Industrial insights of practical deep reinforcement learning projects

Hugo Massonnat[1,*], **Antoine Plissonneau-Duquène**[1,*], **Elie Kadoche**[1], **Louis Verny**[1], **Florence Carton**[1]

{hugo.massonnat,antoine.plissonneau-duquene}@external.totalenergies.com,
{elie.kadoche,louis.verny,florence.carton}@totalenergies.com

[1]**TotalEnergies OneTech, 2 Place Jean Millier, 92400 Courbevoie, France**

[*] Equal contribution

Abstract

Reinforcement learning is a category of artificial intelligence algorithms that has seen significant growth in recent years. Despite impressive results in a wide range of domains such as robotics, game playing, and autonomous systems, its deployment in real-world systems remains very limited. In this paper, we present two case studies on the deployment of RL agents in real-world systems within an industrial context. The paper is structured chronologically around the life cycle of a project, in order to highlight the different stages, key considerations, and challenges involved in deploying an RL agent in a real-world system. This paper aims at providing insights into the following questions: What should be considered when selecting a reinforcement learning project? How can an RL project be successfully implemented in the industry? What challenges arise when deploying RL in production?

1 Introduction

Reinforcement learning (RL) is a branch of machine learning in which agents learn to make decisions by interacting with an environment and receiving feedback in the form of rewards or penalties. Unlike supervised learning, RL does not require labeled datasets; instead, it learns optimal behaviors through trial and error. This makes RL particularly appealing for complex, dynamic systems where explicit programming or large-scale data collection is impractical. Reinforcement Learning has demonstrated promising results in areas such as robotics, game playing, and resource management, offering the potential for autonomous systems that adapt and improve over time. However, these strengths come with significant challenges, especially when transitioning from simulation to real-world deployment (Dulac-Arnold et al., 2019; Henderson et al., 2018). In this paper, we focus on two projects in the field of renewable energy, either already deployed or currently in deployment: one involving an energy management system, and the other focused on optimizing the orientation of solar panels. Our discussion will cover deployment aspects and challenges throughout the lifetime of industrial projects, without delving into algorithmic or technical details. While such details are crucial to system performance, the industrial nature of these projects prevents us from disclosing further information about their internal workings.

2 The choice of relevant use cases

Generally speaking, Reinforcement Learning is used for sequential optimization, where actions have long-term consequences. Through its trial-and-error mechanism, RL learns by interacting with its

environment and does not require an explicit or differentiable optimization function. It provides particularly significant benefits when the problem falls into one or several of the following cases:

1. *Real time optimization:* when the system requires a real-time response, a trained policy can provide near-instantaneous solutions.
2. *Combinatorial explosion:* when facing a problem with a high number of possible combinations, RL is an interesting alternative to classical heuristics or meta heuristics methods.
3. *Uncertainties in the input:* when using forecasts (such as weather forecasts to predict PV yield) as part of the input, a RL agent is able to learn bias and potential errors in the forecast, and to react accordingly.

The type of use case is equally important, as certain application domains may not be suited to RL approaches for safety reasons. Deep learning is frequently used to scale RL systems for industrial applications. However, as it will be discussed, assessing the robustness of deep RL controllers poses significant challenges. This inherent difficulty in guaranteeing predictable and safe behavior makes it exceptionally challenging to gain business confidence for deploying such controllers in safety-critical systems. Ethical considerations, including bias, transparency, and accountability, are also crucial. Consequently, prioritizing non-critical systems for initial industrial deployments is advisable, enabling the accumulation of experience and building confidence in the technology.

In the remainder of this article, we will introduce two distinct projects that are particularly interesting for RL testing and deployment: an RL-based Energy Management System (EMS) and a controller for solar panels in agrivoltaic plants.

The first project explores how RL can improve energy management in a residential building, represented as a microgrid that includes the building itself, an EV station, solar panels, a cogenerator, and a battery. The goal is to compare the existing Model Predictive Control (MPC) based EMS with an RL-based controller. The RL model is trained using time-series data, including forecasts for solar production, energy consumption, and electricity prices and learns to optimize battery usage to reduce costs and/or carbon emissions. This RL agent is currently deployed in a residential building, where it autonomously controls a battery system to optimize energy usage.

