Optimisation of Energy Consumption of Data Center using Artificial Intelligence

Léo Grill

▶ **To cite this version:**

Léo Grill. Optimisation of Energy Consumption of Data Center using Artificial Intelligence. Artificial Intelligence [cs.AI]. Université de Poitiers, 2023. English. NNT : 2023POIT2261 . tel-04227294

HAL Id: tel-04227294
https://theses.hal.science/tel-04227294v1

Submitted on 3 Oct 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

THÈSE de DOCTORAT

Pour l'obtention du grade de
DOCTEUR DE L'UNIVERSITÉ DE POITIERS
Faculté des sciences fondamentales et appliquées
Laboratoire de mathématiques et applications - LMA (Poitiers) UMR 7348 CNRS
(Diplôme National - Arrêté du 25 mai 2016)

Ecole Doctorale 651
Mathématiques, Informatique, Matériaux, Mécanique, Énergétique-MIMME

Mention : Mathématiques Appliquées
Discipline : Statistique

Présentée par :
Léo Grill

Optimisation of Energy Consumption of Data Centers using Artificial Intelligence

---

Soutenue le 23 Juin 2023

Devant le jury composé de :

Madame Béatrice Laurent-Bonneau Professeur, INSA de Toulouse	**Présidente**
Monsieur Vincent Vandewalle Professeur, Université Côte d'Azur	**Rapporteur**
Monsieur Alain Celisse Professeur, Université Paris 1	**Rapporteur**
Monsieur Hussein Obeid Maitre de conférences, Université de Caen Normandie	**Examinateur**
Monsieur Salah-Eddine El Adlouni Professeur, Université de Moncton, Canada	**Examinateur**
Monsieur Yousri Slaoui Maitre de conférences des universités (HDR), LMA de Poitiers	**Directeur de thèse**
Monsieur David Nörtershauser Orange Labs, Lannion	**Co-encadrant de thèse**
Monsieur Stéphane Le Masson Orange Labs (HDR), Lannion	**Co-encadrant de thèse**

Je dédie cette thèse à ma mère Violette, et à mon ami Chadi.

"Pour moi, c'est la définition même de la bêtise : croire qu'on représente l'intelligence."
-Jacques Rancière

Remerciements

Aux Dr. David Nörtershauser et Dr. Le Masson Stéphane, je tiens à vous remercier de m'avoir accueilli dans votre formidable équipe et de m'avoir guidé et montré la complexité du monde de la thermodynamique. Grâce à vous, j'ai acquis un savoir-faire inestimable tant sur le plan professionnel que personnel. Votre franchise, exigence et passion contagieuse m'ont permis de me développer et de me motiver. Je vous remercie de m'avoir pris au sein de l'équipe il y a quatre ans et de m'avoir supporté jusqu'au bout de ma thèse. Je remercie également Orange et plus précisment Orange Labs d'avoir financer mes travaux.
Je tiens également à remercier mon directeur de thèse, le Pr. Yousri Slaoui pour ses conseils tout au long de ce projet de thèse et pour m'avoir donné l'opportunité de travailler avec le laboratoire de Mathématiques et Applications de Poitiers. Sans toi, rien de tout cela n'aurait été possible. Je suis très heureux et honoré d'avoir été l'un de tes étudiants, et d'avoir pu profiter du partage de ta connaissance ainsi que de tout le temps que tu m'as consacré. Je ne pourrai jamais te remercier assez pour ta bienveillance et ta volonté d'aider et pour ton soutien moral et professionnel durant cette thèse. Tu as toujours été de bon conseil et de bon humeur ce qui a été un moteur important pour l'avancement de cette thèse. Je te suis très reconnaissant pour le temps conséquent que tu m'as accordé le long de mon parcours.

Je souhaite également remercier les membres du Jury pour avoir accepté de consacrer du temps afin d'évaluer mes travaux. Pr. Béatrice Laurent-Bonneau, c'est un très grand honneur pour moi que vous ayez accepté de présider le jury de ma thèse. Votre grande expérience en modélisation statistique honore mon travail. Je suis très reconnaissant au Pr. Vincent Vandewalle et au Pr. Alain Celisse pour l'intérêt que vous portez à mon travail. Je vous remercie d'avoir accepté de rapporter ce travail. C'est un grand honneur pour moi de pouvoir bénéficier de votre expertise. Pr. Hussein Obeid et Pr. Salah-Eddine El Adlouni c'est un plaisir pour moi que vous ayez accepté d'examiner ma thèse.
Je tiens à remercier tous mes collègues doctorants et docteurs au LMA. Je remercie Amine Ounajim, pour sa prévenance et sa gentillesse. Je remercie également Rachad Bentbib, Sahar Slama, Abir El Haj, Nesrine Chelbi, Oumaima Ben Mrad et tous les autres doctorants que j'ai eu l'honneur de rencontrer au LMA.

Et pour garder le meilleur pour la fin. Je tiens à remercier toute ma famille, en particulier mon épouse, qui a toujours été là et m'a beaucoup soutenu et encouragé tout au long de la thèse. Je remercie mon père, pour m'avoir toujours soutenu dans les moments difficiles. Je ne remercierai jamais assez mes frères et soeurs.

Résumé

Cette thèse explore la manière dont l'intelligence artificielle peut diminuer l'énergie consommée dans les centres de données. L'intelligence artificielle est un vaste domaine de recherche très populaire. Ce domaine est souvent idéalisé, mais comment les nouveautés de la recherche peuvent-elles être appliquées à des cas d'utilisation réels ? Entre considérations physiques, théories mathématiques et technologies informatiques, cette thèse utilise différents domaines pour aider à déployer et améliorer les technologies récentes afin de répondre aux défis énergétiques actuels.
La première partie de la thèse évalue les problèmes énergétiques et les technologies déployées dans les centres de données et les bâtiments de télécommunications. Les infrastructures d'information et de communication sont de gros consommateurs d'énergie et nécessitent des systèmes de climatisation spécifiques en raison des conditions de travail du matériel informatique. L'optimisation des systèmes de climatisation et de leur consommation est une préoccupation majeure dans la réduction de la consommation d'énergie. La deuxième partie de la thèse explore l'apprentissage statistique et probabiliste pour optimiser la consommation d'énergie. Elle se concentre principalement sur les modèles d'apprentissage par renforcement profond (Deep Reinforcement Learning) pour la prise de décision automatisée basée sur la méthode axée sur les données. L'apprentissage profond est flexible dans la modélisation et peut s'adapter à de nombreux problèmes, ce qui est pratique lorsqu'une méthode doit être généralisée.
La dernière partie décrit la mise en œuvre et l'application dans des environnements simples afin de soulever et de traiter les problèmes courants et de discuter de la manière d'adapter le modèle à la réalité. On sait qu'un grand nombre de projets de science des données sont très prometteurs mais ne sont pas mis en œuvre. Ce travail vise à faire un pas en avant dans l'introduction de l'IA dans des environnements sensibles tels que les centres de données.

Abstract

This thesis explores how Artificial Intelligence can diminish the energy consumed in Data centres. Artificial Intelligence is a vast research area that is very popular. This domain is often idealised, but how can the research novelties be applied to actual use cases ? Between physical considerations, mathematics theory and computer science technologies, this thesis employs various fields to help deploy and improve recent technologies to address current energy challenges.

The first part of the thesis assesses the energy issues and the technologies deployed in Data centres and telecommunication buildings. Information and communication infrastructures are massive energy consumers and need specific air conditioning systems due to the working conditions of IT material. Optimising air conditioning systems and their consumption is a primary concern in reducing energy consumption.

The second part of the thesis explores statistical and probabilistic learning to optimise energy consumption. It mainly focuses on Deep Reinforcement Learning models for automated decision-making based on the data-driven method. Deep learning is flexible in modelisation and can suit many problems, which is convenient when a method must be generalised and industrialised.

The last part describes the implementation and application in simple environments to raise and deal with common issues and discuss how to scale the model to reality. It is known that a lot of Data Science projects have great promises but fail to be implemented due to practical difficulties. This work aims to make a step in introducing AI in sensitive environments such as Data Centres.

Scientific Contributions

CMStatistics 2022 — Décembre 2022
15th International Conference on Computational and Methodological Statistics, London, UK.

Rencontres des Statisticiens et Probabilistes — Novembre 2022
Modélisation et Algorithmes Stochastiques, Université de Sousse, Tunisie.

COMSTAT 2022 — Juillet 2022
24th International Conference, Bologna, Italy.
Computational Statistics.

Apprentissage Statistique et Applications — 22 au 24 Juin 2022
Workshop, Poitiers.

SDFS — Juin 2022
53èmes Journées de Statistique de la SFDS, Lyon, p. 128.

EcoSta 2022 — 4 au 6 Juin 2022
5th International Conference on Econometrics and Statistics, Kyoto, Japan.

Journée Colloquium — 2021
Modélisation dans l'océan indien, JMOD'21 sur Mayotte.

Publications

Deep Bayesian Reinforcement Learning for Air Conditioning System with Inertia	Currently being finalised
Energy and Buildings	2023
Bayesian Neural Network for Policy Search	Submitted
Artificial Intelligence	2023
Bayesian Actor for Deep Reinforcement Learning	Published
53ème Journée de Statistique de la SFDS, Lyon, p. 128.	2022
L'Intelligence Artificielle, un enjeu pour réduire les coûts énergétiques des Data Centers	Published
Microscoop	2021

Table des matières

1 Introduction **23**
- 1.1 Data Centres . . . 23
  - 1.1.1 Data Centres and Energy Consumption . . . 23
  - 1.1.2 Data Centres: Introduction . . . 24
  - 1.1.3 Data Centre Standards . . . 25
- 1.2 Cooling Systems and Management . . . 28
  - 1.2.1 Physical Considerations . . . 28
  - 1.2.2 Cooling Technologies . . . 29
  - 1.2.3 Artificial intelligence for System Control . . . 31
  - 1.2.4 Implementation of Artificial Intelligence Systems . . . 33
  - 1.2.5 Existing Framework and Environment . . . 34
  - 1.2.6 Contribution . . . 35

2 Bayesian Actor in Deep Reinforcement Learning **37**
- 2.1 Introduction . . . 37
- 2.2 Preliminaries . . . 38
  - 2.2.1 Dynamic Optimization and Markov Decision Processes . . . 38
  - 2.2.2 Deep Learning . . . 39
  - 2.2.3 Bayesian Probability . . . 40
  - 2.2.4 Bayesian Deep Learning . . . 40
- 2.3 Reinforcement Learning . . . 42
  - 2.3.1 Goals and Returns . . . 43
  - 2.3.2 Reinforcement Learning Methods . . . 45
  - 2.3.3 Policy Gradient . . . 49
  - 2.3.4 Parameters and issues . . . 51
- 2.4 Deep Bayesian Policy Search . . . 52
  - 2.4.1 Bayesian Deep Policy . . . 53
  - 2.4.2 Bayesian Deterministic Policy Gradient . . . 55
  - 2.4.3 Implementation . . . 57
- 2.5 Results and Discussion . . . 57
  - 2.5.1 Environment . . . 57
  - 2.5.2 Compared Models . . . 58
  - 2.5.3 Results . . . 60
  - 2.5.4 Limitation and perspective . . . 60
- 2.6 Conclusion . . . 61

2.7 Appendix . . . 62
2.7.1 Proof of Theorem 3 . . . 62
2.7.2 Likelihood and Compression Cost . . . 64

3 Application to Data Center Simulation **67**
3.1 Introduction . . . 67
3.2 Related Work . . . 68
3.3 Numerical simulation . . . 68
3.4 Physical Nodal Environment . . . 76
3.4.1 Reward . . . 78
3.4.2 Baselines . . . 79
3.4.3 Inertia and differential impact . . . 80
3.5 Results and Discussion . . . 86
3.5.1 Discussion . . . 86
3.5.2 Deploying a model in actual conditions . . . 87

4 Conclusion and Perspective **93**
4.1 Conclusion . . . 93
4.2 Perspective . . . 94
4.2.1 VRAE/LSTM . . . 94
4.2.2 Bayesian Methods . . . 94
4.2.3 Offline Methods . . . 95
4.2.4 Non-Parametric Recursive Regression . . . 95

Abbreviation and Notation

Abbreviation

DC	Data Centre
AI	Artificial Intelligence
ML	Machine Learning
RL	Reinforcement Learning
DL	Deep Learning
DRL	Deep Reinforcement Learning
MLP	Multi-Layer Perception
ANN	Artificial Neural Network
BNN	Bayesian Neural Network
SGD	Stochastic Gradient Descent
VI	Variationnal Inference
MDP	Markov Decision Process
POMDP	Partially Observed Markov Decision Process

