W*τ* ; reward accumulated so far Reward; reward function parameters R1, R2 and R3; random chance of breakdown P*bd*; tool replacements made so far tool replacements; maximum allowable operations T

**Input:** Current length of episode t; current tool-wear w*t*; current policy action A*t*; current index of training data data index

2: **Initialize** p from uniform probability distribution

3: **if** t ≥ T **then**:

4: ▷ Termination based on length of episode.

5: Terminate ← True

6: Reward ← 0.0

7: data index ← 0 ▷ Reset the training data-frame index to the beginning.

8: **else if** w*t* > W*τ* **and** p < P*bd* **then**:

9: ▷ Termination based on chance of breakdown.

10: Terminate ← True

11: Reward ← 0.0

12: data index ← 0

13: **else**

14: Terminate ← False

15: **if** w*t* < W*τ* **then**:

16: ▷ Healthy tool

17: Reward ← Reward + t · R1

18: **else**

19: ▷ Tool deteriorating

20: Reward ← Reward − t · R2

21: **end if**

22: **if** A*t* is REPLACE\_TOOL **then**:

23: ▷ Tool is replaced

24: Reward ← Reward − R3

25: data index ← 0 ▷ Tool replaced, therefore roll back tool life

26: Increment tool replacements

27: **end if**

28: **end if**

29: **end procedure**

### Network architecture and basic hyperparameters

Stable-Baselines3 (SB3) provides robust open-source implementations of many RL algorithms, [[40](#_bookmark118)]. As of 20- Sep-2024, 13 algorithms have been implemented however REINFORCE has *not* been implemented, [[41](#_bookmark119)]. For this research we use the *default* SB3 implementations of DQN, A2C and PPO and compare its performance to a custom implementation of the REINFORCE algorithm using a very simple network architecture. Table [4](#_bookmark33) shows a comparison of the architecture and the basic common hyperparameters.

The REINFORCE uses a *single* internal layer of 64 units. The default architecture for all three SB3 algorithms (A2C/DQN/PPO) consists of *two* fully connected layers with 64 units per layer [[42](#_bookmark120)]. While

**Table 4** Comparing the network architecture and basic hyper-parameters across algorithms.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **A2C** | **DQN** | **PPO** | **REINFORCE** |
| Network | input dim x | input dim x | input dim x | input dim x |
| architecture | [64*|*Tanh x  64*|*Tanh] x output dim | [64*|*Tanh x  64*|*Tanh] x output dim | [64*|*Tanh x  64*|*Tanh] x output dim | [64*|*ReLU]  x output dim |
| Layers | 2 | 2 | 2 | 1 |
| Units | 64 x 64 | 64 x 64 | 64 x 64 | 64 |
| Activation | Tanh, Tanh | Tanh, Tanh | Tanh, Tanh | ReLU |
| Optimizer | RMSprop | Adam | Adam | Adam |
| Learning rate | 0.0007 | 0.0001 | 0.0003 | 0.01 |
| Gamma | 0.99 | 0.99 | 0.99 | 0.99 |

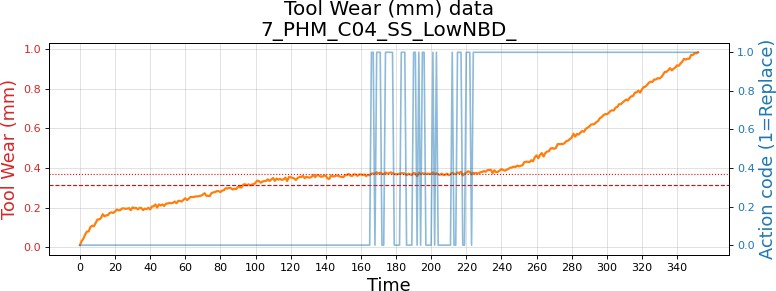
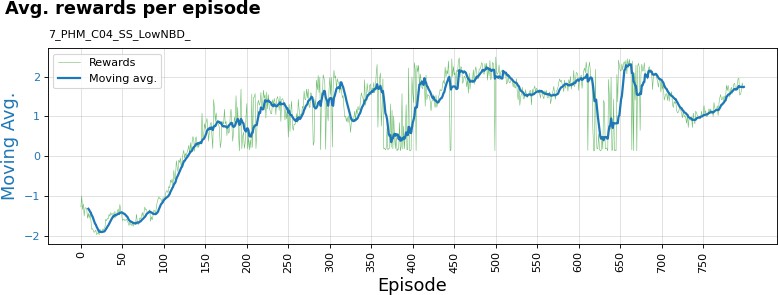
REINFORCE used ReLU (rectified linear unit) as the activation function, the other three algorithms used hyperbolic tangent (Tanh). Finally, REINFORCE’s learning-rate is much larger.

## Training

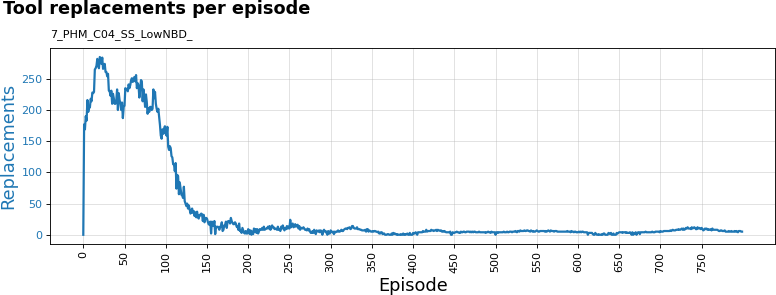
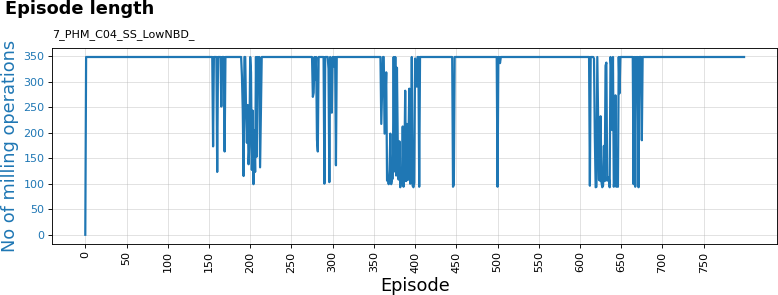
The training strategy must ensure a uniform comparison of the algorithms. We maintained the exact *same* environment variant, wear dataset, noise parameter, probability of breakdown parameter, and the three reward function parameters; across all four algorithms during a single training round. As the wear data is time-series data, the training and test sets are created by systematic sampling. Simulated data and real tool wear data (PHM) was randomly sampled at a certain frequency and down sampled into training and test sets. The REINFORCE was trained for 800 episodes for the simulated and PHM univariate variants, for all three noise and breakdown levels (none, low and high) – Table [3](#_bookmark29) items 1-12. For the PHM multivariate variant, Table [3](#_bookmark29) items 13-15, REINFORCE was trained for 1000 episodes. SB3 algorithms were trained for 10,000 episodes for all variants. We ran ten rounds of training, tested each generated model and averaged

results over the 10 rounds. Testing is explained in the next section while results are presented in Sec. [4](#_bookmark44).

Fig. [9](#_bookmark37) shows the training plots for the algorithm of our interest – REINFORCE. It displays how the wear plot looked for C04 with low noise and low chance of breakdown settings. The average rewards increase over the course of 800 episodes (Fig. [9(b)](#_bookmark34)). The episode length (Fig. [9(c)](#_bookmark35)) demonstrates the complexity introduced by random breakdown (which abruptly terminates the episode). It is the tool replacement policy that is of interest to the industrial practitioner – Fig. [9(d)](#_bookmark36) shows that it decreases to optimal levels as the agent learns over time. Similarly, Fig. [10](#_bookmark38) demonstrates the training for the PHM C06 dataset affected by high noise and higher breakdown probability. Finally, for the more complex multivariate state variant, the training plots are as seen in Fig. [11](#_bookmark40); as we do not introduce noise or breakdown here, the episodes are always completed (Fig. [11(c)](#_bookmark39)).

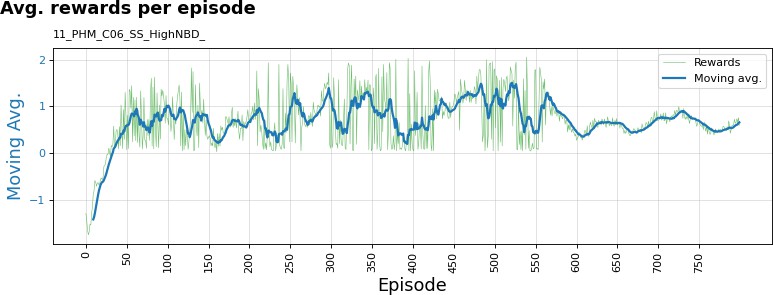
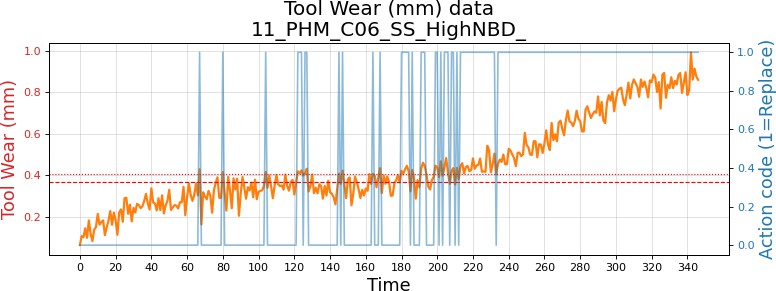
 

(a) PHM C04 wear data (b) Average rewards per episode

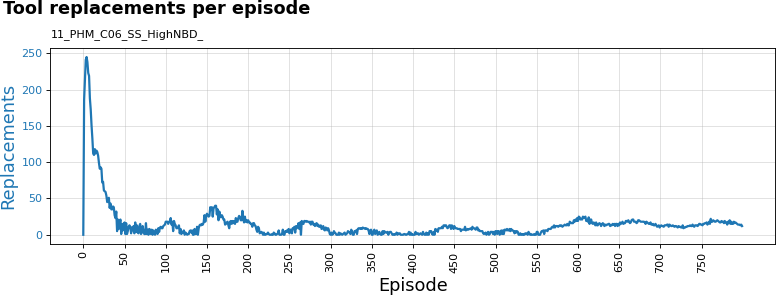
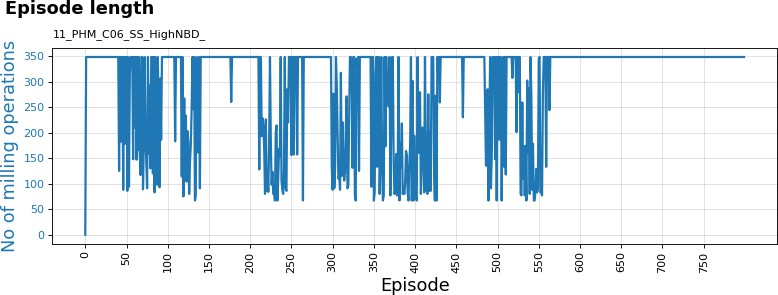


(c) Episode length completed per episode (d) Tool replacements per episode

**Fig. 9** Training plots of REINFORCE. Dataset: PHM-C04. Variant: Univariate state, low-noise and low chance of breakdown.