The second project uses RL to optimize the operation of solar panels in an agrivoltaic setting, in which PV panels are installed above agricultural fields. The panels are mounted on trackers and their tilt angle can be adjusted to improve light collection for PV generation, or to reduce the shade which might impact crop growth (as in Alam & Butt (2024)). The key challenge lies in balancing two competing objectives across different time horizons: maximizing electricity generation in the short term, while preserving crop yield over the long term. The RL agent learns an optimal control policy of the panels, using various inputs such as weather forecasts or crop-related measurements. The inherent uncertainty in weather forecasts, coupled with the unpredictability of projected yield growth, poses significant challenges. Additionally, the vast number of possible decision-making strategies makes exhaustive exploration impractical, further highlighting the potential of RL as a powerful tool for navigating such complex, high-dimensional environments. This project is currently being tested on a private, company-owned pilot site.

3 The timeline of a PoC

Once the project has been identified and the relevance of using RL has been established, several hurdles must be overcome to ensure the project can be successfully completed. These challenges fall into three categories: convincing the business, validating the technical feasibility, and addressing the scientific challenges.

3.1 Convincing the business

The very first phase of a proof-of-concept involves thoroughly substantiating the project's value to key business stakeholders. It should always be aligned with the objectives of the company (financial or others, such as decarbonation in the field of renewable energy) and estimated with Key Performance Indicators (KPI). To effectively demonstrate this value, initial tests can be performed on simplified versions of the problem.

For the EMS project, we began with a simplified problem that does not capture the full complexity present in some microgrids, allowing us to quickly establish an initial benchmark of the added value RL can bring to this type of challenge. The first validation stage was carried out on a fairly simple simulator, which enabled us to compare the performance of the RL approach with that of the conventional method (MPC, OR-based algorithms, genetic algorithms, etc...). Fast iteration cycles are a decisive advantage: they keep business stakeholders engaged, allow for a steady stream of feedback, and sustain the momentum of the project. We also observed that involving them in the solution design process, such as discussing reward trade-offs in multi-objective scenarios,facilitates greater acceptance of the project.

Although immediate financial returns are key, a RL initiative also presents significant long-term strategic value. Even if a specific RL application does not deliver an immediate competitive advantage, its significance is likely to escalate in the coming years. Proactive engagement with emerging technologies allows a company to cultivate deep internal expertise, mitigating future risks and ensuring readiness to leverage the technology once it achieves the necessary maturity to drive substantial impact. This contrasts with the rapid publication cycle of the state-of-the-art academic research, as industry generally prefers a more gradual, incremental adoption curve.

3.2 Validating the technical feasibility

Even for non-critical use cases such as solar tracker control and microgrid management, a strategy purely generated by a RL algorithm may not be applicable as is on real production sites; there are often essential notions of reliability and safety to consider. When operating industrial systems, there may be limitations that the RL agent is unaware of during training (such as hardware constraints or limits set by the manufacturer to preserve the warranty of the system). This motivates a post-processing of the RL strategy to enforce these constraints.

The deployment on the agrivoltaics site is a good illustration of this process:

- The RL agent controls the photovoltaic (PV) panels during regular operation, but the tracker manufacturer manages a security layer that takes precedence if needed. During high-wind events (or any other environmental hazard), the trackers will default to a safety position no matter the command sent by the agent.
- Different PV sites have different values for the maximum tilt angle of PV modules. The output of the RL agent is therefore clipped to ensure that the trackers are not damaged.
- Tracker manufacturers will often revoke the warranty of their products if they are used in a way that might degrade them more quickly (e.g., by moving the panels often and with large rotations). Smoothing the output (by sending commands less frequently or applying a Gaussian filter) might help with these issues.