Mathematical Notations

$\mathcal{A}$	Action space
$\mathcal{S}$	State space
r	A reward function
R	Cumulative rewards
a	Action
s	State
π	A stochastic policy
$\pi(a\|s)$	The probability of getting the action a, knowing the policy π and the state s
μ	A deterministic policy
$\mu(s)$	The action generated by the policy μ, for the state s
$V^{\pi}(s)$	Value function for the state s under the stochastic policy π
$Q^{\pi}(s,a)$	State-action value function for the state s and action a under the stochastic policy π
$V^{\mu}(s)$	Value function for the state s under the deterministic policy μ
$Q^{\mu}(s,a)$	State-action value function for the state s and action a under the deterministic policy μ
$X \sim p$	Drawing X from the distribution p
$\mathbb{E}[X]$	The expectation of the random variable X. If X is continuous then $\mathbb{E}[X] = \int_x p(x)x$ if it exists.
$f\colon \mathcal{A} \to \mathcal{B}$	Function f from elements of set $\mathcal{A}$ to element set $\mathcal{B}$

Physical Notations

$\mathcal{Q}$ the heat transfer rate [J]

$\mathcal{W}$ Amount of work transferred [J] nominally in the form of electricity or mechanical work

m Mass flow [kg]

E Energy stored in the system [J]

e Energy transported per unit mass flow [J/kg]

$\dot{f}$ The time derivative of f if it exists

List of Figures

1.1 DCs servers room with cold air distributed through the raised floor. The picture was taken in Orange's DC in Val De Reuil, in 2013. 24
1.2 Example of architecture for air cooling and distribution for a DC. 25
1.3 Representation of energy consumption in DCs [73] 26
1.4 Air condition advised by ASHRAE for IT material [100] 27
1.5 The norm given by ETSI for "Temperature-controlled locations" such as DCs or telecom centres. 28
1.6 Representation of the energy conservation on a system with mass exchanges. 29
1.7 Introducing a decision model to drive the distribution and conditioning systems. The decision model's purpose is first to deal with the distribution system to transfer the heat smartly by anticipating the system evolution. Then limit the cold and humidity production by mastering the risk. 32

2.1 The reparametrization trick used a random variable and two parameters to generate a random parameter. The relationship between the parameters μ, ρ and θ is deterministic. 41
2.2 The interaction of an agent and an environment in a RL context. 43
2.3 Learning a policy to produce an estimate with Monte Carlo setting. The estimate of the price every month is updated giving the final price. 44
2.4 Learning a policy to produce an estimate with TD-learning setting. The estimate of the price every month is updated giving the next month's price. . 46
2.5 The on-policy learning architecture of an actor-critic algorithm. The agent interacts with the environment and stacks the information generated in a data history. The tuple state-action-reward-state usually encodes the trajectory of the system. The critic maps state-action pairs to the Q-value based on a supervised estimation. The data history is a buffer with a capacity. When it is complete, the new data overwrites the oldest one. 50
2.6 Sampling policies encoded neural networks from a Bayesian neural network. We obtain standard neural networks by drawing the whole parameter set from the distributions. 52

3.1 Simple continuous environment for RL, the point is to train an agent to move the ball to the highest position. We maximise the reward when the ball is uphill. 69

3.2 Learning of the Bayesian Neural Network agent in our plain environment. The left part represents the agent and the policy generated for the exploration, while the right part represents the gain estimated by the critic depending on the states and actions. 70
3.3 Learning of the Bayesian Neural Network on our plain environment for alpha equals 10. 71
3.4 Learning of the Bayesian Neural Network on our plain environment for alpha equals 0. 72
3.5 Comparison for different α. We ran one hundred runs and averaged the rewards. The best configuration is for $\alpha = 0.00001$. 73
3.6 The trajectories of the rewards and averaged reward of one hundred runs for different α. The last figure compares the average reward for the different α values. The best value is 10^{-5}. For α equals, some runs don't converge, while for $\alpha = 10^{-5}$, all the runs have converged. 74
3.7 The trajectories of the Bayesian Neural Network parameters during the learning for 30,000 steps. A parameter's horizontal movements alter the parameter's mean, while vertical movements alter the deviation of the parameter. We can observe the impact of α and the η_ρ of the prior on the parameters trajectory. 75
3.8 Schema of a ventilated room with a thermal load. 77
3.9 The optimised ventilation based on the parametric exploitation of the temperature delta. This model ventilates when the temperature difference between outside and inside is negative, and the room requires cooling. 80
3.10 The impact of wall material on the temperature regulation system of a room. The unseen nodes of the model are represented in the figure and show how the material's characteristics alter the temperature behaviour. 82
3.11 These policies show the impact of inertia on the state transitions of the system. The environment is not Markovian, as we can observe a policy that is not cooling over the night but lowering the temperature before the temperature starts increasing won't succeed in keeping the temperature in the authorized range as the temperature increases faster than the policies on the two other policies 83
3.12 The directed acyclic graph (DAG) represents the transition in a partially observable Markovian decision process. We assume the reward is computed only from the state and action at the same timestep, even if other setups exist. The arrows represent conditional dependencies of the variables 84
3.13 The states of our environment with time and state dependence transitions. The environment has three states with associated rewards. 85
3.14 Comparison of the system's action profile shows that coherent schemes of exploration are necessary to explore the intern states of the system correctly. High-frequency action variations are not perceptible on the intern nodes. In the left top figure, the behaviour of the intern nodes is very close to the one in the right top figure, while the last figure shows variation in the intern nodes. 88

3.15 We compare the baseline models to the policies found with DRL. The airflow was scaled by a factor of ten. The last figure represents the policy using colour mapping. From the outside and inside temperature to the action chosen. From this figure, we observe when the model would choose to ventilate. The results for the consumption are in Table 3.1 89
3.16 Data used to add uncertainty and get closer from actual conditions to the model. 90
3.17 In this figure, we can observe how time impacts the policy. It shows the model's ability to adapt the policy for the moment of the day and anticipate. 91

List of Tables

2.1 Estimated rewards from two hundred runs for different state-of-art agents and our agent on the environments of HalfCheetah-v3, LLC-v2 and MCC-v0 . . . 60

3.1 The DRL model performs better than the baseline optimized for this problem. We have a gain of 17% for the plain baseline and 7% for the optimised baseline. None of the models gets out of the authorized range. 86

Chapitre 1

Introduction

1.1 Data Centres

1.1.1 Data Centres and Energy Consumption

In the current energy, environmental and social context [2], diminishing energy consumption is a significant issue for the population's well-being. Data and network usage around the world are growing exponentially [1]. Energy consumption is following the trend. This thesis aims to explore AI and develop tools to diminish the energy consumption of DC and similar buildings such as telecom centres. In 2009 [28], DCs consumed 200 TWh or nearly 2% of the world's electricity. In 2007, in Western Europe, the DCs consumed 56 TWh and information and communication technology was estimated to account for 10% of total UK electricity consumption. At the same time, France consumed a total of 480 TWh. This sector is growing by 10 to 15% per year.
Moreover, with the arrival of 5G, the number of connected devices is expected to reach 50 billion by 2020, generating 2.5 quintillion bytes of data per day [97]. DCs should increase as they are part of the infrastructure related to implementing these networks. The actors of these transformations must anticipate these new needs, especially if they wish to respect their societal commitments, such as the net zero carbon commitment by 2040 of Orange. Equipment and infrastructures must evolve and use new techniques to fulfil this goal. The issue of energy optimisation in DCs makes it possible to reconcile the ever-increasing digital technology needs while limiting energy costs and environmental impacts. Developing more advanced thermodynamic techniques increases the efficiency of DC cooling systems, such as liquid cooling. The cooling systems in DCs consume around 30% of the overall power consumption [14]. However, these improvements require orchestration to perform optimally. Given the systems' complexity and diversity [69], artificial intelligence demonstrates great potential concerning the control of cooling systems and usage in building energy prediction [95]. More complex models, such as RL, can be considered. For example, a study [51] has shown that a model could achieve a 9% energy saving on server cooling, which is particularly effective when these servers receive an exceptionally high load. However, those methods must be deeply studied to be applied in actual conditions.
In this thesis, we focus on the use of AI to improve the efficiency of the DCs cooling system. To explore how AI can reduce energy consumption, we must understand how DCs work and

the physical issues raised by maintaining high computation inside buildings. This is why the first part of the thesis describes the DCs and their physical behaviour and issues. The complexity of the task does not allow us to explore and describe the phenomenon deeply, but introduces the main ideas and gives context to the choices made in the two following parts. The purpose is to provide hints and intuitions on the main concerns and why we focus on RL technologies. The application motivated the theoretical results as the models were developed to tackle the issues encountered while optimising.

1.1.2 Data Centres: Introduction

DCs do not look like ordinary buildings and usually have no windows and minimal air circulation. They come in various sizes, from small rooms to large buildings. DC rooms contain racks that contain servers, storage, and networking equipment. DCs gather IT materials with high density. It is different to individual uses of computers. There are many uses for DCs. They usually centralise data storage, applications and cloud computing. They are essential for developing big data in various fields: commerce and business, research, and society administration [67].

FIGURE 1.1 – DCs servers room with cold air distributed through the raised floor. The picture was taken in Orange's DC in Val De Reuil, in 2013.

There are many configurations for DCs, but one of the most effective is to place the racks in an alternating fashion, making cold aisles through which refrigerated air enters the shelves

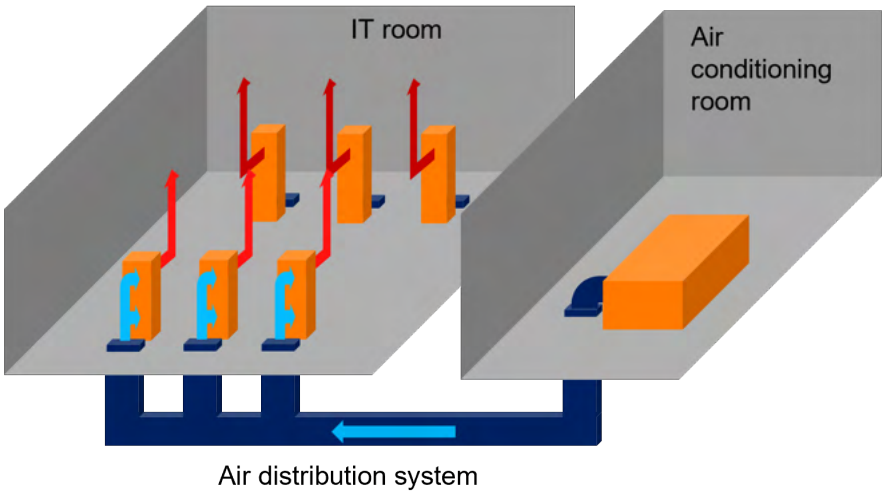


FIGURE 1.2 – Example of architecture for air cooling and distribution for a DC.

and hot aisles through which heated air is evacuated. In DCs, electrical power supplies the equipment. A power unit receives current without interruption and distributes the load to the equipment. However, the electricity used in DCs is almost wholly transformed into heat (Joule effect), so it is necessary to cool them to avoid material deterioration. DCs often have air conditioning units containing fans, filters, and air cooling systems. Usually, these systems use a chilled water unit to cool water before serving it to the IT equipment through hatches on the floor. IT materials also require a specific range of hygrometry to function correctly. There are climate conditions adapted for optimising the performance of the equipment, and there are ranges to keep them in and avoid deterioration. Some organisations provide standards for these ranges.

1.1.3 Data Centre Standards

ASHRAE

In fact, it is essential to keep computer equipment in a suitable condition to function correctly and safely. Normalisation of metrics and evaluation of the efficiency of IT DCs conditioning is essential to compare and evaluate. For this purpose, ASHRAE, the American Society of Heating, Refrigerating and Air-Conditioning Engineers, advises keeping computer equipment in a temperature range of 18 to 27°C. The organisation furnishes guidelines [35] with allowed values for equipment depending on its type. There are six equipment classes (A1 to A4, B and C). These classes separate the equipment according to their constitution. The new equipment is more likely to be A4 than A1, so they have more permissive ranges. These standards make it easier to save energy by adopting behaviours that are more concerned with the equipment in place.