(a) PHM C06 wear data (b) Average rewards per episode

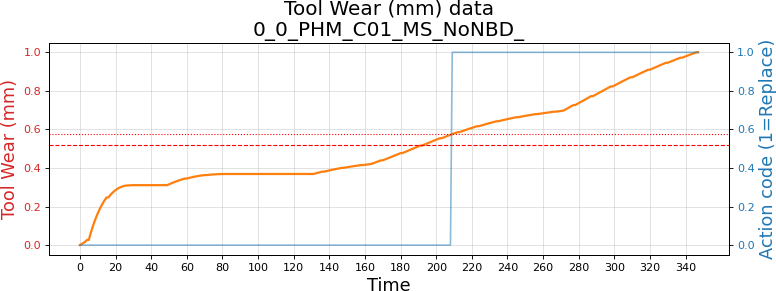
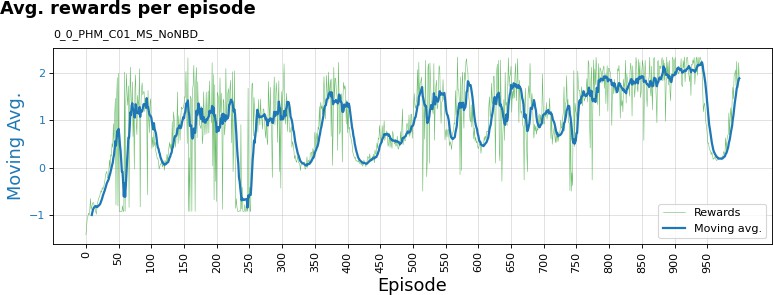


(c) Episode length completed per episode (d) Tool replacements per episode

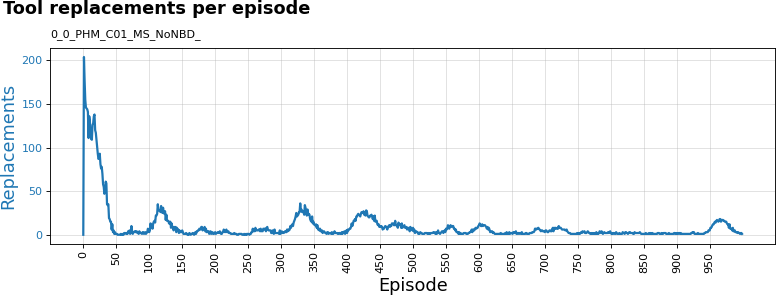
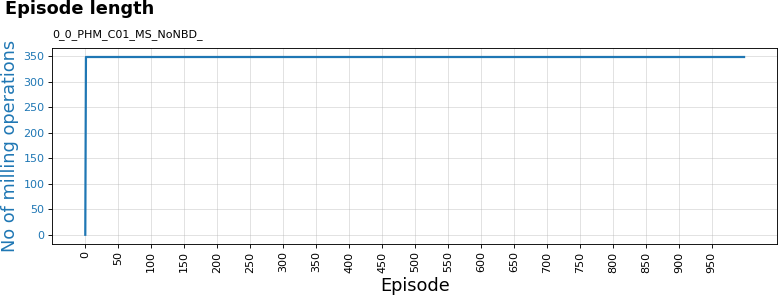
**Fig. 10** Training plots of REINFORCE. Dataset: PHM-C06. Variant: Univariate state, high-noise and high chance of break- down.

## Testing and performance evaluation

Testing was performed with data separate from the training data. 10 rounds of testing are performed, with a *new* set of 40 test cases randomly sampled and frozen across all four algorithms, during each round.

(a) PHM C01 wear data (b) Average rewards per episode



1. Episode length completed per episode (d) Tool replacements per episode

**Fig. 11** Training plots of REINFORCE. Dataset: PHM-C01. Variant: Multivariate state, No noise or breakdown.

### Evaluation metrics

The human decision is based on “preventive maintenance” – replace the tool against a predefined wear threshold. In our data a tool replacement is represented as 1 and a normal operation as 0. We applied classification metrics to evaluate the RL agent decisions of tool replacement. It is worth noting that while we have selected an arbitrary wear threshold as this serves our purpose for algorithm comparison; in reality the threshold is based on several factors like materials of tool and work-piece, the duration of continuous operation, ambient conditions, “cost” of production downtime etc. Threshold could therefore vary significantly from one case to another.

##### Human decision

Replace tool Continue milling

Replace tool T P FP

##### Agent decision

Continue milling FN T N

**Fig. 12** Confusion matrix: Human versus Agent decisions

Classification metrics are based on the confusion matrix shown in Fig. [12](#_bookmark41). T P represents true positive cases, where both the agent and human agree on replacing the tool. False positive FP cases denote the agent falsely suggesting replacements, while false negatives FN are cases where continuation of milling is suggested, when in fact a tool replacement would have helped. Precision (Pr), Recall (Rc) and F1-score metrics can then be computed as shown in ([11](#_bookmark42)).

For a classification problem, the choice of precision versus recall will depend on the industry and appli-

cation. Timely replacements, T P, ensure work piece quality while still maintaining minimal production

downtime. Precision is driven by lower FP s i.e. reducing unnecessary replacements. A high recall is necessary

in several critical industrial environments, where failing to replace the tool can lead to serious consequences,

demanding lower FN s. Depending on the industrial application one can use a *weighted* F1-score equation

([12](#_bookmark43)) that is oriented toward precision or recall. For high precision applications set a β < 1.0; for applications

demanding higher recall, set a β > 1.0. For this research we set β to 0.5 and suggest setting β to 2.0 for

applications where recall is critical.

Throughout the article we use 95% confidence intervals (CI) to quantify the uncertainty of the metrics

and provide an indication of the robustness. In the Results Sec. [4](#_bookmark44), these are presented in the results tables

(columns titled ‘95% CI’) and as error-bars in the accompanying plots.

# Results

We present the summarized results in this section accompanied by commentary referring to detailed results made available in Appendix [A](#_bookmark74). Since there are several tables and figures, for reference, we created a cross- linked Table [5](#_bookmark45). Generally, blue text is used in tables to highlight prominent values. Summary performance and F*β*0.5 plots accompany tables to assist in visualizing the comparative performance. The F*β*0.5 plots show *individual* performance over ten rounds of training-validation and help infer training stability. Similar plots for precision, recall and F1 have been relegated to Appendix [A](#_bookmark74).

## Overall performance

At an aggregated level, Table [6](#_bookmark46) along with Fig. [13](#_bookmark47), show that the REINFORCE scores better than the other algorithms, on all four metrics. Tool replacement precision at 0.687, is highest of the four algorithms and better by 0.215 in absolute terms when compared to the next best, PPO. The standard deviation for precision is the lowest at 0.059. On recall, F1 and F*β*0.5, REINFORCE is better by 0.125, 0.168 and 0.195 compared to the next best.

These metrics were averaged over 10 rounds of training followed by validation. We visualize the behavior of our primary metric F*β*0.5, *individually* across the ten rounds in Fig. [14](#_bookmark48). The blue line floating above the

**Table 5** Reference table for results.

Item Reference

**Summary Results**

1. Overall summary – All 15 environments, averaged over 10 rounds Table [6](#_bookmark46), Fig. [13](#_bookmark47) –”– F*β*0.5 behavior plot over 10 rounds Fig. [14](#_bookmark48)
2. Simple univariate state – Simulated data, 3 variants (3 noise settings), averaged over 10 rounds

Table [7](#_bookmark49), Fig. [15](#_bookmark50)

–”– F*β*0.5 behavior plot over 10 rounds Fig. [16](#_bookmark51)

1. Simple univariate state – Real data, 9 variants (3 datasets *×* 3 noise settings), averaged over 10 rounds

Table [8](#_bookmark52), Fig. [17](#_bookmark53)

–”– F*β*0.5 behavior plot over 10 rounds Fig. [18](#_bookmark55)

1. Complex multivariate state – Real data, 3 variants (3 datasets), averaged over 10 rounds Table [9](#_bookmark54), Fig. [19](#_bookmark56) –”– F*β*0.5 behavior plot over 10 rounds Fig. [20](#_bookmark57)

**Super Models**

1. Super Models – Best model, *averaged* over all 15 environments Table [10](#_bookmark58) Fig. [21](#_bookmark60)

Super Models – Best model, *details* of all 15 environments Appendix [A](#_bookmark74) - Table [A3](#_bookmark76)

**Hypothesis tests and Training time results**

1. Hypothesis tests – p-value and t-statistic of REINFORCE versus other algorithms Table [11](#_bookmark63)
2. Training time – Averaged over 10 rounds and all 15 variants Fig. [22](#_bookmark65)

**Detailed Results** - Appendix [A](#_bookmark74)

1. Complete detailed results – All 15 environments, averaged over 10 rounds Appendix [A](#_bookmark74) - Table [A2](#_bookmark75)
2. Stability behavior plots over 10 rounds, Precision, Recall and F1