These safeguards are analogous to the post-posed shielding method outlined in Alshiekh et al. (2018). Adding safeguards is essential for production because it shifts safety assurance to external components that are easier to certify and audit. To complement the previous examples, it should be noted that when those constraints are imposed after training, the agent never gets to see them: it keeps trying to maximize the reward, only to have some of its actions rejected at inference time, which can cripple performance. Consider a warranty constraint that limits the yearly rotation budget of PV trackers: once the safeguard blocks further movement, the controller can no longer act, even though a more frugal policy would have preserved both performance and warranty. In addition to the

post-processing, future development efforts should try to embed such limits directly in the learning loop (e.g. using safe-RL techniques), so the agent learns to trade off reward and compliance from the outset.

The main take-away in terms of safety is that not all constraints related to the industrial nature of the application may be addressed in the development and the training of the RL algorithms. Besides, RL relies on neural networks and its output may have more variability than other approaches. Adding some post-processing after the inference helps mitigate some of these issues.

3.3 Addressing the scientific challenges

Finally, implementing a RL-based solution to a problem is possible only with the right framework. In order to learn strategies that answer the specific use case, the RL agent must receive rewards that are accurate and represent the problem to its full extent. This is typically done using a simulator that estimates the response of the system to the strategy of the agent (in our projects, PV and crop yields or battery state of charge). If there is a solution that is already developed internally or by the community, only minor development time may be needed to build a RL framework around it. When that is not the case, the challenge in building a simulator and adapting it to an RL framework should not be underestimated. Ideally, the simulator should be validated using real-world data (from open datasets or a small-scale test site with enough instrumentation). For exogenous variables (which are not affected by the actions of the agent), an alternative strategy is to rely on historical time-series data. This approach is especially relevant when the transition dynamics of the variables are difficult to model. It can help reduce the gap between simulation and reality by exposing the algorithm to more realistic operating conditions.

To reduce the friction that the development of a simulator might cause, a good strategy is to build it iteratively, from an early version with simplifying assumptions to a fully-fledged tool that takes all the phenomena specific to the use case into account.

Once the simulator is accurate and powerful enough to represent all the situations that might be encountered during learning, an expression of the reward should be formulated from its outputs. Depending on the application, this step can be more or less straightforward. At minimum, the reward (or the evaluation metric if it is different) should be a quantity that is (a) understandable by the business and the domain experts, and (b) closely related to a business objective. This is essential to keep the RL approach grounded in reality and to make it more explainable to people without a data science background. To add context and gain more insight on the performance of RL solutions, known baselines should be implemented and tested on the RL environment. This also makes it easier to compare the results with other state-of-the-art approaches that might mention the same baselines.

In all industrial applications, strong technical knowledge is required to guarantee that the simulation is adapted to a given problem, and to assess the relevance of the solutions learned by RL algorithms. Data scientists should interact frequently with domain experts and learn enough from them or the literature to avoid the pitfall of solving an abstract black-box problem. This is especially true with RL, which is known for its ability to find exotic solutions and hack the reward signal in unintended ways.

Indeed, a critical aspect of RL lies in the design of the reward function, which is inherently linked to the problems of artificial intelligence alignment and perverse incentives. A reward function, while seemingly intuitive from the perspective of a human expert, may prove suboptimal or even misaligned when interpreted through the lens of a RL agent's objective. A well-optimized RL algorithm can exploit a poorly designed reward function, yielding solutions that are optimal with respect to the defined objective but demonstrably suboptimal or undesirable in the real world. Consequently, meticulous attention to reward engineering is crucial for successful and safe RL deployment.

Finally, one of the main challenges in deploying a RL model in a real industrial setting is the discrepancy between the simulated environment used for training and the complexities of the real world -a phenomenon commonly referred to as the reality gap. This gap can significantly hinder the performance of the model. To address this issue before scaling up the deployment, we typically conduct

tests on smaller-scale sites: a smart building named Bachelor and an agrivoltaic pilot site called InnoAgri. Both are equipped with numerous sensors, allowing us to closely monitor and analyze the differences between simulated and real-world conditions.