From literature benchmark [35], we find that fans are big energy consumers in DCs among the three major power consumers in HVAC (Heating, ventilation, and air conditioning) systems: fans (39%), chillers (39%), and cooling water pumps (18%). This requires consideration of

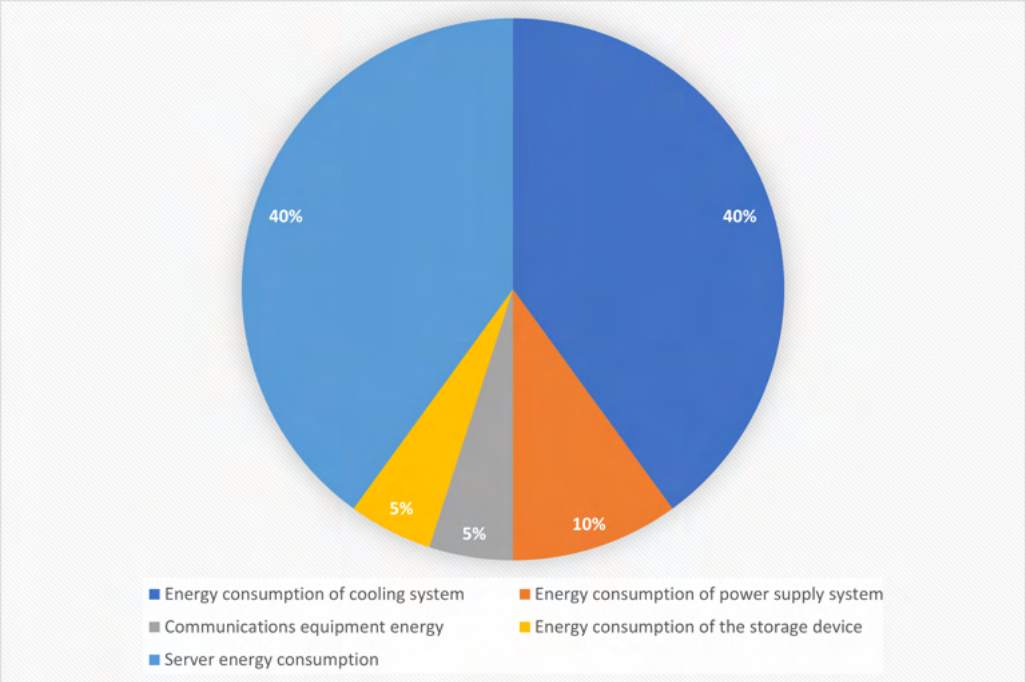


FIGURE 1.3 – Representation of energy consumption in DCs [73]

the optimum conditions for the equipment. Some equipment manufacturers specify a wider window of inlet air temperatures for the safe operation of their equipment. Higher inlet air temperatures in DCs, beyond an optimum value, increase the overall power consumption due to increased power dissipation in many parts of the overall multi-scale system, for example, from the chips and server fans [65]. In this regard, the 2011 ASHRAE guidelines suggest that increasing the inlet air temperature from 15°C to 35°C could result in a 7 to 20% increase in the power consumption of IT equipment due to increased fan speeds.

ETSI

The European Telecommunications Standards Institute (ETSI) defines the standards for telecommunications. It is a non-profit organisation that produces telecommunications standards for the present and the future [25]. This organisation acts for telecommunications and is interested in DCs, which are very energy intensive, and it gives recommendation ranges for telecom equipment. ETSI furnishes standard conditions for telecommunications equipment [24]. This diagram is given by ETSI (see Figure 1.5) frames the conditions in which telecom equipment can operate without degrading, considering the equipment in DCs. It is advisable to stay in the grey area. However, it can leave the grey area and go into the solid area if it is less than 10. % of the time. The dotted line area is the exceptional area where the material cannot exceed 1% of the time. Going outside these limits could damage the equipment. To limit the impact of energy, ETSI gives guidelines for telecom hardware but also proposes other measures, such as implementing efficiency indicators.

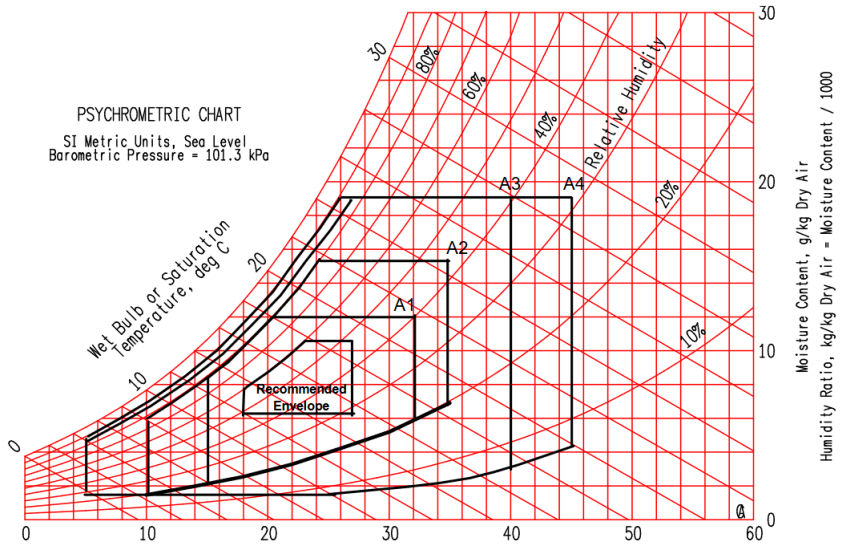


FIGURE 1.4 – Air condition advised by ASHRAE for IT material [100]

Data Centre Efficiency Measures

As said previously, improving the efficiency of DCs needs reference values for comparison. Different metrics have been studied to evaluate DCs' performance [97]. Therefore, there are indicators, such as PUE (Power Use Effectiveness), which corresponds to the total energy consumption of the centre divided by the energy consumption of IT equipment. The closer it is to 1, the more efficient it is.

$$\text{PUE} = \frac{\text{total energy consumption of the centre}}{\text{IT equipment energy consumption}}.$$

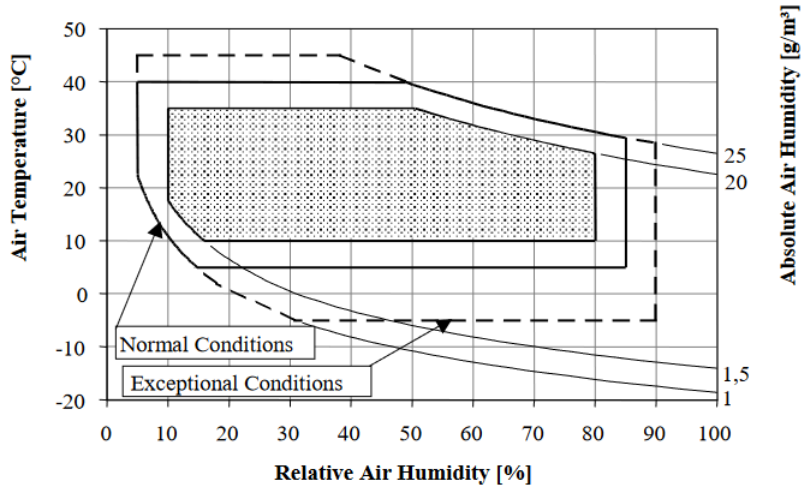
These indicators will allow us to compare the operation of the different centres and the methods used. They are essential tools for setting up studies on the energy efficiency of DCs.

The ETSI also proposes its indicators [23] to compare DCs, the DCEM, which uses four indicators of lower granularity. The DCP indicates the level of energy performance of the DC by comparing the power consumption of the IT and telecom equipment with the total power consumption. According to their power consumption, the Gauge (DCG) classifies DCs into four sizes - from S to XL. REUSE considers the reused calories, and REN the share of renewable energy used. The DCEM has the advantage of measuring energy consumption while considering the efficiency of the tasks performed, the use of renewable energy, and the reuse of the energy consumed.

Indicators have several advantages:

1. Compare different DCs.
2. Get the evolution of the indicator to assess improvement.
3. Detects undesired behaviour that affects the efficiency of DCs.

When dealing with air conditioning systems, as they do not participate in the computations of centres, we can minimise their energy consumption to maximise DCs' efficiency.



Values outside this field have a probability of occurrence of less than 10 % (see IEC 60721-3-0).

Exceptional climatic limits.

Normal climatic limits: Values outside these limits have a probability of occurrence of less than 1 % .

NOTE: Exceptional conditions may occur following the failure of the temperature controlling system. This is described as 3.1E in the tables but it should be noted that there is no separate class 3.1E.

FIGURE 1.5 – The norm given by ETSI for "Temperature-controlled locations" such as DCs or telecom centres.

1.2 Cooling Systems and Management

1.2.1 Physical Considerations

Basic knowledge of thermodynamic concepts helps to understand and tackle the modelling issues of DCs and equivalent systems. Using the following notation:

$\mathcal{Q}$ the heat transfer rate [J]

$\mathcal{W}$ Amount of work transferred [J] nominally in the form of electricity or mechanical work

m Mass flow [kg]

E energy stored in the system [J]

e Energy transported per unit mass flow [J/kg]

The first law of thermodynamics yields energy conservation for a closed system, leading to the following equation, :

$$\Delta E = \mathcal{Q} - \mathcal{W}. \tag{1.1}$$

With Δ indicating the changes. And with mass flow entering and quitting the system, we have:

$$\dot{\mathcal{Q}} - \dot{\mathcal{W}} + \sum_i \dot{m}_i e_i - \sum_o \dot{m}_o e_o = \frac{dE}{dt}. \tag{1.2}$$

Where e_i is the energy transferred per unit mass flow to the system while e_o is the energy out. We consider Equation (1.2) to represent a DC room with airflow entering and exiting the room to ventilate and evacuate the heat.

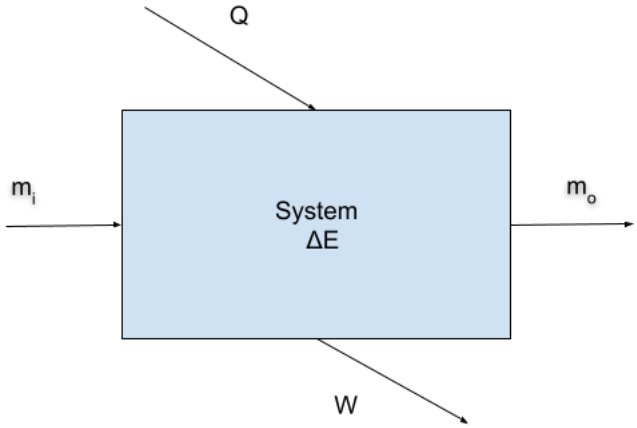


FIGURE 1.6 – Representation of the energy conservation on a system with mass exchanges.

1.2.2 Cooling Technologies

It is possible to use more efficient cooling techniques to reduce the energy impact of DCs. The use of free cooling, for example, is a convenient solution in terms of power consumption. Research is currently also looking toward liquid cooling. The control of the cooling system is also an area of improvement. Control techniques such as PID control (proportional, integral, derivative) are currently used. The use of AI for this type of problem is getting much more attention [73]. The method for reducing energy consumption can be divided into the following groups:

1. Hardware efficiency
2. Software efficiency
3. Strategy and Management

Hardware efficiency concerns the chips in place and other CPU/GPU performances. Software efficiency is dealing with virtualisation, dividing the load between the servers. The last group concerns Strategy and Management. In this section, AI has a significant role to play. It may help make predictions to anticipate. To detect anomalies to prevent material failures.

Free cooling

Free cooling is a technique that uses available natural cold sources to chill DCs, reducing the energy consumption of air cooling. Free cooling has many possible configurations, and its benefits depend on the outside air conditions. Free cooling did not develop in the DCs until 2008 when ASHRAE gave more flexible climate ranges for equipment, allowing for greater use of outside air. Free cooling of DCs can be divided into three categories [102]: airside-free cooling, waterside-free cooling and heat pipe-free cooling. Among the three categories of free cooling systems, a heat pipe system has good energy efficiency and cooling capacity due to its ability to transfer heat at small temperature differences without external energy. In addition, it has no disturbance in the indoor environment and can be integrated with compression systems. Although it has the shortest history, it shows excellent application potential.

Liquid cooling

The production of cold is very energy-consuming, and using air as a heat transfer fluid is not the most efficient solution. Liquids offer a promising alternative with higher convective heat exchange coefficients. One possible approach to reduce power consumption is applying heat transfer to the closest to the equipment. Like free cooling, there are many approaches with their advantages and inconveniences [37], we have spray cooling, pool boiling, heat pipe cooling, and two-phase cooling. One significant advantage of water cooling is focusing directly on the chips. The electronic components can be immersed in mineral oil to keep them at a proper temperature. The results are satisfactory in terms of power consumption. However, there are still some elements to consider, such as cooling the processor. The use of heatsinks dedicated to cooling by immersion in viscous liquids can be improved. The use of liquid cooling has higher maintenance costs than plain free cooling and is more complex to implement. Those technologies are more or less effective depending on the objective and if we consider the long or short term. The efficiency of liquid cooling technologies is often bounded by other technologies, leading to its application mainly in recent and high-performance computing DCs.

Data Centre Management

Different studies show that the DC air condition system management can be seriously improved, as shown in different studies [105], [103]. In most cases, the cooling models of DCs consider a hot source and a cold source. The goal is to maintain the hot source at an acceptable temperature using a classical control, usually PID (proportional, integral, derivative). The cold air is distributed in the cold aisles and does not consider the bays individually. From this observation, we could consider using a more precise air-conditioning system to individually cool the racks with variable opening hatches. For example, this would allow adapting the cooling. However, such optimisation requires controlling many parameters, controlling the blower units, and opening the different hatches while monitoring each rack. The choice of these parameters is an optimisation problem under constraint. It is a question of minimising power consumption while ensuring that the material's environment (temperature, hygrometry) remains adapted to its operation. The ideal purpose is to implement a DC orchestration system based on agents capable of processing large volumes of data. These

agents make predictions based on the DC activity. They collect a maximum amount of data (several sensors) and use external data, such as the weather, to carry out an optimised action plan. The goal is to cool the DC more efficiently and perform predictive and monitoring maintenance to reduce maintenance costs. The implemented infrastructure allows the sharing of information to realise different tasks (cooling optimisation, server activity optimisation, monitoring) related to the maintenance of the DC and decrease the generated costs. This thesis focuses on the air conditioning aspects. The solution could go further by integrating DC software optimisation with the management of the computing structure [20].