Overall (all variants) Appendix [A](#_bookmark74) - Fig. [A1](#_bookmark77)

Simulated state Appendix [A](#_bookmark74) - Fig. [A2](#_bookmark78)

Simple univariate state Appendix [A](#_bookmark74) - Fig. [A3](#_bookmark79)

Complex multivariate state Appendix [A](#_bookmark74) - Fig. [A4](#_bookmark80)

**Table 6** Model performance summary - averaged over all environments.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | Precision |  |  |  | Recall |  |  |  | F1-score |  |  |  | F*β*0.5 |  |
|  | Mean | SD | 95% CI |  | Mean | SD | 95% CI |  | Mean | SD | 95% CI |  | Mean | SD | 95% CI |

(0.443,

|  |  |  |
| --- | --- | --- |
| A2C | 0.449 | 0.088 |
| DQN | 0.418 | 0.185 |
| PPO | 0.472 | 0.144 |
| REINFORCE | 0.687 | 0.059 |

0.455)

(0.406,

0.430)

(0.463,

0.482)

(0.683,

0.691)

0.480 0.084 (0.474,

0.485)

0.504 0.032 (0.502,

0.506)

0.316 0.087 (0.310,

0.321)

0.629 0.051 (0.626,

0.633)

0.442 0.070 (0.437,

0.446)

0.374 0.035 (0.372,

0.376)

0.345 0.091 (0.339,

0.351)

0.609 0.050 (0.606,

0.612)

0.436 0.071 (0.431,

0.440)

0.348 0.058 (0.344,

0.351)

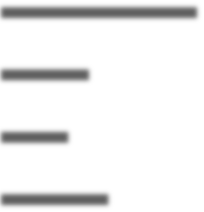
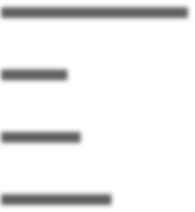
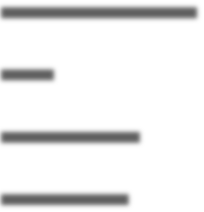
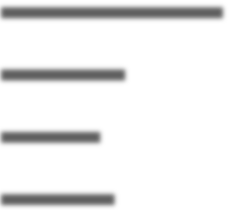
0.393 0.105 (0.386,

0.400)

0.631 0.052 (0.627,

0.634)

rest of the algorithms, for all metrics, is that of REINFORCE and appears fairly stable across the ten rounds. The error-bars are also pretty small, indicating lower uncertainty. We notice that the other three algorithms are all centered closely around, 0.4.



**Overall Performance**

REINFORCE

PPO

DQN

A2C

0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1

F beta (0.5) F1 Recall Precision

**Fig. 13** Overall model performance summary

Overall performance - F-beta 0.5



REINFORCE A2C

DQN PPO

1.0

0.8

0.6

F-beta 0.5

0.4

0.2

0.0

0 1 2 3 4 5 6 7 8 9

Training round

**Fig. 14** Overall performance comparison of the F*β*0.5 metric, over 10 rounds of training and testing.

We now dig into the detailed metrics Appendix [A](#_bookmark74) - Table [A2](#_bookmark75) to better understand the aggregated summary. The largest values are typeset in blue. And a quick visual glance shows the dominance of REINFORCE. The notable exceptions are the PHM C01 single-state and no-noise variant where A2C and DQN do better, followed by PHM C01 and PHM C06 univariate-state, low-noise variants where A2C performs better in every aspect and finally the PHM C04 and C06 multivariate variants where A2C again performs better on recall and which in turns drives the F1 score up. Barring these three complete cases and two cases where A2C recall was better, the REINFORCE performs best in ten variants and in two cases its precision, and therefore F*β*0.5, is highest. This corroborates the overall performance observed in Table [6](#_bookmark46). As seen in Table [A](#_bookmark75)2 there are significant variations in performance across the datasets - for *all* algorithms. One reason for this is by design – the sample datasets we selected had noticeable variation between them, as we saw in Fig.

1. Finally, Fig. [A1](#_bookmark77) shows that the stability across multiple rounds is evident for other metrics as well.

**Table 7** Simulated environments - model performance summary.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | Precision |  |  |  | Recall |  |  |  | F1-score |  |  |  | F*β*0.5 |  |
|  | Mean | SD | 95% CI |  | Mean | SD | 95% CI |  | Mean | SD | 95% CI |  | Mean | SD | 95% CI |

(0.408,

|  |  |  |
| --- | --- | --- |
| A2C | 0.416 | 0.120 |
| DQN | 0.432 | 0.184 |
| PPO | 0.500 | 0.178 |
| REINFORCE | 0.806 | 0.040 |

0.423)

(0.420,

0.443)

(0.488,

0.511)

(0.803,

0.808)

## Simulated environment

0.385 0.073 (0.380,

0.390)

0.510 0.031 (0.508,

0.512)

0.215 0.081 (0.209,

0.220)

0.915 0.038 (0.913,

0.918)

0.363 0.072 (0.358,

0.367)

0.374 0.034 (0.372,

0.376)

0.285 0.099 (0.278,

0.291)

0.841 0.035 (0.839,

0.844)

0.373 0.082 (0.368,

0.379)

0.351 0.056 (0.347,

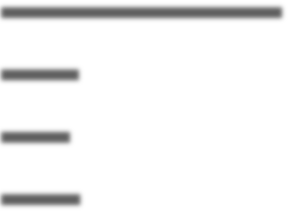
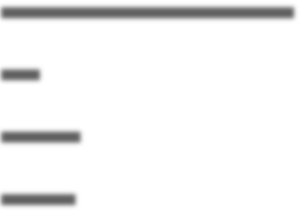
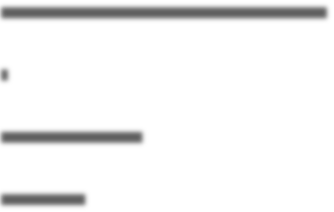
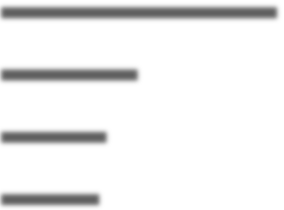
0.354)

0.370 0.122 (0.362,

0.378)

0.816 0.037 (0.813,

0.818)



**Simulated Environment**

REINFORCE

PPO

DQN

A2C

0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1

F beta (0.5) F1 Recall Precision

**Fig. 15** Simulated environments - model performance summary.

The simulated tool-wear environment is relatively the simplest for the agent to learn, even with added noise and chance of breakdown. Table [7](#_bookmark49) shows the averaged performance over the three variants. The REIN- FORCE performs the best and in absolute terms it is better than the next best advanced algorithm by very high margins: precision by 0.306, recall by 0.405, F1 by 0.468 and F*β*0.5 by 0.442, with standard deviation lower or marginally higher than others. F*β*0.5 plot Fig. [16](#_bookmark51) shows an exceptionally consistent performance, over all ten trained models, vis-a-vis the other algorithms. Plot Fig. [A2](#_bookmark78) shows DQN fluctuating heavily, occasionally showing recalls at the REINFORCE levels (rounds 2 and 8).

Simulated environment - F-beta 0.5



REINFORCE A2C

DQN PPO

1.0

0.8

0.6

F-beta 0.5

0.4

0.2

0.0

0 1 2 3 4 5 6 7 8 9

Training round

**Fig. 16** Simulated environments - F*β*0.5 metric, over 10 rounds of training and testing.

**Table 8** Model performance summary - averaged over PHM-2010 environments with simple single-variable environment.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | Precision |  |  |  | Recall |  |  |  | F1-score |  |  |  | F*β*0.5 |  |
|  | Mean | SD | 95% CI |  | Mean | SD | 95% CI |  | Mean | SD | 95% CI |  | Mean | SD | 95% CI |

(0.442,

|  |  |  |
| --- | --- | --- |
| A2C | 0.447 | 0.077 |
| DQN | 0.419 | 0.179 |
| PPO | 0.450 | 0.146 |
| REINFORCE | 0.605 | 0.046 |

0.452)

(0.408,

0.431)

(0.440,

0.459)

(0.602,

0.608)

0.477 0.091 (0.471,

0.483)

0.507 0.032 (0.505,

0.509)

0.314 0.082 (0.309,

0.319)

0.603 0.046 (0.600,

0.606)

0.452 0.072 (0.448,

0.457)

0.379 0.036 (0.376,

0.381)

0.333 0.087 (0.327,

0.339)

0.570 0.041 (0.567,

0.572)

0.446 0.070 (0.442,

0.451)

0.352 0.057 (0.348,

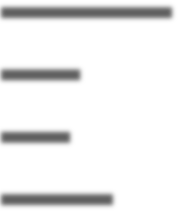
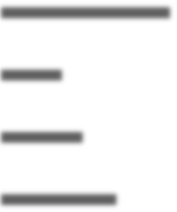
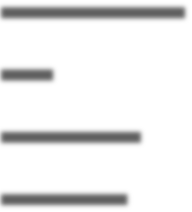
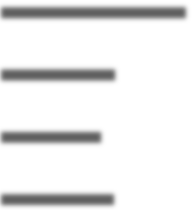
0.355)

0.374 0.102 (0.367,

0.381)

0.576 0.040 (0.574,

0.579)



**PHM data - Univariate state**

REINFORCE

PPO

DQN

A2C

0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1

F beta (0.5) F1 Recall Precision

**Fig. 17** PHM real data - Model performance summary for the simple single-variable environment.

## Real data - simple univariate environment

Real data offers a more challenging environment. Despite this, in Table [8](#_bookmark52), we notice that REINFORCE still performs better than the other algorithms. The margins are understandably lower: precision by 0.155, recall

**Table 9** Model performance summary - averaged over PHM-2010 environments with complex multivariate environment.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | Precision |  |  |  | Recall |  |  |  | F1-score |  |  |  | F*β*0.5 |  |
|  | Mean | SD | 95% CI |  | Mean | SD | 95% CI |  | Mean | SD | 95% CI |  | Mean | SD | 95% CI |

(0.481,

|  |  |  |
| --- | --- | --- |
| A2C | 0.487 | 0.086 |
| DQN | 0.399 | 0.204 |
| PPO | 0.512 | 0.107 |
| REINFORCE | 0.813 | 0.119 |

0.492)

(0.386,

0.412)

(0.505,

0.518)

(0.805,

0.821)

0.582 0.075 (0.577,

0.587)

0.491 0.032 (0.489,

0.493)

0.422 0.107 (0.414,

0.429)

0.421 0.079 (0.416,

0.426)

0.488 0.063 (0.484,

0.493)

0.361 0.035 (0.358,

0.363)

0.441 0.096 (0.435,

0.447)

0.495 0.090 (0.489,

0.501)

0.467 0.065 (0.463,

0.472)

0.332 0.060 (0.328,

0.336)

0.472 0.096 (0.465,

0.478)

0.609 0.101 (0.602,

0.615)

by 0.097, F1 by 0.117 and F*β*0.5 by 0.130. For these variants the F*β*0.5 plot Fig. [16](#_bookmark51) shows a considerably higher performance over 9 of the 10 training rounds. Behavior of the other metrics is seen in Fig. [A3](#_bookmark79). While the REINFORCE precision is higher for most rounds, the recall seems to be occasionally surpassed slightly by DQN’s (rounds 1, 8 and 9) and by a larger margin once (round 5).