4 Towards industrialization

Once a reinforcement learning (RL) solution demonstrates consistent and significant performance improvements over previously used approaches, the project may enter a new phase focused on upscaling for industrialization. It comes with new challenges in the form of generalizing the solution to more diverse application settings, while engaging continuous efforts for monitoring and performance improvements. As of today, while the projects mentioned in this article have already been deployed on pilot sites, none has completed this transformation to an industrial product.

4.1 When is generalization worth the effort?

Training a RL policy has three main cost drivers: data and simulation (high-fidelity simulators, historical data, forecasts), training compute and engineering, and validation. For a single high-value asset (e.g., a 1 GW nuclear plant whose annual revenue exceeds $1B) the potential incremental gain from a custom RL controller easily dwarfs those costs. Conversely, fleets of modest assets, each worth only a few hundred euros of annual savings, cannot amortize per-site training. In that regime we must either reduce training costs (lighter simulators, transfer learning, automated hyperparameter search) or raise the per-policy value by deploying a single model across many devices. A single general policy is particularly appealing for stakeholder for its operational simplicity (software updates, monitoring, and validation happen once, not hundreds of times.), data leverage (experience collected on one site refines the policy for all others) and cost amortization (a single training run is spread over the whole fleet; per-site cost is affordable).

4.2 The example of microgrids

Microgrids fall in the second category: to be adopted by the business, the RL solution must be scalable. Our internal benchmarks show that a RL based controller can achieve higher performances on annual operating cost or carbon intensity compared to MPC methods. But as training a specific agent for every site would erase that margin, there is a need for developing more general policies.

Microgrids differ more than their common label suggests in the following aspects:

- *Resource mix* – microgrids can be composed of different components, for instance PV-only island resorts, off-grids villages with PV panels and fuel generators, or university campuses with PV panels and wind turbines.
- *Scale* – batteries ranging from 10 kWh cabins to 5 MWh industrial parks; peak loads spanning three orders of magnitude.
- *Main grid pricing* – pricing may vary depending on the country or region of the world, or on the contract chosen with the provider, with different energy market products, time-of-use tariffs, demand-charge penalties, or no grid at all.
- *Controllable assets* – some EMS can only manage battery charging and discharging, while others have the capability to control electric vehicle (EV) terminals or activate generators as well.

4.3 Generalization Challenges

Domain shift & heterogeneity A single policy must cope with multiple operating domains whose data distributions and dynamics differ, sometimes substantially, from those seen in training. Shifts appear along several axes: scale (battery size, power rating), behavioural regime (load patterns, weather, market signals), and structural constraints (tariffs, safety rules, ...). These variations induce covariate and concept shift: the same state features acquire new statistical properties, and the

environment itself may respond differently to actions. A policy that is optimal on one domain can therefore underperform or violate constraints in another domain unless it has been trained for, or can adapt to, the full spectrum of domains it will encounter.

Task diversity Each site can expose a different state vector (relative assets contained in the microgrid), action space (on/off generators, battery charging, EVs charging) and reward (cost, CO_2, outage). Aligning these tasks in a single Markov Decision Process (MDP) is non-obvious.

Specialization vs. generalization A general model risks leaving money on the table for atypical sites; over-specialized models are too expensive to maintain. Choosing the right point on this spectrum is a strategic decision.

4.4 Research Directions (Work in Progress)

The ideas below reflect the current state of our exploration rather than settled conclusions. They catalogue the hypotheses we are actively testing and the design choices we believe hold most promise Korkmaz (2024).

Instance clustering. A first idea to make this complex topic easier, is to cluster sites (Zhang & Yang, 2021) that "look alike", based on descriptive features or on a similarity distance (Dynamic Time Warping for timeseries for example), and train a single policy for each group. Clustering does not eliminate diversity, but it narrows the range of conditions any one agent must master, making training and validation more tractable without requiring a tailor-made controller per site.

Architecture-driven generalization. The choice of neural architecture can itself encode the inductive biases needed to span heterogeneous sites. Graph Neural Networks (GNNs), for instance, treat each generator, storage unit, and load as a node and pass messages along electrical or logical edges; because message passing is local and parameter-sharing is global, the same model scales naturally from a three-node microgrid to a hundred-node campus without zero-padding or retraining (Munikoti et al., 2023). More broadly, architectures that are permutation-invariant, order-agnostic, or modular can make generalization easier (Huang et al., 2020).