1.2.3 Artificial intelligence for System Control

AI has grown significantly in recent years with advances in computer hardware and the emergence of Big Data. The AI market for enterprise applications is estimated to be worth over $36 billion by 2025 in AI compared to $643 million in 2016. Intelligence is a human concept that can be defined with [27]: "Intelligence is that faculty, of mind, by which order is perceived in a situation previously considered disordered." This seems incompatible with artificial intelligence, as observed: "Thus machines may have intelligent-like performance but cannot be intelligent because the machine's instruction set is a proper subset of man's instruction set." But intelligence is a broad concept with different meanings. An interesting concept is intellectual emancipation [72], the point is for the machine to emancipate itself from humans by operating in an environment with the least human guidance possible. Even if intelligence was defined for living creatures when it comes to machines, the meaning of artificial intelligence is not a consensus and depends on the context [94]. The main idea is to use cognitive abilities to structure information by censoring, memorising, and learning.

1. Information structuring.
2. Intellectual emancipation.

Those two considerations are a way of judging the intelligence of an entity, the ability to gather data to answer a question, and the ability to do it a minimal instruction. However, our point is not to make philosophy or neuroscience but to give a starting point and understanding of the methods used and their limits. In data science, "artificial intelligence" usually describes a set of concepts inspired by human cognition or the biological brain and intended to assist or replace the individual in processing massive amounts of information [8]. AI research has many domains associated: automatic learning, knowledge representation, reasoning modelling, uncertainty management, constraint satisfaction, planning, heuristic search, autonomous agents, multi-agent systems, semantic web, automatic language processing, robotics, vision, pattern recognition, cognition modelling, neuro-informational systems. Using AI models to optimise physical systems needs to formulate an approach. Most AI algorithms use Data to solve a problem. The first step is to formulate the issues and choose the best mathematical formulation for the appropriate models. We seek to orchestrate the system in the best possible way. We want to provide the optimal configuration of the system in real-time. To do this, we need to characterise it as well as possible while identifying the levers. These levers will allow acting to minimise energy consumption while respecting the constraints of the material. Therefore, it is interesting to study the models used for energy-related systems and those used for system control.

From a data science perspective, using "machine learning" is more appropriate to designate the training of models with data. ML uses specific models to perform supervised and unsupervised learning and statistical tools to analyse the results. Supervised learning imitates the relationship provided in data to make predictions or estimations, either regression if a quantity is estimated or classification if the estimation is a category. Unsupervised learning uses the data to separate and find patterns in the data. A basic example would be to have a set of unlabelled images of dogs and cats and to use an algorithm to separate the images. The main difficulty of this task is the poorly defined objective. Various fields use ML tools with different approaches [6].

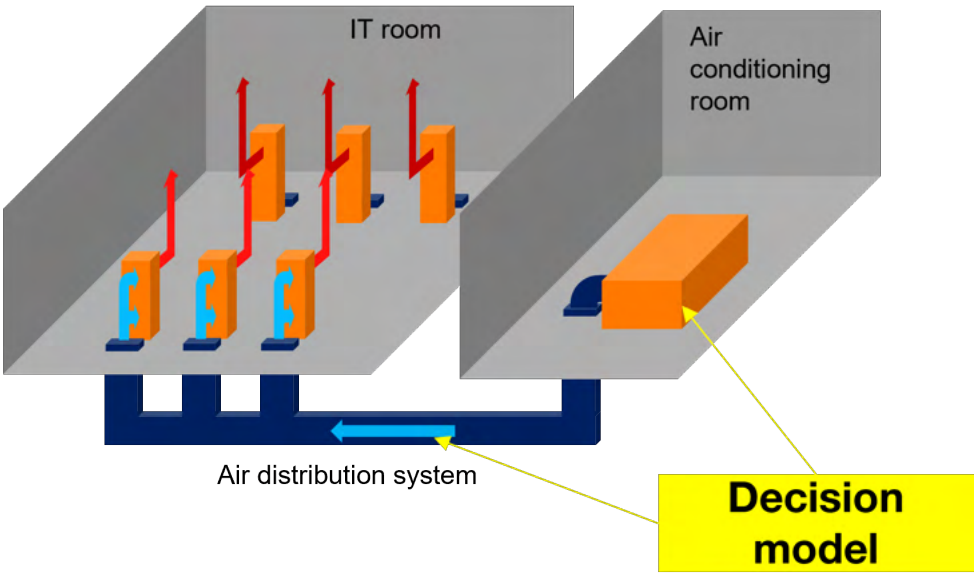


FIGURE 1.7 – Introducing a decision model to drive the distribution and conditioning systems. The decision model's purpose is first to deal with the distribution system to transfer the heat smartly by anticipating the system evolution. Then limit the cold and humidity production by mastering the risk.

To deal with the physical phenomenon happening in DCs, we use a digital twin [88], established on the information collected with sensors. The placement of these sensors is essential. The sensors allow us to assess the temperature and the hygrometry of the data-processing material to check if it is well in the imposed standards. Then, sensors are needed to understand the phenomena occurring in the DC (air flow, solar and wind impact) to control its evolution and thus have all the elements necessary to develop a management model. DC modelling may require thousands of sensors using all available data, such as server loads. Thus, this modelling allows for controlling the air conditioning and managing the DC's security by observing and predicting the failures. DC cooling management is a sensitive subject. A defect in the cooling system can lead to severe consequences, which is why explaining the models put in place and highlighting abnormal behaviour is necessary. In France, 24% of the cost of a DC is related to maintenance. Thus, it is possible to monitor the reliability of systems. Graphical probabilistic models are particularly well suited to this

problem. The usual paradigms of supervised and unsupervised learning are inadequate for solving our task. We choose to explore other tools based on statistical and probabilistic theory. The evolution of an algorithm in an environment such as a DC is suitable for reinforcement learning algorithms. Indeed, we update the model during its use: it performs learning by interaction.

Models used in the literature

Concerning predicting energy-related behaviours, many studies have been conducted. The main models used are neural networks, SVR (Support Vector Machine Regression) [96], multiple regressions and dynamic Bayesian networks [7]. And that the use of ensemble methods has encouraging results in the field. The purpose of these models is to predict values such as temperatures and energy consumed. To diminish the costs, strategies have to be implemented based on those predictions keeping men in the loop. More complex modelling must be considered to automate the system entirely with Artificial Intelligence. Machine learning is spread to model variables' behaviour and anticipates consumption or temperature peaks. One major misconception about modelling and ML is how it transforms Data, it transforms the shape of the Data more understandably. A model, at some point, gives a particular knowledge, knowing the system behaviour from past data and its current state. The current state is the model's input, and the behaviour knowledge is encoded in the learned parameters of the model. Recycling prediction as input of a second model exploits the error of the first model. This is why the complexity scales up quickly when modelling more complex tasks. Graphical models such as Bayesian or hidden Markov networks are commonly used in biology, aerospace and finance to perform prediction, simulation or control tasks. These methods are easily interpreted because they combine particularly well with business expertise and work well for dynamic systems (evolution in time). This algorithm can also be used to control asynchronous systems: having action delays between the activation of the actuator and the impact on the system. This can correspond to the time between the activation of the fans and the expected effect and consider disturbances. Particle swarm optimisation is one of the algorithms used to optimise systems with actors acting in parallel. They are used to manage the allocation of resources to servers.

1.2.4 Implementation of Artificial Intelligence Systems

When applying models to physical systems, the application needs coherent implementation development. The model has to be able to furnish decisions from data observation. Regularly the model produces instructions for the system's actuators. A model must be easy to interface with different systems. Training and deploying models require computer science skills. Using tools like National Instrument [22], [29]. Most equipment comes with its software. We must deal with communications when creating a solution integrated into a functioning ecosystem. It requires developing interfaces and creating a new potential source of errors. The model should use streaming Data to optimise the DC efficiency. It has to anticipate to avoid latent effects. From the learning perspective, we often have Data about our system. The model ideally starts learning from this Data and finishes the training online. For modelling DCs, having a good Data management setup is very important. It needs some software, Data engineering, and statistical modelisation skills. The Data transformation from the sensors

to the instructions sent to the air conditioner system must be integrated. Keeping in mind the whole pipeline and getting close to the application is a must for avoiding standard modelisation errors, like using Data that can't be used, gathering the Data with the shape adapted for the model, or ensuring the time steps are fixed. If the learning is in two steps, an offline and an online, then the Data must be the same in the two stages. These considerations might seem trivial but are often harder to consider and can lead to a failure of the application of proof of concept. As explained in the article [63], the challenges result from going from theory to practice. Deploying the models in production requires legal considerations. Our work considers the deployment until the proof of concept on actual Data. Considering the company ecosystem and workspace is necessary when working in a company. Often it can harden the implementation of recent research algorithms for security purposes. It means most recent algorithms must be reimplemented with the risk of errors. We implement our algorithm to wholly control and customise the behaviour. This introduces the risk of making errors in the implementation.

Computer Science in Artificial Intelligence Application

Applying statistical modelling or ML to a specific domain requires exploiting computer science skills, especially for physical applications. Physical applications consider theoretical values such as temperatures and hygrometry that can be measured and changed over time. Then we have to collect those values to generate a dataset. The question of time is also essential, the timestep is relative, and in reality, it is a continuous value. When collecting data from the sensors, we make approximations over time of collection and on values collected. We are mixing computer science with physics. Then we have to feed the models with those data, we can serve them with datasets, batches of data or streams of data. All of this requires rigorous development to limit approximations. Computer science is the bridge between statistical learning and physical reality.

Data Center Replica

We developed a module to collect the data of a replica of DC. Then we serve the data to a model in streaming based on a computation station and send back the decision to the replica. The material used was a National Instrument for communicating and controlling the replica. An interface was developed to ensure the data collection on time, some security had to be developed to avoid material deterioration.

1.2.5 Existing Framework and Environment

There are various frameworks existing to help develop and practising reinforcement learning, we can cite:

1. gym
2. stable-baseline3
3. garage

Environments are the equivalent of datasets for more common machine learning but are more complex to use. Implementing an environment must comply with the Reinforcement Learning

paradigm, and the environment must be formulated as Decision Markov Process. Some frameworks already gather environment in an easy-to-integrate framework like Gym. Some environments are built on physical engines. For example, MuJoCo is a free and open-source physics engine that aims to facilitate research and development in robotics, biomechanics, graphics and animation, and other areas where fast and accurate simulation is needed. The point is to have accurate simulations to perform dynamic optimisation, the primary purpose is to amend reinforcement learning in realistic environments. If reinforcement learning performs well in realistic simulations, it can perform well in the real world. Yet there are still a lot of differences between simulation and the real world. Using Reinforcement Learning, especially with Deep-Learning tracking the hyper-parameter and comparisons with other algorithms in various environments is challenging. There are frameworks like stable-baselines3 [70] which compare runs of Reinforcement Learning. Moreover, scores and benchmarks are available on their website and saved in their framework. This kind of framework is essential to improve this field. The implementation of the algorithms is standardised.

1.2.6 Contribution

For this thesis, we built a framework to realise the experiments. RL and, more specifically, DRL is a recent topic, and few packages are available. We need to use packages such as Tensorflow and some computer science skills for DRL to implement the interaction between environments and learning algorithms efficiently. The code must interface the physical environment described by temperatures and other physical values obtained through sensors, with agents encoded via statistical models. The major contribution to Orange is the code interfacing environments with RL and DRL algorithms in scenarios that emulate reality.