PHM Real Data - Simple uni-variate state - F-beta 0.5



REINFORCE A2C

DQN PPO

1.0

0.8

0.6

F-beta 0.5

0.4

0.2

0.0

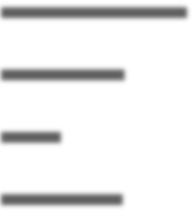
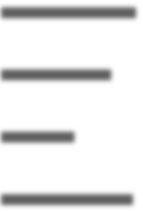
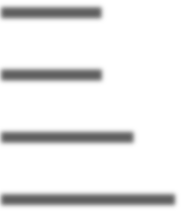
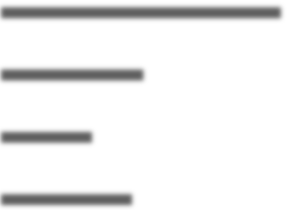
0 1 2 3 4 5 6 7 8 9

Training round

**Fig. 18** Univariate simple state environments - F*β*0.5 metric, over 10 rounds of training and testing.

## Real data - complex multivariate state

This environment offers the highest difficulty. We use real data, from *multiple* sensors. As with other scenarios, we used untuned, default settings of all algorithms. In Table [8](#_bookmark52), we see that the REINFORCE has the poorest recall at 0.421, however it demonstrates a surprisingly high tool replacement precision at 0.813, which drives the F1 and F*β*0.5 to the top of the table. F*β*0.5 performance is reflected in the Fig. [16](#_bookmark51); where it remains high for most rounds. The plot, in Appendix [A](#_bookmark74), Fig. [A4](#_bookmark80) shows REINFORCE’s high precision behavior. However, it is important to note that the error-bars are occasionally larger (rounds 0, 2, 4 and 6). The recall seemed higher at some points but lower half the times and this manifests in the F1 behavior.



**PHM data - Multivariate state**

REINFORCE

PPO

DQN

A2C

0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1

F beta (0.5) F1 Recall Precision

**Fig. 19** PHM real data - Model performance summary for the complex multivariate environment.

PHM Real Data - Multi-variate state - F-beta 0.5



REINFORCE A2C

DQN PPO

1.0

0.8

0.6

F-beta 0.5

0.4

0.2

0.0

0 1 2 3 4 5 6 7 8 9

Training round

**Fig. 20** Multivariate complex state environments - F*β*0.5 metric, over 10 rounds of training and testing.

## Super models

Finally, we look at performance of “select” models. Logically, one would choose the best model from a set of 10 trained models, just as an AutoML or AutoRL framework would. Selecting models based on the highest F1 with a *minimum* satisfying criteria for precision and recall allowed us to evaluate performance of “super- models” for each algorithm. At an overall averaged level Table [10](#_bookmark58) demonstrates the remarkable performance of the simple REINFORCE algorithm. It does show a lower recall when compared to the DQN, however it is a much more *balanced* model on an overall basis. Appendix [A](#_bookmark74) Table [A3](#_bookmark76) shows details where the REINFORCE performs better than the other three algorithms by a huge margin, for 14 of the 15 variants. For PHM C06, univariate environment, it was the DQN that performed best, with extremely high metrics throughout and an F1 of 0.969 to 0.831 of REINFORCE.

**Table 10** Super models: Best of 10 rounds; performance averaged over all 15 environments.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | Precision |  |  |  | Recall |  |  |  | F1-score |  |  |  | F*β*0.5 |  |
|  | Mean | SD | 95% CI |  | Mean | SD | 95% CI |  | Mean | SD | 95% CI |  | Mean | SD | 95% CI |

(0.518,

|  |  |  |
| --- | --- | --- |
| A2C | 0.520 | 0.031 |
| DQN | 0.651 | 0.022 |
| PPO | 0.558 | 0.076 |
| REINFORCE | 0.884 | 0.042 |

0.522)

(0.649,

0.652)

(0.553,

0.563)

(0.881,

0.887)

0.859 0.053 (0.855,

0.862)

0.937 0.031 (0.935,

0.939)

0.643 0.097 (0.636,

0.649)

0.884 0.042 (0.882,

0.887)

0.639 0.036 (0.636,

0.641)

0.740 0.022 (0.739,

0.742)

0.580 0.079 (0.575,

0.585)

0.873 0.034 (0.871,

0.875)

0.560 0.032 (0.558,

0.563)

0.678 0.021 (0.677,

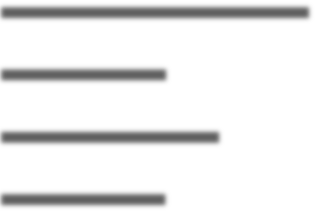
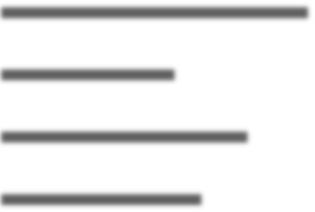
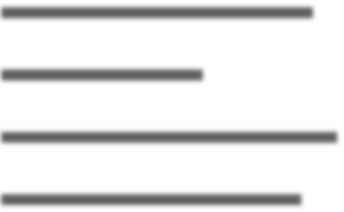
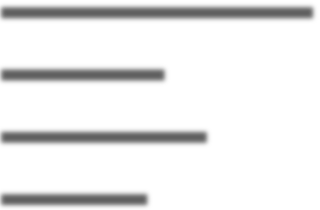
0.680)

0.562 0.075 (0.557,

0.567)

0.876 0.036 (0.873,

0.878)



**Super Models**

REINFORCE

PPO

DQN

A2C

0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1

F beta (0.5) F1 Recall Precision

**Fig. 21** Super models: Best of 10 rounds; performance averaged over all 15 environments.

## Hypothesis testing

We undertake a statistical analysis of the results, postulated by the hypothesis ([13](#_bookmark61)), where the subscripts *RF*

and *AA* stand for REINFORCE and “advanced algorithms” respectively. Table [11](#_bookmark63) shows the result of a one- sided, two-sample t-test, conducted for a significance level α = 0.05. Sample size of the test are mentioned against each category[6](#_bookmark62).

6What does a single data point indicate?: Consider the “Simulated” category – 10 training rounds × 10 test rounds × 3 noise settings

= 300 sample points. Note that a single test round is conducted with 40 randomly sampled wear datapoints and therefore 300 samples

= 300 ×40 i.e. 12,000 samples of wear data. Similarly, for the real-data univariate state: 3 datasets × 10 training rounds × 10 test rounds ×3 noise settings = 900 sample points.

**Table 11** Statistical test: One-sided two-sample t-tests. *H*0 : *µRF − µAA* = 0; *Ha* : *µRF − µAA >* 0, where *AA* is one of A2C, DQN or PPO.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | **p Value** |  |  |  | **t Statistic** |  |
| *Ha* | *Ha* | *Ha* |  | *Ha* | *Ha* | *Ha* |
| Metric | RF ≧ A2C | RF ≧ DQN | RF ≧ PPO |  | RF ≧ A2C | RF ≧ DQN | RF ≧ PPO |
|  | *H*0 | *H*0 | *H*0 |  | *H*0 | *H*0 | *H*0 |
| **Overall** (1500 samples) | | | | | | | |
| Precision | 4.31E-126 | 2.17E-109 | 2.81E-106 | 25.071 | | 23.170 | 22.804 |
| Recall | 4.20E-35 | 3.37E-16 | 4.36E-150 | 12.522 | | 8.206 | 27.650 |
| F1 score | 1.99E-64 | 1.46E-88 | 5.29E-155 | 17.364 | | 20.634 | 28.160 |
| **Simulated environment** (300 samples) | | | | | | | |
| Precision | 3.20E-98 | 1.69E-63 | 2.65E-81 | 25.611 | | 19.032 | 22.427 |
| Recall | 8.12E-104 | 2.56E-41 | 1.57E-264 | 26.665 | | 14.558 | 62.541 |
| F1 score | 9.60E-134 | 8.56E-99 | 2.96E-242 | 32.402 | | 25.719 | 56.575 |
| **PHM Real data - Simple univariate state** (900 samples) | | | | | | | |
| Precision | 2.27E-32 | 7.29E-31 | 9.95E-31 | 12.082 | | 11.770 | 11.742 |
| Recall | 1.27E-16 | 1.55E-06 | 8.19E-71 | 8.357 | | 4.821 | 18.607 |
| F1 score | 1.94E-19 | 4.67E-34 | 2.19E-67 | 9.121 | | 12.423 | 18.098 |
| **PHM Real data - Complex multivariate state** (300 samples) | | | | | | | |
| Precision | 1.64E-60 | 3.34E-54 | 7.88E-59 | 18.451 | | 17.207 | 18.122 |
| Recall | 2.69E-10 | 2.69E-02 | 9.68E-01 | -6.425 | | -2.219 | -0.041 |
| F1 score | 7.27E-01 | 1.44E-08 | 1.35E-03 | 0.349 | | 5.748 | 3.220 |

We see that the p-values are extremely low and test statistic positive, for all cases except for the *recall* in the PHM multivariate state case. We can thus reject H0.