Unified MDP design. By rewriting state variables and actions in normalized or per-unit terms (e.g. relative to battery capacity or net load) we aim for a problem formulation that is similar at any scale, making one policy transferable across sites. For example, instead of feeding the neural network the production and consumption of every individual asset, we can aggregate these flows into a single net-load signal. This removes any dependence on the number of generators or loads. If a battery is the only controllable device, we can go a step further: divide the net load by the battery's maximum charge/discharge power and clip the result to $[-1, 1]$. The resulting dimensionless feature tells the agent what fraction of the current power imbalance the battery can offset, regardless of absolute ratings. Such normalization makes the state and action spaces invariant across microgrids of different sizes and compositions, and can ease generalization. This kind of modification requires a good understanding of the specific use-case: by creating a more general MDP there is a risk of losing useful site-specific information. Directly learning an invariant representation (Zhang et al., 2020) can also be promising.

Training method. The selection of the training strategy can also help generalization. One option is using a multi-task RL method (Teh et al., 2017), where a single network is optimized across many synthetic or historical sites drawn from a wide parametric prior; the shared weights encourage the discovery of invariants that re-appear in new environments. Complementary to that is meta-RL (Beck et al., 2023), which tunes the initial parameters so that only a handful of gradient steps are required to adapt to an unseen task. Robustness can be further promoted by heavy domain randomization and adversarial noise during simulation, while a gentle curriculum that progresses from simplified to fully realistic dynamics often stabilizes optimization.

5 Conclusion

In this paper, we have outlined several key considerations regarding the lifecycle and deployment of a real-world system, based on two concrete examples: one focused on microgrid management, and the other on agrivoltaics. We hope that this work will contribute to a broader deployment of RL, as well as to the sharing of best practices related to a technology with tremendous potential.

References

Habeel Alam and Nauman Zafar Butt. How does module tracking for agrivoltaics differ from standard photovoltaics? food, energy, and technoeconomic implications. *Renewable Energy*, 235: 121151, 2024. ISSN 0960-1481. DOI: https://doi.org/10.1016/j.renene.2024.121151.

Mohammed Alshiekh, Roderick Bloem, Rüdiger Ehlers, Bettina Könighofer, Scott Niekum, and Ufuk Topcu. Safe reinforcement learning via shielding. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Jacob Beck, Risto Vuorio, Evan Zheran Liu, Zheng Xiong, Luisa Zintgraf, Chelsea Finn, and Shimon Whiteson. A survey of meta-reinforcement learning. *arXiv preprint arXiv:2301.08028*, 2023.

Gabriel Dulac-Arnold, Daniel Mankowitz, and Todd Hester. Challenges of real-world reinforcement learning. *arXiv preprint arXiv:1904.12901*, 2019.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Wenlong Huang, Igor Mordatch, and Deepak Pathak. One policy to control them all: Shared modular policies for agent-agnostic control. In *International Conference on Machine Learning*, pp. 4455–4464. PMLR, 2020.

Ezgi Korkmaz. A Survey Analyzing Generalization in Deep Reinforcement Learning, 2024.

Sai Munikoti, Deepesh Agarwal, Laya Das, Mahantesh Halappanavar, and Balasubramaniam Natarajan. Challenges and opportunities in deep reinforcement learning with graph neural networks: A comprehensive review of algorithms and applications. *IEEE transactions on neural networks and learning systems*, 2023.

Yee Teh, Victor Bapst, Wojciech M Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning. *Advances in neural information processing systems*, 30, 2017.

Amy Zhang, Rowan McAllister, Roberto Calandra, Yarin Gal, and Sergey Levine. Learning invariant representations for reinforcement learning without reconstruction. *arXiv preprint arXiv:2006.10742*, 2020.

Yu Zhang and Qiang Yang. A survey on multi-task learning. *IEEE transactions on knowledge and data engineering*, 34(12):5586–5609, 2021.