Chapitre 2

Bayesian Actor in Deep Reinforcement Learning

2.1 Introduction

Various industrial domains encounter dynamic optimisation problems. Many issues are hard to solve with physical methods. Some physical control applications may benefit from using artificial intelligence especially Deep Reinforcement Learning (DRL). Some recent research [16] has shown impressive results. They manage to use Reinforcement Learning (RL) for controlling tokamak plasma. It helps master nuclear fusion. Such work is promising regarding the current environmental challenges. RL may face convergence issues [90] [89] such as over-optimism bias or stability [91]. Traditional learning issues, such problems as overfitting [101] or catastrophic forgetting [5], [64] amplify the problem. Bayesian methods introduce new ways of computing the weights of neural networks. It tackles the overfitting problem [47] while modelling the uncertainty. It also has appreciable properties for sampling from the model. When experimenting with the current framework of DRL, we struggled with the lack of control over the actor exploration. The actor tends to saturate the action quickly and falls into sub-optimum solutions. The mainstream uses of noise for exploration consider using noisy action leading to the policy with close action. We expect to use more complex exploration schemes. Bayesian methods have already been considered, especially for the value function approximation [21].
Policy search is a hot topic in RL, and parameter exploration changes the perspective on RL exploration. We formalise the use of the Bayesian Neural Network as policy in DRL, emphasise the parameter space search of the framework and highlight its strength. Bayesian neural networks 'naturally' implement parameter-space search as the randomness comes from the policy parameters. The model learns noise parameters through iterations. We provide a new policy gradient theorem to benefit from the deterministic policy gradient theorem's efficiency while performing parameter exploration for Bayesian neural networks' properties. Actor-critic algorithms in DRL have gotten much attention and solved many tasks. They tackle both continuous and discrete domains. RL has taken advantage of deep learning through the policy gradient theorems (see [86], [80]). DRL has seen arisen efficient algorithms, such as Deep Deterministic Policy Gradient [55], Soft-Actor-Critic [36], and Policy Proximal

Optimization [76]. One recurrent issue with RL is exploring the environment efficiently. Parameter exploration was developed [74] to reduce variance during learning. Then it has been used to tackle unstructured exploration's "shaky" behaviour. While classic exploration is made through noises sampled at each action, leading to jerky motion patterns (see [61]), state-dependant noise generation avoids using noisy action generation to explore. The idea is to focus on structural generation (see [71]). One major contribution [68] introduces layer noise to generate policies from a central policy for deep RL algorithms. This approach is close to our work, but with a Bayesian neural network, the noise is contained in the parameter and shaped by learning. Another issue is the stability of RL algorithms. Distributional RL has got attention in research recently (see [84], [11], [41]), but their approaches focus on the critic. Our work considered that the actor also has a significant role in the stability to avoid divergence and introduce Bayesian Neural Networks as policies in DRL. It can be assimilated into the methods using ensemble methods [44] to encode policies. Some techniques already use ensemble methods, like Neuroevolution methods (see [48], [85], [53]). However, thanks to the Bayesian paradigm, we attempt to give a theoretical foundation to an ensemble approach by using probabilities. Bayesian learning methods are being deeply studied by the community. Another approach consists of modelling the parameters' behaviour while learning is an effective way of understanding learning [57] ; their work compares learning through stochastic gradient descent to a stochastic process. They use it as a variational EM algorithm and give tools for tuning the learning by choosing hyperparameters to avoid finding degenerate solutions. Our current objective is to use Bayesian neural to improve policy search and, more precisely, to encode policies during learning. Our work differs from the usual Bayesian RL methods [33] because it focuses on using neural networks. We expect to benefit from their recent improvements and understanding. We also provide the corresponding codebase. We chose to demonstrate the value of our approach using more common environments. During this thesis, we based our conceptualisation of RL on the book "Reinforcement Learning: An Introduction" [86]. This chapter briefly reminds us of the main concepts underlying RL and Bayesian conception.

2.2 Preliminaries

2.2.1 Dynamic Optimization and Markov Decision Processes

Dynamic optimisation formalises problems with a gain indexed on a discrete variable, usually associated with time. Many applications are built on Dynamic Optimisation (see [66], [13]). For example, we have computer communication networks, production operations, computer operating systems, and macroeconomic system behaviour. Dynamic optimisation consists in finding the optimal control function $u\colon X, \mathcal{N} \to X$ to optimise a gain function $g\colon X, U, \mathcal{N} \to \mathbb{R}$ depending on a state function x, which transition is given by $f\colon X, U, \mathcal{N} \to X$:

$$\begin{aligned} x(t+1) &= f(x(t), u(x(t), t), t) \\ J(t) &= g(x(t), u(x(t), t), t). \end{aligned}$$

The point is to find optimal control for the system. For our purposes, we use the discrete sequence $t = 0, 1, 2 \dots$ for the time steps, and we consider the following notation:

1. A set of states $\mathcal{S}$,
2. A set of actions $\mathcal{A}$,
3. A distribution over initial states $p(s_0)$,
4. A reward function $r: \mathcal{S} \times \mathcal{A} \to \mathbb{R}$,
5. A discount factor $\gamma \in [0, 1)$,
6. Transition probabilities $p(s_{t+1}|s_t, a_t)$,
7. A deterministic policy $\pi: \mathcal{S} \to \mathcal{A}$ or stochastic $\pi: \mathcal{S} \to \mathcal{P}(\mathcal{A})$,
8. We consider the trajectory $\tau = (s_0, a_0, \dots s_n, a_n \dots)$ with $s_0 \sim p(s_0)$, $s_{t+1} \sim p(s_{t+1}|s_t, a_t)$ and $a_t \sim \pi(s_t)$.

The state and action sets can be continuous or discrete. The policy is the main matter. The point is to encode decision methods that can be learned to optimize the reward trajectory. If the reward attribution is stochastic, we can equivalently define the expected reward:

$$r(s, a) = \mathbb{E}[R_t | S_{t-1} = s, A_{t-1} = a].$$

But in most cases, the reward is a deterministic function of the state-action pair.

2.2.2 Deep Learning

Deep learning is a hot topic in research, but it is also a broad subject. It comes with a bunch of models, algorithms and concepts [79]. We start by setting the bases. Multilayer perceptrons are hierarchical models. Given the composition notation $\bigcirc$ as

$$\bigcirc_{i \in [\![0\,;\,n]\!]} f_i(x) = f_0 \circ f_1 \circ ... \circ f_n(x), \tag{2.1}$$

we define a layer $\psi: \mathbb{R}^i \to \mathbb{R}^j$ with the weights parameters W and the bias parameters B respectively of size $[i \times j]$ and $[1 \times j]$ and an activation function φ:

$$\begin{aligned} \psi: \mathbb{R}^i &\to \mathbb{R}^j \\ x &\mapsto \varphi(Wx + B), \end{aligned} \tag{2.2}$$

where j is the number of neurons of the layer.
The neural network Ψ is given by:

$$\begin{aligned} \Psi: \mathbb{R}^i &\to \mathbb{R}^j \\ x &\mapsto \bigcirc_{l \in L} \psi^l(x), \end{aligned} \tag{2.3}$$

with l the number of layers and $\{\psi^l\}_{l \in L}$ the layer sequence. Neural networks appeal to function approximation or policy encoding due to their flexibility and gradient learning. They are adapted to handle various inputs, such as images or audio streams. They are trained with a gradient signal, which is convenient when learning is not structured as supervised or unsupervised learning. Deep Learning potential is no more to prove but comes with its difficulties. Models' complexity increases the risk of not handling the convergence of algorithms.

2.2.3 Bayesian Probability

Probability is about evaluating uncertainty about events. When considering uncertainty, the two main philosophies exist ; The frequentist interpretation is based on the law of large numbers, and many repetitions of events reveal the probabilities. In contrast, the Bayesian method is subjective and assumes knowledge about an event. Then the data observations update the beliefs. The Bayesian paradigm has two main ideas:

1. Considering parameters as variables.
2. Usage of prior and posterior knowledge.

With the first idea, the parameter space can be explored to find the most realistic model that best fits the data. The parameter space exploration is evaluated through the Data and its likelihood, leading to the second idea: We build our posterior belief from our prior knowledge and the likelihood. Those two concepts are formulated into the Bayes rule:

$$p(\theta|y) = \frac{p(y|\theta)p(\theta)}{p(y)}. \tag{2.4}$$

Where for the data y, and the parameter θ, $p(\theta|y)$ is the posterior, $p(y)$ is the prior, $p(y)$ the marginal and $p(y|\theta)$ the likelihood. Bayesian theory is widely discussed in literature [3] The trust associated with the Bayesian concepts is often discussed.

The Bayesian approaches introduce the prior and posterior and use Bayes' theorem when frequentist approaches are based on the likelihood. Frequentists assume hypotheses are either true or false, while Bayesian gives a probability to the hypothesis' truth. In the same idea, Bayesian concepts use random variables as parameters for their models, leading to beliefs about a model. Then, with experiments or Data, we update the beliefs to reduce uncertainty. This is where we get close to RL, and using Bayesian concepts to find a policy seems appealing. We start with a policy, which is updated with policy-environment interactions. Using Bayesian tools, such as Bayesian Neural Networks, captures the uncertainty around a policy and maintains a probability over its parameters. The uncertainty of the policy can be integrated and learned through RL. Another point is Deep Learning uses over-parameterised methods, making the models indiscernible. Bayesian methods diminish the issue by inferring the posterior shaped by the data and constrained by the prior.

2.2.4 Bayesian Deep Learning

Using the Bayesian paradigm to learn the neural network parameters is a way of dealing with issues such as over-fitting or uncertainty of the model. Moreover, the Bayesian paradigm brings some exciting methods, such as variational inference and the ability to sample the parameters, which can be advantageous in RL. To deal with the inference over the massive amount of parameters, using a variational approximation to exact Bayesian update is efficient (see [34], [43]).

Reparametrization Trick

The reparametrisation trick adds stochasticity to a DL model without breaking differentiation properties. It was first introduced by Kingma and Welling [49] to infer latent variables in an auto-encoder.

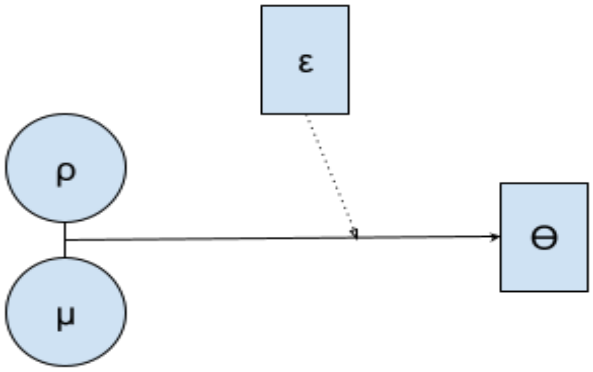


FIGURE 2.1 – The reparametrization trick used a random variable and two parameters to generate a random parameter. The relationship between the parameters μ, ρ and θ is deterministic.

The reparametrized parameters are estimated using a Kullback-Liebler Divergence and variational methods to approximate the true distribution maximising the usual loss. This deterministic transformation is used not to break the back-propagation. It is a way of making inferences efficiently in neural networks. The parameters are deterministic, and the stochasticity comes from supplementary random variables usually drawn from standard normal distributions to compute the gradient.

Bayes by Backprop

From these works, Charles Blundell [9] developed Bayes by Backprop. It regularises the weights by minimising a compression cost, known as the variational free energy or the expected lower bound on the marginal likelihood. This method obtains excellent performance in practice and is easy to implement. Moreover, as it uses gradient updates, it can readily be scaled using multi-machine optimisation schemes such as asynchronous SGD [15], and it is compatible with GPU computations, which is convenient for RL and its huge computation need.
Bayesian neural networks infer the parameters w by using the maximum a posteriori (MAP):

$$P(w|D) = \frac{P(D|w)P(w)}{P(D)}.$$

It comes that the MAP parameters w^* are:

$$w^* = \arg\max_w \log P(w|D) = \arg\max_w \left[\log P(D|w) + \log P(w)\right].$$

Bayes by backdrop [9] aims at minimizing the divergence of a variational posterior to the real posterior: $KL[q(w|\theta)||P(w|D)]$. They claim that for ϵ, a random variable having a probability density given by $q(\epsilon)$ and $w = t(\theta, \epsilon)$ where $t(\theta, \epsilon)$ is a deterministic function. With the marginal probability density of w, $q(w|\theta)$, is such that:

$$q(\epsilon)d\epsilon = q(w|\theta)dw. \tag{2.5}$$

Then for a function f with derivatives in w:

$$\frac{\partial \mathbb{E}_{q(w|\theta)}\left[f(w,\theta)\right]}{\partial \theta} = \mathbb{E}_{q(\epsilon)}\left[\frac{\partial f(w,\theta)}{\partial w}\frac{\partial w}{\partial \theta} + \frac{\partial f(w,\theta)}{\partial \theta}\right]. \tag{2.6}$$

For simplicity, we usually consider independent Gaussian distribution for each network parameter. Instead of the casual deterministic parameter ξ, we use the reparametrization trick and the parameters η_μ, η_ρ, and a random variable ϵ are used in a Bayesian context. From the learning perspective, the parameters are deterministic. When using the network, drawing ε from the distribution of ϵ gives the value used for computation. This method doubles the number of parameters but trains an infinite ensemble of networks. Uncertainty in the hidden units allows the expression of uncertainty about a particular observation. Uncertainty in the weights is complementary in that it captures uncertainty about which neural network is appropriate, leading to regularisation of the weights and model averaging. As in the original paper [9], we use the following deterministic transformation t:

$$w = t(\eta_\mu, \eta_\rho, \varepsilon) = \eta_\mu + \log\left(1 + \exp(\eta_\rho)\right)\varepsilon. \tag{2.7}$$

This lead to the following cost with $\theta = (\eta_\mu, \eta_\rho)$:

$$F(D,\theta) = KL\left[q(w|\theta)||P(w)\right] - \mathbb{E}_{q(w|\theta)}\left[\log P(D|w)\right]. \tag{2.8}$$

Which is the variational free energy [30], which is also called the expected lower bound [45]. It can be approximated by:

$$F(D,\theta) \approx \sum_{i=1}^{n}\left[\log q(w^{(i)}|\theta) - \log P(w^{(i)}) - \log P(D|w^{(i)})\right]. \tag{2.9}$$

With i representing Monte Carlo samples drawn from the variational, using the variance reduction technique known posterior as common random numbers [78]. The computation of the parameters is usually done through gradient descent. The prior can be seen as a form of regularisation.