### Training times and model byte size

Our final set of results are related to the training time required and size of models produced, for each algorithm. The naïve REINFORCE algorithm is extremely slow, with a very high variance in training time, as seen in Fig. [22](#_bookmark65).

The model sizes produced by all algorithms is extremely small: A2C 89.8 KB, DQN 56.4 KB, PPO 135.5

KB and REINFORCE 121.5 KB. We see that the DQN produced the smallest while the PPO produced the

largest.

Avg. training times across 10 training rounds (s)

200

Model training time (s)

139.42

150

100

1.32

50 31.55

30.23

REINFORCE A2C DQN PPO

**Fig. 22** Training time, averaged over 10 rounds and all 15 variants.

## Sensitivity analysis of hyperparameters on the REINFORCE algorithm

### Selection of hyperparameters for analysis

RL algorithms are known to be extremely sensitive to hyperparameter and network architecture settings, [[17](#_bookmark96), [43](#_bookmark121)]. We conduct a basic hyperparameter interaction and sensitivity analysis in this section. To select

the best hyperparameters to conduct the sensitivity analysis, we referred to recent research [[3](#_bookmark83)], [[19](#_bookmark98)], and [[44](#_bookmark122)]. [[3](#_bookmark83)] created the “AutoRL-Bench 1.0” for benchmarking AutoRL performance. We selected the *same*

common hyperparameters they studied across all RL algorithms they benchmark i.e. learning rate and the

discount factor γ. [[44](#_bookmark122)] studied empirically, the impact of exactly these two hyperparameters and their impact

on performance and instabilities of RL neural networks. Finally, [[19](#_bookmark98)] mention how some often overlooked

hyperparameters, such as the discount factor, significantly impacts the algorithm’s performance. In addi-

tion to these suggestions, we selected the network activation function for analysis, thus covering network

architecture as well.

##### Our hyperparameter space:

1. Learning rate: 1 × 10−4, 5 × 10−4, 1 × 10−3, 5 × 10−3, 1 × 10−2
2. Discount factor γ: 0.90, 0.95, 0.99
3. Network activation: Hyperbolic Tangent (Tanh) and Rectified Linear Unit (ReLU)

For our analysis we will train the model and compute the average episodic rewards, over 10 rounds of

training. Fig. [23](#_bookmark66) shows the impact of changing learning rates on the average episodic rewards, for a particular

γ value, with the network activation function set to ReLU. Uncertainty is indicated by the light blue region

bounded by 95% CI (confidence intervals). Similarly Fig. [2](#_bookmark67)4 shows the impact when the activation function is changed to Tanh. We observe that the optimal settings are the *higher* learning rates 5 × 10−3 − 1 × 10−2

and highest γ value 0.99, using the ReLU activation function. As a norm, *lower* learning rates should show

better performance and we will discuss this in Sec. [5](#_bookmark70).

activation:ReLU

= 0.99

= 0.95

activation:ReLU

0.5

2

1.5

Avg. Episodic Return

1.0

Avg. Episodic Return

1.5

1.0

1

Avg. Episodic Return

0.5

0.0 0

2.0

2.5

1e-4

activation:ReLU

= 0.90

5e-4

1e-3

Learning Rate

5e-3

1e-2

0.5

1.0

1e-4

5e-4

1e-3

Learning Rate

5e-3

1e-2

1

1e-4

5e-4

1e-3

Learning Rate

5e-3

1e-2

* 1. Discount factor *γ*=0.90
  2. Discount factor *γ*=0.95
  3. Discount factor *γ*=0.99

**Fig. 23** Impact of changing learning rates and discount factor *γ* for a REINFORCE policy network with ReLU activation.

2

1

0

1

2

activation:Tanh

= 0.95

activation:Tanh

= 0.99

0.5

0.0

0.5

1.0

1.5

2.0

2.5

= 0.90

activation: Tanh

1

Avg. Episodic Return

Avg. Episodic Return

Avg. Episodic Return

0

1

1e-4

5e-4

1e-3

Learning Rate

5e-3

1e-2

1e-4

5e-4

1e-3

Learning Rate

5e-3

1e-2

1e-4

5e-4

1e-3

Learning Rate

5e-3

1e-2

1. Discount factor *γ*=0.90
2. Discount factor *γ*=0.95
3. Discount factor *γ*=0.99

**Fig. 24** Impact of changing learning rates and discount factor *γ* for a REINFORCE policy network with Tanh activation.

### Impact of hyperparameter setting on training time

While REINFORCE performs reasonably well, it needs substantial time to train. To understand this better,

we analyze it with respect to the same hyperparameter settings. In Fig.[25(a)](#_bookmark68), we plot the time to train in

panels of increasing learning rate, and in Fig.[25(b)](#_bookmark69), we plot against panels of increasing γ. It is interesting

to see that for Tanh based activation the training time increases for increasing learning rate and γ, whereas

for ReLU it is relatively steady.

### Compute cost of performance improvement

Increased training time generally translates to more intense use of compute resources and therefore increased

cost. With Fig. [2](#_bookmark71)5 it is evident that the cost of compute is impacted primarily by the choice of activation

function. One final question we seek to answer is; does this increase show any correlation with the model

performance? Fig. [2](#_bookmark72)6 shows that, for both the activation functions, generally increased time translates to

increased performance; however due to REINFORCE’s high variability, this is not a sharply defined pattern.

Interestingly the ReLU, with significantly lower training cost compared to Tanh function, gives the best

performing model (and as expected for higher training time *within* the ReLU experiments).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **1e-4** | **5e-4** | **1e-3** | **5e-3** | **1e-2** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

(a) Impact of learning rate (b) Impact of discount factor *γ*

Training time (s) - Effect of Learning Rate

164

162

160

158

156

154

152

150

148

146

144

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

Experiment

ReLU Tanh

Training time (s) - Effect of Gamma (Discount rate)

164

162

160

158

156

154

152

150

148

146

144

**0.90**

**0.99**

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

Experiment

ReLU Tanh

**0.95**

Training Time (s)

Training Time (s)

**Fig. 25** Impact of learning rate and discount factor *γ* on the time to train a REINFORCE model.

**Fig. 26** The cost of performance improvement.



F1-score - Compute time (s)

1.00

0.90

0.80

0.70

Precision: **0.964**

Recall: **0.900**

F-1: **0.929**

Learning Rate: **0.01**

Gamma: **0.99**

Activation: **ReLU**

0.60

0.50

0.40

0.30

0.20

0.10

0.00

150.00 152.00 154.00

156.00

Training time (s)

158.00

160.00

162.00

F1-score

# Discussion

In Section [4](#_bookmark44) we conducted a series of extensive experiments comparing REINFORCE with A2C, DQN, and

PPO, with REINFORCE demonstrating significantly better performance than the other algorithms. As a

closed-loop system, the performance of REINFORCE, like any RL system, will experience the effects of inter-

action of hyperparameters [[5](#_bookmark85)], therefore to better understand REINFORCE’s performance we conducted

a basic interaction and sensitivity analysis of hyperparameters, in Sec. [4.7](#_bookmark64). In this Discussion section, we

first lay down views of researchers on the REINFORCE algorithm. We follow that by observations from our

research, covering properties of this algorithm (both positive and limiting), aspects related to its architec-

ture, training time, model size and cost effectiveness. We end the section with limitations of our research

and potential areas for future research.

As a simple model-free RL algorithm, REINFORCE offers a derivative-free optimization of unconstrained

problems.[[45](#_bookmark123)] explains that, provided one can sample efficiently, REINFORCE can be used to solve essentially

any problem! The inventor of the REINFORCE algorithm, in his original paper [[6](#_bookmark86)], mentions the inability to predict the convergent and asymptotic properties of this algorithm using analytical means. The author mentions that *simulation studies* are the primary source of understanding its behavior. 27 years later, [[46](#_bookmark124)] notes that rigorous convergence proofs are as of yet *not* available for many algorithms based on REINFORCE and in their paper provide an intuition for convergence properties. This supports the extensive experimental studies on REINFORCE performed by [[46](#_bookmark124)–[49](#_bookmark127)] as well our empirical research.

Both [[48](#_bookmark126)] and [[49](#_bookmark127)] studied the behavior using multiple OpenAI Gym environments, while [[46](#_bookmark124)] provide empirical results using MuJuCo. [[49](#_bookmark127)] observe that, despite its simplicity it is an effective algorithm in opti- mizing policies based on deep neural networks. On the other hand, their empirical study concluded that REINFORCE is occasionally trapped in premature convergence to local optima. REINFORCE suffers from high variance [[46](#_bookmark124)] and converges very slowly when it does. High variance implies that a large sample of environment interactions are needed to obtain a reasonable approximation of the actual gradient.

[[47](#_bookmark125)] provides the first known global convergence and sample efficiency results for the REINFORCE algorithm. Their study involved controlling the number of “bad” episodes as well as applying a “doubling” trick for studying the global convergence. Interestingly our training rounds resulted in multiple situations of “zero” performance (“bad” episodes) for REINFORCE (18 instances) as also for Stable-Baseline’s A2C implementation (12 instances); DQN and PPO did not show any occurrences.

Since RL algorithm implementations are incredibly sensitive to hyperparameters and the neural network architectures they employ, the working implementations need several decisions to be made during high-, as well as low- level design; these strongly impact the performance, [[17](#_bookmark96), [43](#_bookmark121)]. In fact the same algorithm could produce very different results depending on the code base one uses, [[17](#_bookmark96), [40](#_bookmark118)]. Our analysis in Sec. [4.](#_bookmark64)7 Fig.

[2](#_bookmark66)3 indicates that, in general, higher model performance (average episodic rewards) is attained with higher discount factors γ > 0.95, learning rates in the range 5 × 10−3 and 1 × 10−2, and when the policy network

is enabled by the ReLU activation function.