2.3 Reinforcement Learning

By RL, we consider the machine learning framework built to perform dynamic optimisation. Usually, machine learning trains models from an already existing dataset. In contrast, RL is a paradigm that differs from supervised and unsupervised learning because of two main concepts: exploration and reward. RL is about mapping situations to actions to optimize a signal. From a possible action set, the learning agent must discover the interactions of actions on the environment to find an optimal policy. The learning is divided into two components exploration and parameters learning. The reward signal suggests how good the system's state is. RL optimises a dynamic system from which we lack knowledge but can interact with it. The learning agent explores an environment response and improves the policy through trials and evaluations. Like the other paradigm, it is a data-driven method, but data generation is part of the learning process. So the agent has to explore the environment enough to discover

the optimal policy but must also keep improving the current policy to converge toward the optimal policy. It is the exploration-exploitation dilemma. The main specificity of RL is learning by trial and not by example, and the main idea is that the data is generated during the learning. RL has to both optimise and explore. RL encodes its knowledge in a policy, which is the agent behaviour for a system state. It is a deterministic or stochastic mapping between states and actions. The policy is guided by a reward signal, a mapping between situations represented by system state and values, called reward. The policy goal is to maximize the whole signal. In RL, we use environment, agent and action, but using a controller, controlled system, and controlled signal is also common. It depends on the practitioners.

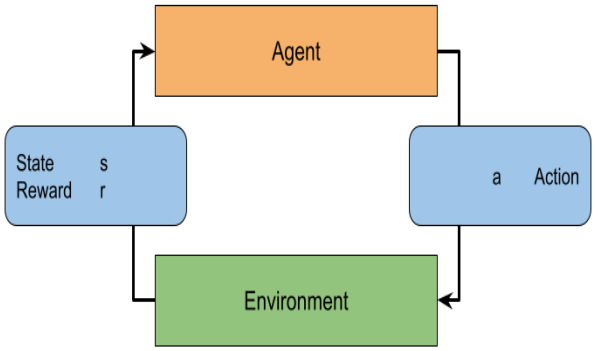


FIGURE 2.2 – The interaction of an agent and an environment in a RL context.

2.3.1 Goals and Returns

Reward Design

When considering RL, the reward is significant, and in many cases, the reward function is not determined by the environment. The signal associated with the reward has to be shaped to reach an adequate policy. The policy that maximises the sum of the timestep rewards must solve the task. The signal's sparsity is a real optimisation issue (see Section 17.4 [86]). In an episodic task, with a finite horizon for every episode, the return is the sum of the reward for every step of the episode: $G = \sum_{t=0}^{T} r(s_t, a_t)$. While for continuous tasks, the return is the discounted sum of the reward $G = \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)$. For the episodic task, industrial robots are automated for manufacturing welding, painting, assembly, and disassembly. We have ongoing control for continuous tasks, such as temperature regulation or self-driving cars.

In a dynamic optimisation context (as developed in Section 2.2.1), we use the value functions to estimate the efficiency of a policy on the environment:

$$V_{\pi_\theta}(s) = \mathbb{E}_\tau[G|s_0 = s, \pi_\theta], \tag{2.10}$$

and the state-action value function:

$$Q_{\pi_\theta}(s, a) = \mathbb{E}_\tau[G|s_0 = s, a_0 = a, \pi_\theta]. \tag{2.11}$$

Then for an agent, the goal is to get a policy that maximises the long-term objective, also called the total discounted reward or the return.

In RL, we use the return to achieve a goal. The return depends on the nature of the task, which can be either a finite or infinite horizon. Both representations are unified with the Return concept. Usually, the concepts of averaged return and discounted return are used in the literature. Sutton suggests using the averaged return in a continuous context [86], but in practice, the discounted return is convenient to use and spread in DRL literature.

Monte Carlo

Monte Carlo methods look for optimal policies by sampling whole trajectories. For each state-action pair, the complete averaged return is computed. For this, we need to sample trajectories from state-action pairs and estimate the average return for the current policy. Monte Carlo requires framing the RL problem to be episodes. In RL, Monte Carlo methods are opposed to methods that estimate the return from partial knowledge and anticipate the full return. The advantages lie in the estimation's seeming more trustworthy as no assumptions are made on the return. Still, it is more time-consuming and more sensitive to variations in the policy.

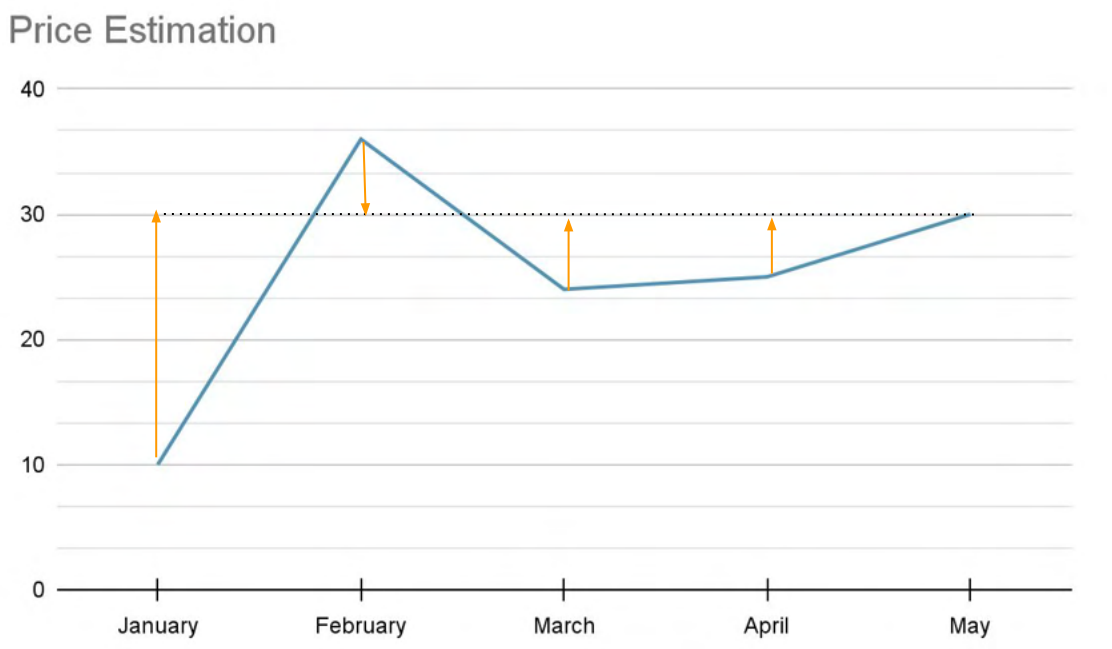


FIGURE 2.3 – Learning a policy to produce an estimate with Monte Carlo setting. The estimate of the price every month is updated giving the final price.

TD-learning

Temporal difference learning is a method of learning in RL. The idea is not to wait for the whole trajectory to be sampled but to estimate the return at every step. The more estimates we have, the more precise it is. It is a bootstrap method. Having a return estimation at every step is very convenient for improving the trajectory the agent is choosing. The core idea in TD-learning is using Bellman Equations to estimate the return recursively. The Bellman equations are a way of splitting a sum into a current part and a left-over. Assume the existence of $(a_t)_{t\geq t_0}$ taking values in the set $\mathcal{A}$ and $G\colon \mathcal{A} \to \mathbb{R}$ such as the following sum exist:

$$\begin{aligned} B(t_0) &= \sum_{t\geq t_0} \gamma^{t-t_0} G(a_t) = G(a_{t_0}) + \gamma \sum_{t>t_0} \gamma^{t-t_0-1} G(a_t) \\ B(t_0) &= G(a_{t_0}) + \gamma B(t_1). \end{aligned}$$

Then we observe that it is possible to consider a temporal solution in solving the current part of the Bellman equation and then recursively the future decision. Then the TD error is given by:

$$\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t).$$

Updating the agent at every step is an interesting property for agents learning online. They do not require the episodic formulation of the problem and can directly tackle continuing tasks.

N-steps

N-step methods compromise temporal difference and Monte Carlo for estimating the averaged return. Many applications do not require bootstrapping over one step. Using more information from the following steps can benefit learning. With n-step consideration, the impact of actions is observed over several steps. Using our previous example, we now have for n-steps:

$$B(t_0) = \sum_{t_0\leq t<t_0+n} \gamma^{t-t_0} G(a_t) + \gamma^{t_0+n} B(t_0+n).$$

2.3.2 Reinforcement Learning Methods

Tabular and Approximate Methods

RL theory first focused on a Markov Decision Process with finite action and state spaces. It considered low dimensional spaces and used tabular methods for finding the optimal policy. Many of these methods have the advantage of finding exact solutions [86].

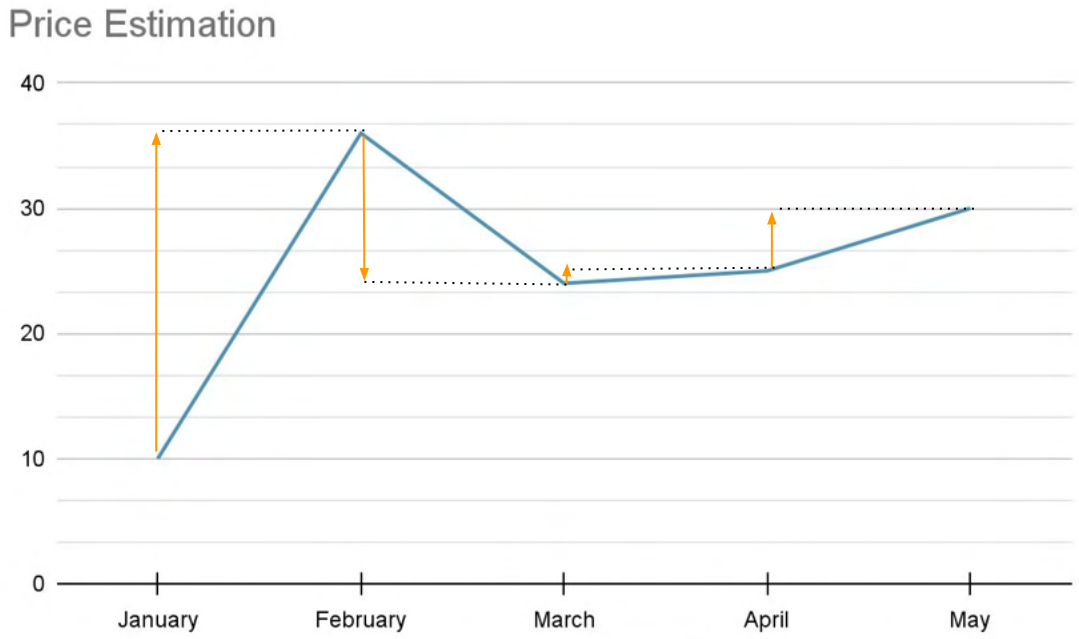


FIGURE 2.4 – Learning a policy to produce an estimate with TD-learning setting. The estimate of the price every month is updated giving the next month's price.

Then, methods maintaining an estimation of the return or the policy have been developed to tackle more complex situations with continuous spaces. We have the state visitation measure:

$$\rho^{\pi}(s) = \int_S \sum_{t=0}^{\infty} \gamma^t p_0(s) p(s \to s', t, \pi) ds, \tag{2.12}$$

leading to the following long-term objective:

$$\begin{aligned} J(\pi_\theta) &= \int_S \rho^{\pi}(s) \int_A \pi_\theta(a|s) r(s,a) da ds \\ &= \mathbb{E}_{s\sim\rho^{\pi}, a\sim\pi_\theta}[r(s,a)]. \end{aligned}$$

Many RL algorithms and methodologies have been built depending on the nature of the tasks. Environments may have either discrete or continuous state and action spaces. For complex tasks, we use approximation methods instead of tabular methods. in the thesis, we focus only on approximation methods, as we consider problems where the state and action spaces cannot be represented with tables and lists. The approximation methods are often divided into the following methods: Model-Based, Valued-Based, Policy-Based, and Actor-Critic.

Model Based Reinforcement Learning

In model-based methods, the environment's behaviour is captured in a mathematical model. The transition is modelled then a policy is derived. This kind of method is efficient for low dimensions and discrete environments.

Advantages	Drawbacks:
1. Easy to learn (supervised learning)	1. Learn unuseful behaviour
2. Learn from data	2. The policy is still to learn

Model-based methods are attractive when the environment mechanism is plain and easy to model, for example, if the environment is deterministic and the state space is discrete. A complex stochastic environment may need a considerable data generation process to fit a model. To use a model, we must be sure that the model is close enough to reality and that we are not producing counterfactual data without risk analysis [60]. The region where the model is valid depends on the model training and hyperparametrisation. Model-based methods are known to be unstable when the environment is too complicated due to uncertainty and approximation errors in the model.

Value Based Reinforcement Learning

Value-based algorithms learn how the state-action couple performs in the environment and derive an actor from the relationship. The straightest algorithms use greedy policy derived from the value function.

Advantages	Drawbacks:
1. Directly target the gain	1. We are not focused on the policy
2. The policy is easier to obtain	
3. A lot of progress has been made on the subject	

Value-based methods focus on modelling the gain associated with state-action pairs, avoiding modelling the complexity of the environment. This conception is better when the environment's transitions become more complex, but the policy is easy to define. When the value function is found, we construct the policy that optimises the value function. If the action space is complex, finding the policy might be tricky.