There could be three possible reasons for the positive performance demonstrated by REINFORCE. We used ReLU (rectified linear unit) as the activation function, while the other three algorithms used hyperbolic tangent (Tanh). [[17](#_bookmark96)] studied four policy-gradient methods, including the PPO and TRPO, and observed that ReLU activation performs best. Our study, Sec. [4.](#_bookmark64)7 Fig. [2](#_bookmark66)3 show higher average episodic rewards for

ReLU based network and corroborates with the research findings of [[17](#_bookmark96)]. The second reason is based on the

observations by [[49](#_bookmark127)]. They observed that even with a small learning rate REINFORCE sometimes resulted

in large changes to policy distribution, which possibly explains the fast convergence to local optima. On the other hand we used a larger learning rate (almost 50:1) and this could have assisted in reaching global optimas when it did. In their survey of Auto RL, [[5](#_bookmark85)], states that the design of neural network architec-

tures receives very little attention. Their survey concludes that there is a lack of conceptual understanding

regarding architectural design choices and the benefits they offer. They state that most implementations use

two or three hidden layers, similar to the algorithm implementations we studied in (Sec. [3.1](#_bookmark32)). Our REIN-

FORCE implementation was even simpler with just a single hidden layer and could be the third reason for

the superior performance on simple environments.

Training a REINFORCE algorithm requires more time. What strategies might help reducing this? [[44](#_bookmark122)]

has conducted an extensive analysis on the interaction of γ and the learning-rate. Lower values of γ lead

to faster convergence but by the very definition of γ, often result in myopic policies. Our study agrees with

this, as seen in Fig. [25(b)](#_bookmark69), for both the activation functions. According to [[44](#_bookmark122)] one strategy for speeding up

the learning process is to increase the discounting rate over time, rather than keeping it static. They further

show that simultaneously decreasing the learning rate while increasing γ further reduces training time and

thus improves learning speed. Our study points to one simple yet effective strategy of using ReLU activation

layer for reducing training time, without sacrificing performance, Fig. [26](#_bookmark72).

We saw that the model sizes for all algorithms are under 140 KB. This implies that the runtime memory

requirements are extremely small thus offering fast inferences. This suggests that such models can be used

in embedded form on IoT devices and housed near the machines. This is a significant benefit for creating

Industry 4.0 predictive maintenance solutions.

As a final point, we compare RL to the simpler ML methods, listed in Table [1](#_bookmark6), used for a similar PdM

problem. Supervised machine learning requires labeled data, RL on the other hand determines the right PdM

action via trial and error. The manufacturing industry often lacks labeled fault data as reported by extremely

recent surveys, [[50](#_bookmark128), [51](#_bookmark129)], and this is one good reason for promoting RL. As can be seen in the existing liter-

ature, supervised ML, when labeled fault data is available, will outperform RL. Unlike the closed-loop RL,

supervised ML is open-looped and the impact of hyperparameters is ascertainable and predictable and there-

fore are easier to tune, [[5](#_bookmark85)]. In addition to being stable, they require far lower compute resources to train. It

is therefore advisable to use ML in situations where good quality data is available.

## Limitations and future scope for research

Our research carries limitations – some due to the very design of our objective and some due to unavail-

ability of more resources and time. These however offer opportunities for further research and increase the

depth of this topic and provide more knowledge in the domain of AutoRL for predictive maintenance.

1. We used raw features, with no feature engineering. This is in tune with the “untuned” algorithms concept.

However, this creates a *possible* gap, where the more advanced models may demonstrate better perfor-

mance. With AutoRL one could automate such feature engineering techniques, followed by automated selection of significant features.

1. Advanced algorithms provide a richer set of hyperparameters and it is fair to assume that tuned versions

of these algorithms will perform better than the naïve REINFORCE.

1. We assume that the human preventive maintenance policy, based on experience, is ideal. In practice this might not always be the case. We had to use expert guidelines to decide the tool replacement wear

thresholds as these were not available for the PHM dataset.

**Avenues for future research**: Our research covers single-agent RL. Multi-agent and multi-objective RL

can solve more complex problems and are areas we plan to research in the near future. Offline RL (also called

Batch RL) is a paradigm where offline data in the form of previously collected data such as expert demon-

strations and experiments, offer a path to generalize agents to a real world environment. No environment

interaction is required in offline RL. Offline RL is suitable for the PdM domain and as such recommended

for future research.

# Conclusion

The REINFORCE is a simple algorithm and our naïve implementation can be improved drastically, for example with the implementation of a baseline to reduce variance. We evaluated the REINFORCE along with industry grade implementations of more advanced algorithms, DQN, A2C and PPO, over 15 environment variants (simulated and real data). Despite its simplicity, known variance and convergence issues, it performed surprisingly well, as observed through numerical results, plots and statistical tests. The hyperparameter

interaction study provided insights into the impact on performance, of the two standard hyperparameters

studied in the RL domain, i.e., the learning rate and the discount factor γ. Additionally, we studied the

impact of the two most common activation functions used for implementing RL algorithms i.e. ReLU and

Tanh. All this has surely created a valuable knowledge resource for the simple, often neglected, REINFORCE

algorithm. Future REINFORCE and HPO research is suggested in areas covering advanced RL paradigms

such as multi-agent, multi-objective and offline RL. If related breakthroughs can be achieved, AutoRL in

the hands of the practioner could truly solve problems of real industrial significance.

Implementing robust RL algorithms is complex and [[40](#_bookmark118)] puts this beautifully, mentioning how small implementation details can significantly impact performance *“that is often greater than the difference between the algorithms themselves”*.

For the AutoRL domain this research is significant. The results of *auto-selecting* the best untuned models

(Table [10](#_bookmark58)), along with the REINFORCE hyperparameter studies, encourages us to take a renewed look at

AutoRL methods for finding the optimum algorithm-hyperparameter combination and give fair consideration

to simpler algorithms such as the REINFORCE.

We hope our research contributes to empirical evidence related to the application of REINFORCE to predictive maintenance. This contribution aims to play a vital role in advancing the prospects of automated reinforcement learning i.e. AutoRL to Industry 4.0 revolution.

**Acknowledgements.** We would like to sincerely thank the anonymous reviewers of this research article.

They highlighted some deeper analysis that we performed, adding significantly to the empirical richness of

this research.

# Declarations

## Funding

None.

## Conflict of interest/Competing interests

The authors report there are no competing interests to declare.

## Ethics approval and consent to participate

Not applicable.

## Consent for publication

All listed authors grant permission for publication.

## Data availability

The data used for model training and validation is from [[39](#_bookmark117)] and publicly available at [IEEE*DataPort*](https://doi.org/10.21227/jdxd-yy51) .

## Materials availability Code availability

Will be available on request, via a GitHub link.

**Author contribution**

# Appendix A Detailed results

The Appendix contains detailed tabulated results as well as validation plots.

**Table A1** Reference table for results.

Item Reference

1. Complete detailed results – All 15 environments, averaged over 10 rounds Table [A2](#_bookmark75)
2. Super Models – Best model, *details* of all 15 environments Table [A3](#_bookmark76)
3. Stability behavior plots over 10 rounds, Precision, Recall and F1

Overall (all variants) Fig. [A1](#_bookmark77)

Simulated state Fig. [A2](#_bookmark78)

Simple univariate state Fig. [A3](#_bookmark79)

Complex multivariate state Fig. [A4](#_bookmark80)

37

**Table A2** Model performance comparison all variants of the environments, over 10 rounds of training. Maximum values indicated in blue.

**REINFORCE** A2C DQN PPO

Environment Prec. Recall F1 F*β*0.5 Prec. Recall F1 F*β*0.5 Prec. Recall F1 F*β*0.5 Prec. Recall F1 F*β*0.5

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Simulated | - No noise | 0.842 | 0.878 | 0.838 | 0.834 | 0.424 | 0.451 | 0.423 | 0.421 | 0.426 | 0.674 | 0.471 | 0.410 | 0.504 | 0.200 | 0.271 | 0.360 |
| Simulated | - Low noise | 0.777 | 0.929 | 0.834 | 0.796 | 0.465 | 0.423 | 0.409 | 0.427 | 0.421 | 0.338 | 0.270 | 0.283 | 0.482 | 0.236 | 0.296 | 0.369 |
| Simulated | - High noise | 0.798 | 0.940 | 0.851 | 0.816 | 0.358 | 0.281 | 0.256 | 0.272 | 0.447 | 0.519 | 0.380 | 0.360 | 0.514 | 0.207 | 0.286 | 0.382 |
| PHM C01 | SS - No noise | 0.478 | 0.363 | 0.400 | 0.439 | 0.501 | 0.500 | 0.493 | 0.496 | 0.472 | 0.807 | 0.568 | 0.490 | 0.440 | 0.417 | 0.387 | 0.395 |
| PHM C01 | SS - Low noise | 0.507 | 0.311 | 0.332 | 0.383 | 0.503 | 0.598 | 0.535 | 0.513 | 0.393 | 0.502 | 0.351 | 0.317 | 0.522 | 0.338 | 0.388 | 0.448 |
| PHM C01 | SS - High noise | 0.693 | 0.562 | 0.579 | 0.623 | 0.266 | 0.282 | 0.267 | 0.262 | 0.458 | 0.525 | 0.400 | 0.384 | 0.456 | 0.369 | 0.372 | 0.400 |
| PHM C04 | SS - No noise | 0.751 | 0.878 | 0.784 | 0.757 | 0.487 | 0.442 | 0.449 | 0.463 | 0.439 | 0.684 | 0.472 | 0.411 | 0.500 | 0.510 | 0.469 | 0.473 |
| PHM C04 | SS - Low noise | 0.662 | 0.756 | 0.672 | 0.657 | 0.409 | 0.455 | 0.428 | 0.416 | 0.411 | 0.500 | 0.370 | 0.341 | 0.488 | 0.280 | 0.324 | 0.386 |
| PHM C04 | SS - High noise | 0.611 | 0.713 | 0.620 | 0.598 | 0.518 | 0.607 | 0.552 | 0.530 | 0.358 | 0.451 | 0.325 | 0.294 | 0.428 | 0.262 | 0.286 | 0.333 |
| PHM C06 | SS - No noise | 0.830 | 0.726 | 0.754 | 0.792 | 0.517 | 0.509 | 0.507 | 0.511 | 0.360 | 0.309 | 0.256 | 0.258 | 0.409 | 0.248 | 0.275 | 0.321 |
| PHM C06 | SS - Low noise | 0.205 | 0.279 | 0.228 | 0.212 | 0.510 | 0.577 | 0.530 | 0.516 | 0.434 | 0.266 | 0.266 | 0.296 | 0.417 | 0.181 | 0.232 | 0.294 |
| PHM C06 | SS - High noise | 0.709 | 0.843 | 0.759 | 0.726 | 0.316 | 0.324 | 0.311 | 0.308 | 0.449 | 0.518 | 0.400 | 0.375 | 0.388 | 0.222 | 0.265 | 0.317 |
| PHM C01 | MS - No noise | 0.835 | 0.652 | 0.656 | 0.716 | 0.461 | 0.444 | 0.397 | 0.404 | 0.384 | 0.558 | 0.393 | 0.348 | 0.513 | 0.383 | 0.416 | 0.460 |
| PHM C04 | MS - No noise | 0.739 | 0.255 | 0.359 | 0.494 | 0.498 | 0.589 | 0.490 | 0.470 | 0.323 | 0.209 | 0.160 | 0.168 | 0.499 | 0.393 | 0.421 | 0.457 |
| PHM C06 | MS - No noise | 0.864 | 0.356 | 0.469 | 0.616 | 0.501 | 0.713 | 0.578 | 0.527 | 0.489 | 0.705 | 0.529 | 0.479 | 0.523 | 0.488 | 0.485 | 0.498 |