Policy Based Reinforcement Learning

Policy-based methods build a policy model and update it by estimating how it performs in the environment online. When iterating on the environment, we use the data generated to construct the policy gradient, which indicates how to improve the policy. This method uses the policy gradient theorem.

Advantages	Drawbacks:
1. Directly target the objective	1. Local optimum
2. Focus on the policy	2. The learned policy is specific to the seen interactions.
3. Easy to consider stochastic policy	
4. Good trade when the policy is much simpler than the environment	

Off-policy methods have also been developed to make learning easier.

Actor-Critic Reinforcement Learning

Actor-critic algorithms are a compromise between value-based and policy-based RL. It maintains two models, one for the policy and one for the critic, both learned simultaneously. The critic evaluates a state-action couple for the environment's transitions associated with a policy. The policy tries to find the actions that suit the best states following the critic indications. Actor-critic methods are often used with deep learning. The policy and critic are then encoded with neural networks. Some algorithms use a buffer to stack the transitions for the learning, then sampling at each step to train the critic and the policy.

On-Policy and Off-Policy Reinforcement Learning

We can split RL algorithms into on-policy and off-policy algorithms. The on-policy algorithms update the policy, which interacts with the environment, while the off-policy algorithms split the learning into exploration and learning. With off-policy methods, the parameter updates are asynchronous of the exploration. The policy for exploring the environment, usually called the behaviour policy, differs from the target learned policy. The updates are typically based on the history of interactions of the exploration policy. If the exploration policy is far from the target policy, the interaction may be irrelevant to learning. The advantages and drawbacks of using off-policy algorithms, from my perspective, are:

Advantages	Drawbacks:
1. Using parallel exploration is easier.	1. The learning uses samples that are not always relevant.
2. Using a data buffer.	2. The learned policy might differ greatly from the exploration policy and have unexpected behaviour.
3. The exploration benefits from separating the learned and exploration policies.	

Using off-policy learning is advantageous when data is available before the training. However, it is complicated to implement off-policy due to the counterfactual issue with a target policy which differs from the behaviour policy. Different approaches have been employed to tackle this issue, but much work might be necessary to learn policy from past experiences.

Using off-policy RL with function approximation is cutting-edge in research [86]. Methods like importance sampling or true gradient are necessary to deal with the challenge of the target update and the distribution of the update. These challenges arise since the interactions generated by an exploration policy may be unrelated to the target policies. The return must be estimated for the target policy, not the exploration one.

2.3.3 Policy Gradient

Policy Gradient theorems

Policy gradient methods are prevalent in continuous action RL. The main idea is to parameterise and update a policy toward the performance gradient. The main result under policy gradient methods is the policy gradient theorem giving the formulation for the gradient of the long-term objective. Policy gradient methods use the following assumption :

Assumption A1. *We assume that the policy is differentiable with respect to its parameters.*

Assumption A2. *S and A are compacts from $\mathbb{R}^n$ and $\mathbb{R}^m$.*

Assumption A3. *$p(s)$, $q(\epsilon)$, $p(s'|s,a)$, $\nabla_a p(s'|s,a)$, $r(s,a)$, $\nabla_a r(s,a)$, $\mu(s,w)|_{w=t(\theta,\epsilon)}$, $\nabla_\theta \mu(s,w)|_{w=t(\theta,\epsilon)}$ are continuous for all s,a,ϵ, θ.*

Assumption A4. *There are b and L from $\mathbb{R}$ such as $\sup_{s,a} r(s,a) < b$, $\sup_{s,a} \nabla_a r(s,a) < L$, $\sup_s p(s) < b$, $\sup_{s,a,s'} p(s'|s,a) < b$, $\sup_{s,a,s'} \nabla_a p(s'|s,a) < L$.*

We have the policy gradient theorem [87], that state that:

Theorem 1 (Policy Gradient Theorem [87])**.** *Using the same notation as in the original paper, let $Q^\pi : S \times A \to \mathbb{R}$ be the unbiased function approximation of the state-action function for the policy π:*

$$\begin{aligned} \nabla_\theta J_{PG}(\pi_\theta) &= \int_S \rho^\pi(s) \int_A \nabla_\theta \pi_\theta(a|s) Q^\pi(s,a) da ds \\ &= \mathbb{E}_{\substack{s\sim\rho^\pi \\ a\sim\pi_\theta}} [\nabla_\theta \log \pi_\theta(a|s) Q^\pi(s,a)]. \end{aligned}$$

This theorem states that we can learn a policy by evaluating the gain associated with state-action pairs and then using the gradients of this approximation to improve the policy. In practice, we observe that the gradients are independent of the data generation process, they do not appear on the state distribution ρ^π, so the gradient estimation can be done through the interaction process between the policy and the environment.

Theorem 2 (Deterministic Policy Gradient Theorem [80])**.**

$$\begin{aligned} \nabla_\theta J_{DPG}(\mu_\theta) &= \int_S \rho^\mu(s) \nabla_\theta \mu_\theta(s) \nabla_a Q^\mu(s,a)|_{a=\mu_\theta(s)} ds \\ &= \mathbb{E}_{s\sim\rho^\mu} [\nabla_\theta \nabla_a \mu_\theta(s) Q^\mu(s,a)|_{a=\mu_\theta(s)}]. \end{aligned}$$

The gradient expression informs that the derivative of the objective with regard to the policy parameters does not involve computation of the derivative of the state distribution, making it possible to sample trajectories to estimate the policy gradient. We use the policy gradient theorems to learn a policy for policy-based or actor-critic methods.

Deep Reinforcement Learning

DRL has seen emerged exciting algorithms, such as Deep Deterministic Policy Gradient [55], Soft-Actor-Critic [36], and Policy Proximal Optimization [76]. One recurrent issue with RL is exploring the environment efficiently. In applications, the exploration has some supplementary issues tackled with coherent exploration methods. Paradoxically RL is famous for its applications with deep learning models. At the same time, the theoretical guarantees are poor. Sutton even said [86]: "Although current RL theory is mostly limited to methods using tabular or linear function approximation methods, the impressive performances of notable RL applications owe much of their success to nonlinear function approximation by multi-layer ANNs". Policy gradient methods helped introduce neural networks as policy and critic encoders.

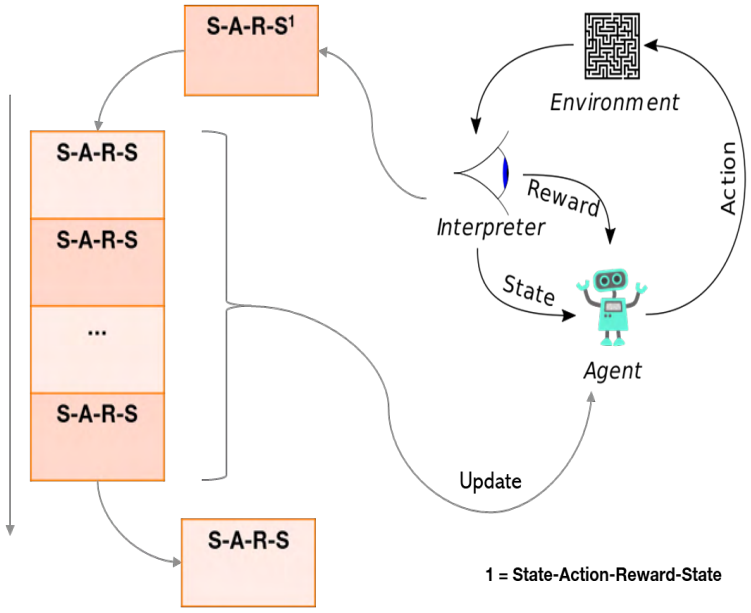


FIGURE 2.5 – The on-policy learning architecture of an actor-critic algorithm. The agent interacts with the environment and stacks the information generated in a data history. The tuple state-action-reward-state usually encodes the trajectory of the system. The critic maps state-action pairs to the Q-value based on a supervised estimation. The data history is a buffer with a capacity. When it is complete, the new data overwrites the oldest one.

In an actor-critic representation of RL, we use the following Bellman Operator:

$$\mathcal{T}Q(s_t, a_t) = r(s_t, a_t) + \gamma \underset{\substack{s_{t+1}\sim p,\\ a_{t+1}\sim\pi(s_{t+1})}}{E} \left[Q(s_{t+1}, a_{t+1})\right]. \tag{2.13}$$

The Bellman's Operator ensures the convergence of the critic by contraction and allows consideration of iterative algorithms to find the best policy. The RL agent comprises an actor encoding the policy and a critic encoding the value estimation. We are considering off-policy methods with a historic $\mathcal{H}$, filled with the interactions between the policy and the environment. We use the history to update the critic and the policy. The goal is to estimate the most suitable action for the environment at every step. The gradient methods are used to

update the policy and Q function parameters simultaneously. The Bellman operator estimates the Q function at every time step with its recursive formula. For continuous states and actions space, it is well known that both the policy and the Q-value estimator can be encoded with neural networks. The losses of both networks are minimised simultaneously. The usual objective loss is for the actor:

$$J_\pi(\theta) = \mathbb{E}_{s\sim\mathcal{H}}[-Q(s, \pi(s|\theta))],$$

and for the critic:

$$J_Q(\theta, \theta_{\text{target}}) = \mathbb{E}_{s_t,a_t,s_{t+1}\sim\mathcal{H}}[(Q(s_t, a_t|\theta) - (r_t + \gamma Q(s_{t+1}, \pi(s_{t+1})|\theta_{\text{target}})))^2].$$

The critic estimates the long-term reward for a given policy, and a target critic is used to increase the stability of the learning. The discount value determines the algorithm's horizon ; one must choose it to fit the environment. It is a way of characterising how far an action impacts the reward. Selecting a significant discount value might impact the convergence if an action impacts a few subsequent rewards. However, a small value will not let the actor converge to a long-term optimisation. The following expression can give the gradient estimator of the actor losses for a batch S drawn from the dataset and the number of samples N:

$$\hat{J}_\pi(\theta) = \frac{1}{N}\sum_{s\in\mathcal{H}}[-Q(s, \pi(s|\theta))].$$

Respectively for the critic, we have:

$$\hat{J}_Q(\theta) = \frac{1}{N}\sum_{s,a\in\mathcal{H}}[(Q(s_t, a_t|\theta) - (r_t + Q_{\text{target}}(s_{t+1}, \pi(s_{t+1}))))^2].$$

When using the RL algorithm, having an exploratory policy and a target policy is convenient. By separating the exploration and the learning, we can update the target policy with the data collected by the exploratory policy. This approach has considerable advantages [17]: " Off-policy methods have a wider range of applications and learning possibilities". Still, these methods are more challenging to apply [86]: "Off-policy bootstrapping combined with function approximation can lead to divergence and infinite MSE.".

2.3.4 Parameters and issues

RL and DRL have a lot of hyperparameters. Choosing them is challenging, and typical methods are complicated to implement due to the computation time. A run can last for hours or days. So testing different combinations can take weeks or months. It would be ideal to have criteria for the other parameters. Still, they are dependent on one another and deriving the objective is nearly impossible since we don't have the data distribution before exploring the environment. This is why it is crucial to understand how the hyperparameters impact the learning to choose coherent values.

learning-start is a parameter that delays the start to ensure the model explores and generates enough data to start learning properly. Starting very soon to learn might be learning onto too specific data leading the model to failure.

$\boldsymbol{\alpha}$ is a parameter specific to our model. It is a temperature parameter, adjusting the balance between the regularisation and the performance in the loss.

discount factor is the parameter balancing the consideration of the future with the current reward. Values too close to one another will lead to massive critic estimation and might produce divergence. Usually, this parameter is chosen around 0.9.

learning rate has a significant impact on learning [46]. With a significant learning rate, the model may not converge, oscillate over the optimal solution, and even saturate the action. In contrast, a too-small learning rate will be too slow to converge.

$\boldsymbol{\tau}$ is used with target networks to stabilise the learning critic, and the actor may use a target updated slower and toward the leading network. It is often associated with the "Polyak update". Using two critic networks [31] is also common to reduce the over-estimation of the critic.

Buffer length is the parameter controlling the size of the history. Large values will stack for a long time previous experiences, even if those states are no longer beneficial for policy improvement. While short values may "forget" too fast, the policy will not be able to tackle the forgotten states.

Batch length represents the interactions fed to the policy and gradient during the learning. A significant value will stabilize the learning but slow the learning and reduce the variance, which can be beneficial to not fall too fast in a local optimum.

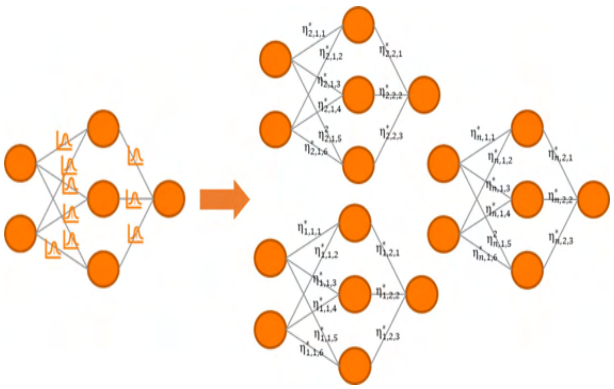


FIGURE 2.6 – Sampling policies encoded neural networks from a Bayesian neural network. We obtain standard neural networks by drawing the whole parameter set from the distributions.