38

**Table A3** Super Models: Best models selected over 10 rounds of training. Maximum performance values indicated in blue.

**REINFORCE** A2C DQN PPO

Environment Prec. Recall F1 F*β*0.5 Prec. Recall F1 F*β*0.5 Prec. Recall F1 F*β*0.5 Prec. Recall F1 F*β*0.5

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Simulated | - No noise | 0.897 | 0.960 | 0.926 | 0.908 | 0.500 | 1.000 | 0.667 | 0.556 | 0.505 | 0.980 | 0.667 | 0.560 | 0.669 | 0.430 | 0.518 | 0.597 |
| Simulated | - Low noise | 0.960 | 0.945 | 0.952 | 0.957 | 0.516 | 1.000 | 0.680 | 0.571 | 0.500 | 0.980 | 0.662 | 0.554 | 0.633 | 0.460 | 0.530 | 0.586 |
| Simulated | - High noise | 0.922 | 0.990 | 0.955 | 0.935 | 0.503 | 1.000 | 0.669 | 0.558 | 0.504 | 0.990 | 0.668 | 0.559 | 0.569 | 0.355 | 0.434 | 0.505 |
| PHM C01 | SS - No noise | 0.889 | 0.995 | 0.939 | 0.908 | 0.586 | 0.625 | 0.603 | 0.592 | 0.647 | 0.970 | 0.776 | 0.693 | 0.543 | 1.000 | 0.703 | 0.597 |
| PHM C01 | SS - Low noise | 0.988 | 0.765 | 0.861 | 0.932 | 0.499 | 0.995 | 0.664 | 0.554 | 0.504 | 0.990 | 0.668 | 0.559 | 0.623 | 0.740 | 0.675 | 0.643 |
| PHM C01 | SS - High noise | 0.850 | 0.970 | 0.905 | 0.871 | 0.521 | 0.680 | 0.588 | 0.546 | 0.505 | 0.985 | 0.668 | 0.560 | 0.520 | 0.725 | 0.604 | 0.551 |
| PHM C04 | SS - No noise | 0.811 | 1.000 | 0.895 | 0.842 | 0.536 | 0.645 | 0.583 | 0.554 | 0.501 | 0.965 | 0.660 | 0.554 | 0.579 | 0.895 | 0.702 | 0.622 |
| PHM C04 | SS - Low noise | 0.798 | 0.980 | 0.879 | 0.829 | 0.556 | 0.665 | 0.603 | 0.573 | 0.734 | 0.990 | 0.843 | 0.774 | 0.546 | 0.660 | 0.596 | 0.565 |
| PHM C04 | SS - High noise | 0.708 | 0.840 | 0.767 | 0.730 | 0.521 | 0.835 | 0.641 | 0.563 | 0.511 | 0.985 | 0.672 | 0.565 | 0.517 | 0.820 | 0.633 | 0.558 |
| PHM C06 | SS - No noise | 1.000 | 0.895 | 0.944 | 0.977 | 0.520 | 0.680 | 0.587 | 0.545 | 0.935 | 0.975 | 0.954 | 0.942 | 0.587 | 0.650 | 0.615 | 0.597 |
| PHM C06 | SS - Low noise | 0.943 | 0.795 | 0.861 | 0.908 | 0.501 | 1.000 | 0.668 | 0.557 | 0.961 | 0.725 | 0.826 | 0.901 | 0.552 | 0.370 | 0.438 | 0.497 |
| PHM C06 | SS - High noise | 0.821 | 0.845 | 0.831 | 0.825 | 0.540 | 0.755 | 0.628 | 0.572 | 0.980 | 0.960 | 0.969 | 0.976 | 0.521 | 0.615 | 0.564 | 0.537 |
| PHM C01 | MS - No noise | 0.827 | 0.995 | 0.903 | 0.856 | 0.500 | 1.000 | 0.667 | 0.556 | 0.505 | 0.985 | 0.668 | 0.560 | 0.512 | 0.595 | 0.549 | 0.526 |
| PHM C04 | MS - No noise | 0.910 | 0.425 | 0.577 | 0.738 | 0.500 | 1.000 | 0.667 | 0.556 | 0.501 | 0.975 | 0.662 | 0.555 | 0.501 | 0.635 | 0.558 | 0.522 |
| PHM C06 | MS - No noise | 0.934 | 0.865 | 0.896 | 0.918 | 0.500 | 1.000 | 0.667 | 0.556 | 0.969 | 0.600 | 0.741 | 0.863 | 0.497 | 0.690 | 0.577 | 0.526 |

#### Overall performance - Precision



REINFORCE A2C

DQN PPO

1.0

0.8

0.6

Precision

0.4

0.2

0.0

0 1 2 3 4 5 6 7 8 9

Training round

* 1. Precision

#### Overall performance - Recall



REINFORCE A2C

DQN PPO

1.0

0.8

0.6

Recall

0.4

0.2

0.0

0 1 2 3 4 5 6 7 8 9

Training round

* 1. Recall

#### Overall performance - F-1 score



REINFORCE A2C

DQN PPO

1.0

0.8

0.6

F-1 score

0.4

0.2

0.0

0 1 2 3 4 5 6 7 8 9

Training round

* 1. F1-score

**Fig. A1** Overall performance – Average over 10 modelsOverall performance – Average over 10 models

#### Simulated environment - Precision



REINFORCE A2C

DQN PPO

1.0

0.8

0.6

Precision

0.4

0.2

0.0

0 1 2 3 4 5 6 7 8 9

Training round

1. Precision

#### Simulated environment - Recall



REINFORCE A2C

DQN PPO

1.0

0.8

0.6

Recall

0.4

0.2

0.0

0 1 2 3 4 5 6 7 8 9

Training round

1. Recall

#### Simulated environment - F-1 score



REINFORCE A2C

DQN PPO

1.0

0.8

0.6

F-1 score

0.4

0.2

0.0

0 1 2 3 4 5 6 7 8 9

Training round

1. F1-score

**Fig. A2** Simulated environment – Average over 10 models

#### PHM Real Data - Simple uni-variate state - Precision



REINFORCE A2C

DQN PPO

1.0

0.8

0.6

Precision

0.4

0.2

0.0

0 1 2 3 4 5 6 7 8 9

Training round

1. Precision

#### PHM Real Data - Simple uni-variate state - Recall



REINFORCE A2C

DQN PPO

1.0

0.8

0.6

Recall

0.4

0.2

0.0

0 1 2 3 4 5 6 7 8 9

Training round

1. Recall

#### PHM Real Data - Simple uni-variate state - F-1 score



REINFORCE A2C

DQN PPO

1.0

0.8

0.6

F-1 score

0.4

0.2

0.0

0 1 2 3 4 5 6 7 8 9

Training round

1. F1-score

**Fig. A3** Univariate environment – Average over 10 models

#### PHM Real Data - Multi-variate state - Precision



REINFORCE A2C

DQN PPO

1.0

0.8

0.6

Precision

0.4

0.2

0.0

0 1 2 3 4 5 6 7 8 9

Training round

1. Precision

#### PHM Real Data - Multi-variate state - Recall



REINFORCE A2C

DQN PPO

1.0

0.8

0.6

Recall

0.4

0.2

0.0

0 1 2 3 4 5 6 7 8 9

Training round

1. Recall

#### PHM Real Data - Multi-variate state - F-1 score



REINFORCE A2C

DQN PPO

1.0

0.8

0.6

F-1 score

0.4

0.2

0.0

0 1 2 3 4 5 6 7 8 9

Training round

1. F1-score

**Fig. A4** Multivariate environment – Average over 10 models

# References

1. Future Market Insights: Milling Machine Market Outlook (2023 to 2033). Future Market Insights, Inc.

Accessed: 2023-06-23 (2023). <https://www.futuremarketinsights.com/reports/milling-machine-market>

1. Sutton, R., Barto, A.: Reinforcement Learning: An Introduction, 2nd. edition edn. The MIT Press, Cambridge, England (2018)
2. Shala, G., Arango, S.P., Biedenkapp, A., Hutter, F., Grabocka, J.: AutoRL-bench 1.0. In: Sixth Workshop on Meta-Learning at the Conference on Neural Information Processing Systems (2022). <https://openreview.net/forum?id=RyAl60VhTcG>
3. Afshar, R.R., Zhang, Y., Vanschoren, J., Kaymak, U.: Automated reinforcement learning: An overview. arXiv preprint (2022)
4. Parker-Holder, J., Rajan, R., Song, X., Biedenkapp, A., Miao, Y., Eimer, T., Zhang, B., Nguyen, V., Calandra, R., Faust, A., *et al.*: Automated reinforcement learning (autorl): A survey and open problems. Journal of Artificial Intelligence Research **74**, 517–568 (2022)
5. Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. Reinforcement learning, 5–32 (1992)
6. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.: Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602 (2013)
7. Mnih, V., Badia, A.P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., Kavukcuoglu, K.: Asyn- chronous methods for deep reinforcement learning. In: International Conference on Machine Learning,

pp. 1928–1937 (2016). PMLR

1. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)
2. Panzer, M., Bender, B.: Deep reinforcement learning in production systems: a systematic literature review. International Journal of Production Research (2021)
3. Erhan, L., Ndubuaku, M., Di Mauro, M., Song, W., Chen, M., Fortino, G., Bagdasar, O., Liotta, A.: Smart anomaly detection in sensor systems: A multi-perspective review. Information Fusion **67**, 64–79 (2021) <https://doi.org/10.1016/j.inffus.2020.10.001>
4. Siraskar, R., Kumar, S., Patil, S., Bongale, A., Kotecha, K.: Reinforcement learning for predictive maintenance: a systematic technical review. Artificial Intelligence Review, 1–63 (2023)
5. Sandeep Varma, N., Sinha, V., Pradyumna Rahul, K.: Experimental evaluation of reinforcement learning algorithms. In: International Conference on Computational Intelligence and Data Engineering, pp. 469– 484 (2022). Springer
6. Velivela, K., Yarram, S.: Comparison of Reinforcement Learning Algorithms. Department of Computer Science and Engineering, University at Buffalo (2020)
7. Dulac-Arnold, G., Levine, N., Mankowitz, D.J., Li, J., Paduraru, C., Gowal, S., Hester, T.: Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. Machine Learning **110**(9), 2419–2468 (2021)
8. Dulac-Arnold, G., Levine, N., Mankowitz, D.J., Li, J., Paduraru, C., Gowal, S., Hester, T.: An empirical investigation of the challenges of real-world reinforcement learning. arXiv preprint arXiv:2003.11881 (2020)
9. Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., Meger, D.: Deep reinforcement learning that matters. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32 (2018)
10. Ford, S., Ritchie, M.: Cognitive radar mode control: a comparison of different reinforcement learning algorithms. In: International Conference on Radar Systems (RADAR 2022), vol. 2022, pp. 107–112 (2022). IET
11. Eimer, T., Lindauer, M., Raileanu, R.: Hyperparameters in reinforcement learning and how to tune them. In: International Conference on Machine Learning, pp. 9104–9149 (2023). PMLR
12. Fu, L., Yan, K., Zhang, Y., Chen, R., Ma, Z., Xu, F., Zhu, T.: Edgecog: a real-time bearing fault diagnosis system based on lightweight edge computing. IEEE Transactions on Instrumentation and Measurement (2023)
13. Pandey, R., Uziel, S., Hutschenreuther, T., Krug, S.: Towards deploying dnn models on edge for predictive maintenance applications. Electronics **12**(3), 639 (2023)
14. Siraskar, R., Kumar, S., Patil, S., Bongale, A., Kotecha, K.: Application of the Nadaraya-Watson esti- mator based attention mechanism to the field of predictive aintenance. MethodsX **12**, 102754 (2024).

[Online; accessed 2024-12-26]

1. Nadaraya, E.A.: On estimating regression. Theory of Probability & Its Applications **9**(1), 141–142 (1964) <https://doi.org/10.1137/1109020>
2. Niu, B., Sun, J., Yang, B.: Multisensory based tool wear monitoring for practical applications in milling of titanium alloy. Materials today: proceedings **22**, 1209–1217 (2020)
3. Yang, B., Guo, K., Liu, J., Sun, J., Song, G., Zhu, S., Sun, C., Jiang, Z., *et al.*: Vibration singularity analysis for milling tool condition monitoring. International Journal of Mechanical Sciences **166**, 105254 (2020)
4. Lei, Y., Yang, B., Jiang, X., Jia, F., Li, N., Nandi, A.K.: Applications of machine learning to machine fault diagnosis: A review and roadmap. Mechanical systems and signal processing **138**, 106587 (2020)
5. Zhou, C., Guo, K., Sun, J., Yang, B., Liu, J., Song, G., Sun, C., Jiang, Z.: Tool condition moni- toring in milling using a force singularity analysis approach. The International Journal of Advanced Manufacturing Technology **107**, 1785–1792 (2020)
6. Ou, J., Li, H., Huang, G., Zhou, Q.: A novel order analysis and stacked sparse auto-encoder feature learning method for milling tool wear condition monitoring. Sensors **20**(10), 2878 (2020)
7. Madhusudana, C., Kumar, H., Narendranath, S.: Condition monitoring of face milling tool using k- star algorithm and histogram features of vibration signal. Engineering science and technology, an international journal **19**(3), 1543–1551 (2016)
8. Patange, A., Jegadeeshwaran, R., Dhobale, N.: Milling cutter condition monitoring using machine learn- ing approach. In: IOP Conference Series: Materials Science and Engineering, vol. 624, p. 012030 (2019).

IOP Publishing

1. Liu, R., Kothuru, A., Zhang, S.: Calibration-based tool condition monitoring for repetitive machining operations. Journal of Manufacturing Systems **54**, 285–293 (2020)
2. Schulman, J., Levine, S., Abbeel, P., Jordan, M., Moritz, P.: Trust region policy optimization. In:

International Conference on Machine Learning, pp. 1889–1897 (2015). PMLR

1. Bouchrika, I.: What is empirical research? Definition, types and samples in 2024. https://research.com/research/what-is-empirical-research. [Online; accessed 2024-12-26] (2024)
2. Conduct empirical research. Emerald Publishing Limited. [Online; accessed 2024-12-26] (2021)
3. Gamper, J.: Research METHODS empirical/experimental CS research methods. Faculty of Engineering Free University of Bozen-Bolzano (2017). [Online; accessed 2024-12-29]
4. Groot, A.D.D., A., S.J.A.: The Empirical Cycle In Science, pp. 1–32. De Gruyter Mouton, Berlin, Boston (1969). <https://doi.org/10.1515/9783112313121-003> . <https://doi.org/10.1515/9783112313121-003>
5. Politzer-Ahles, S.: Exploratory vs. confirmatory research. h[ttps://www.polyu.edu.hk/cbs/sjpolit/classes/cbs6442/Types](http://www.polyu.edu.hk/cbs/sjpolit/classes/cbs6442/Types) vs-confirmatory.html. [Online; accessed 2024-12-28] (2021)
6. Daˇsi´c, P.: Analysis of wear cutting tools by complex power-exponential function for finishing turning of the hardened steel 20crmo5 by mixed ceramic tools. Fascicle VIII Tribology **12**, 54–60 (2006)
7. Li, X.: 2010 PHM Society Conference Data Challenge. IEEE Dataport (2021). [https://doi.org/10.21227/](https://doi.org/10.21227/jdxd-yy51) [jdxd-yy51](https://doi.org/10.21227/jdxd-yy51) . <https://dx.doi.org/10.21227/jdxd-yy51>
8. Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., Dormann, N.: Stable-baselines3: Reliable reinforcement learning implementations. J. Mach. Learn. Res. **22**(1) (2021)
9. SB3-Algorithms: Stable-Baselines3 - Master list of algorithms. Accessed: 2023-06-27 (2022). [https://](https://stable-baselines3.readthedocs.io/en/master/guide/algos.html) [stable-baselines3.readthedocs.io/en/master/guide/algos.html](https://stable-baselines3.readthedocs.io/en/master/guide/algos.html)
10. SB3-Default Network Architecture: Stable-Baselines3 - Default Network Architecture. Accessed: 2023-06-27 (2022). [https://stable-baselines3.readthedocs.io/en/master/guide/custom policy.html#](https://stable-baselines3.readthedocs.io/en/master/guide/custom_policy.html#default-network-architecture) [default-network-architecture](https://stable-baselines3.readthedocs.io/en/master/guide/custom_policy.html#default-network-architecture)
11. Andrychowicz, M., Raichuk, A., Stan´czyk, P., Orsini, M., Girgin, S., Marinier, R., Hussenot, L., Geist, M., Pietquin, O., Michalski, M., *et al.*: What matters for on-policy deep actor-critic methods? a large- scale study. In: International Conference on Learning Representations (2021)
12. Franc¸ois-Lavet, V., Fonteneau, R., Ernst, D.: How to discount deep reinforcement learning: Towards new dynamic strategies. CoRR **abs/1512.02011** (2016)
13. Matni, N.: ESE 680-004: Learning and Control - Lecture 20: Model Free Methods. University of Pennsylvania (2019)
14. Peck, R., Renaux, L.: A review of reinforce algorithms (2019)
15. Zhang, J., Kim, J., O’Donoghue, B., Boyd, S.: Sample efficient reinforcement learning with reinforce. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, pp. 10887–10895 (2021)
16. Zhang, J., Ni, C., Szepesvari, C., Wang, M., *et al.*: On the convergence and sample efficiency of variance- reduced policy gradient method. Advances in Neural Information Processing Systems **34**, 2228–2240 (2021)
17. Duan, Y., Chen, X., Houthooft, R., Schulman, J., Abbeel, P.: Benchmarking deep reinforcement learning for continuous control. In: International Conference on Machine Learning, pp. 1329–1338 (2016). PMLR
18. Leite, D., Andrade, E., Rativa, D., Maciel, A.M.A.: Fault detection and diagnosis in industry 4.0: A review on challenges and opportunities. Sensors **25**(1) (2025) <https://doi.org/10.3390/s25010060>
19. Hexagon: 98Hexagon’s report reveals. https://hexagon.com/company/newsroom/press- releases/2024/98-percent-manufacturers-face-data-woes-that-stifle-innovation-and-time-to-market. [Online; accessed 2025-01-03] (2024)