2.4 Deep Bayesian Policy Search

In this section, we formalise using the Bayesian Neural Network for policy search. We emphasise the parameter space search of the framework and highlight its strength. We consider the Bayesian neural network to 'naturally' implement parameter space search as the noise

is applied to the parameters and is part of the definition of the model. The stochasticity of the model is learned through the learning iterations. We present our new policy gradient theorem to use the benefit of the deterministic policy gradient theorem efficiency with the parameter exploration properties of the Bayesian Neural Network. Distributional RL has gotten attention in research recently [84], [11], but their approaches focus on the critic. In our work, we considered that the actor also has a vital role in stability and avoiding divergence. Our work can be assimilated into the techniques using ensemble methods [44] to encode policies. Some techniques already take advantage of ensemble methods, like Neuroevolution methods [48], [85], [53]. However, thanks to the Bayesian paradigm, we attempt to give statistical substance and theoretical foundation to the approach. Bayesian neural networks are getting significant improvement and understanding from the community. Modelling the behaviour of the parameters while learning is an effective way of understanding learning [57] ; their work compares learning through stochastic gradient descent to a stochastic process. They use it as a variational EM algorithm and give tools for tuning the learning by choosing hyperparameters to avoid finding degenerate solutions.
The importance of parameter space exploration instead of action space has been highlighted in work [68]. For some environments, coherent exploration provided by parameters investigation is necessary. This work does not describe all the features associated with Bayesian Deep Learning, and we aim to create a bridge between the disciplines. Our current objective is not to improve Bayesian neural networks but to check how they can improve policy search and, more precisely, how they can encode policies during learning. Our work differs from the usual Bayesian RL methods [33] because it focuses on using neural networks. We expect to benefit from their recent evolution and development. The Bayesian neural network framework applies noise directly to the network parameters. The following works [104], [68], [77] have already studied the action space versus the parameter space exploration problem. We use parameter space exploration, which is inherent in our framework.

1. We use the recent improvements in Bayesian neural networks to infer parameters in neural networks (see [9]), set and learn the randomness of our policy.
2. We use the reparametrisation approach to sample policies and realize coherent exploration through the policy's parameter space with an algorithm.
3. We furnish a version of the policy gradient theorem to justify using Bayesian neural networks as actors in actor-critic methods.

Using the Bayesian neural network as a policy is new and has the same objective as parameter exploration (see [71]). Another approach is considering parameter space exploration and exploring the policies instead of the actions. One significant contribution introduces layer noise to generate policies from a central policy (see [68]). This chapter briefly reminds RL, policy gradient methods, and the Bayesian Neural Networks trained with Bayes by Backprop [9]. Then introduce our agent and its algorithm and give results on typical benchmarks.

2.4.1 Bayesian Deep Policy

Our framework is built over the policies gradient theorem in an actor-critic configuration. We use Bayesian neural networks for policy encoding. In the original Bayes by backprop paper [9], an example of RL is given on Contextual Bandits. We explored the idea of using

Bayesian neural networks for RL. In the example, the Bayesian neural network is used in a Q-learning method with Thompson sampling. Our work uses a Bayesian neural network as an ensemble policy encoding. The action-value function may use Montecarlo's estimation. Our policy is built on a parameter set θ and the random variable ϵ using our reparametrisation t, and we obtain $w = t(\theta, \epsilon)$. When using our policy, we denote $a = \pi_\theta(s, w)$, meaning for the encoded policy with θ as parameters and w as the generated network. We start by defining the Bayesian Deep Policy:

Definition 1 (Bayesian Deep Policy)**.** *A Bayesian Deep Policy π_θ depends on the following:*

1. *A set of parameters θ from Ω.*
2. *A set of stochastic random variable ϵ from $\mathcal{E}$.*
3. *A reparametrisation function $t: \Omega \times \mathcal{E} \to W$.*
4. *A prior probability p.*

The parameters $\theta = (\eta_\mu, \eta_\rho)$ gives the posterior probability. If we use the reparametrisation given in Equation (2.7), we have: $w|\theta \sim \mathcal{N}(\eta_\mu, \log(1 + \exp(\eta_\rho)))$. For the Bayesian Deep Policy, sampling the random variable ϵ is equivalent to sampling standard neural network μ_w whose parameters are $w = t(\theta, \epsilon)$. Bayesian Neural Network can also be seen as a function from $S \times \mathcal{E}$ to the action space A. $a = \mu(s, \epsilon)$. The probability of getting an action a under the Bayesian policy for the state s is given by:

$$p(a|s, \theta) = \int_\Omega q(w|\theta) p(a = \mu_w(s)) dw = \int_\mathcal{E} q(\epsilon) \mathbb{1}_{\{a=\mu(s,\epsilon)\}} d\epsilon. \tag{2.14}$$

As the generated standard neural networks are entirely deterministic, $p(a = \mu(s, \epsilon))$ is always one or zero. For the value function, we have the following from the definitions ((2.10), (2.11)):

$$V^\pi(s) \;=\; \int_A p(a|s, \theta) Q^\pi(s, a) da.$$

From Equation (2.14), we get rid of action space integral:

$$\begin{aligned} V^\pi(s) &= \int_\Omega q(w|\theta) Q^\pi(s, a)|_{a=\mu_w(s)} dw \\ &= \mathbb{E}_{q(w|\theta)} \left[Q^\pi(s, a)|_{a=\mu_w(s)} \right] = \mathbb{E}_{q(\epsilon)} \left[Q^\pi(s, a)|_{a=\mu_\theta(s,\epsilon)} \right]. \end{aligned} \tag{2.15}$$

We notice that the integral over the action space is not valuable anymore and is replaced by the integral over the parameter space. We denote it as a parameter space search instead of an action space search. We can express the associated Bellman operator for the policy π_θ as follows:

$$\begin{aligned} \mathcal{T} Q^\pi(s, a) &= r(s, a) + \gamma \mathbb{E}_{\substack{p(s'|s,a) \\ q(w|\theta)}} \left[Q^\pi(s', \mu_w(s')) \right] \\ &= r(s, a) + \int_S \gamma p(s'|s, a) \int_\Omega q(w|\theta) Q^\pi(s', a)|_{a=\mu_w(s_{t_1})} dw ds'. \end{aligned} \tag{2.16}$$

Then our objective function becomes:

$$\begin{aligned} J(\pi_\theta) &= \int_S p_0(s) V^\pi(s) ds \\ &= \int_S p_0(s) \mathbb{E}_{q(\epsilon)} \left[Q^\pi(s,a)|_{a=\mu_\theta(s,\epsilon)} \right] ds \\ &= \mathbb{E}_{q(\epsilon)} \left[\int_S p_0(s) Q^\pi(s,a)|_{a=\mu_\theta(s,\epsilon)} ds \right]. \end{aligned} \tag{2.17}$$

We formulate the following distribution, which will be used as the likelihood in classical MAP approaches:

$$P(\pi_\theta = \pi^* | \rho^\pi) = \frac{exp(\int_S p_0(s) Q^\pi(s,a)|_{a=\mu_\theta(s,\epsilon)} ds)}{Z}. \tag{2.18}$$

The normalisation constant does not depend on the action and won't participate in the gradient direction. This consideration has already been seen in the Soft Actor-Critic [36]. This distribution is closely linked to policy performance from the critic's perspective. Now we have formulated the RL objective Equation (2.17), we have to plug it into the Bayesian loss given by Equation (2.8), and we use it as an objective, conducting us to the following policy objective:

$$J^*(\pi_\theta) = \int_S p_0(s) \mathbb{E}_{q(\epsilon)} \left[Q^\pi(s,a)|_{a=\mu_\theta(s,\epsilon)} \right] ds - \alpha KL \left[q(w|\theta) || p(w|\theta_{\text{prior}}) \right]. \tag{2.19}$$

We add a parameter α to balance the complexity cost relative to the likelihood cost. Now we have formulated the objective function. We have to ensure the gradients still have a convenient form. We formulate our version of the policy gradient theorem for Bayesian neural network policies. We follow the same ideas and assumptions as the previous versions.

2.4.2 Bayesian Deterministic Policy Gradient

Theorem 3 (Bayesian Deterministic Policy Gradient)**.** *Our policy gradient theorem, based on the usual assumptions (A1, A2, A3, A4), uses the reparametrisation trick from the Bayesian neural networks. This is from the proposition from Charles Blundell (see [9]), stating: Let ϵ be a random variable having a probability density given by $q(\epsilon)$ and let $w = t(\theta, \epsilon)$ where $t(\theta, \epsilon)$ is a deterministic function. Suppose further that the marginal probability density of w, $q(w|\theta)$, is such that:*

$$q(\epsilon) d\epsilon = q(w|\theta) dw. \tag{2.20}$$

Then for a function f with derivatives in w:

$$\frac{\partial \mathbb{E}_{q(w|\theta)} \left[f(w,\theta) \right]}{\partial \theta} = \mathbb{E}_{q(\epsilon)} \left[\frac{\partial f(w,\theta)}{\partial w} \frac{\partial w}{\partial \theta} + \frac{\partial f(w,\theta)}{\partial \theta} \right]. \tag{2.21}$$

Then we have the gradients:

$$\begin{aligned} \nabla_\theta J^*(\pi_\theta) &= \mathbb{E}_{q(w|\theta)} \Big[\int_S \rho^\pi(s) \nabla_\theta \mu_\theta(s,\epsilon) \nabla_a Q^\theta(s,a)|_{a=\mu_\theta(s,\epsilon)} ds \\ &\quad + \alpha \left(\nabla_\theta \log P(w) - \nabla_\theta \log q(w|\theta) \right) \Big] \\ &= \mathbb{E}_{q(\epsilon), \rho^\pi} \Big[\nabla_\theta \mu_\theta(s,\epsilon) \nabla_a Q^\theta(s,a)|_{a=\mu_\theta(s,\epsilon)} ds \\ &\quad + \alpha \Big(\nabla_\theta \log P(w)|_{w=t(\theta,\epsilon)} - \nabla_\theta \log q(w|\theta)|_{w=t(\theta,\epsilon)} \Big) \Big]. \end{aligned}$$

The Theorem 3 introduce the expectation under the state visitation measure for the gradients, meaning that we can sample from the visited states to estimate the gradients. It also uses the fact that the sampled policies are deterministic. It shares both the ideas of the policy gradient theorem and the deterministic gradient theorem. The main idea is when using a reparametrisation such as $q(\epsilon)d\epsilon = q(w|\theta)dw$, we can update the model by sampling standard neural networks and their interactions with the environment. It is very convenient as we can choose the frequency of neural network sampling and sample one every episode if necessary. Concerning the prior choice, the main principle from the robust Bayesian analysis is not to choose prior, leading to unreasonable variations. Two factors play a significant role when choosing the hyperparameters:

1. The parameter α balances the gradient between the compression cost and the likelihood. To choose this value, it is necessary to consider the size of the reward.
2. The initialisation of the prior and the posterior depends on the environment. It will shape the exploration. The example of the environment LunarLanderContinuous shows the importance of a good initialisation.

Those parameters are keys for dealing with the dilemma of exploitation versus exploration. Choosing them is complex and can be studied from a robust Bayesian perspective. The main principle is to choose them to have reasonable variation. However, while the action variation with respect to the parameter depends on the network architecture, the variation in reward with the action is entirely dependent on the environment. The main issue with learning methods is that the gradient is built on the function approximation. When using non-linear function approximations such as neural networks, we don't have the theoretical guarantee of the equality of gradients. The question of the prior is fundamental in Bayesian Learning. Through our work, we try to explore a way of integrating it into RL. We use it as a regularisation of the parameters.

Algorithm 1 : Bayesian Deep Policy Search

```
procedure LEARNING(θ, θ_Q)
    Env Initialization
    θ Initialization
    θ_prior ← θ
    θ_Q Initialization
    D = {}
    for number of episodes do
        for number of samples do
            Draw ϵ
            s = Env.reset()
            for number of iterations do
                a ← μ_θ(s, ϵ)
                r, s' ← Env.step(a)
                append s, a, s', r to D
                s ← s'
        for number of updates do
            for number of actors do
                Draw w ∼ θ
                Estimate gradients of J(π_θ)
                Estimate gradients of J_Q
            Update θ
            Update θ_Q
```

The algorithm 1 presents the procedure to train an agent using coherent noise application. The noise comes from the drawing of ϵ before each rollout.

2.4.3 Implementation

We used Python, TensorFlow and Keras to develop the Deep Bayesian Policy agent. Tensorflow and Keras allow us to change architecture and loss function in detail by over-classing the existing classes. We have created a Bayesian layer with a random component. Drawing neural networks is a functionality that is available for the user. The point is to let the user choose when sampling the parameters, which is very useful for keeping coherent noise during a rollout.

2.5 Results and Discussion

2.5.1 Environment

To assess the performance of our agent, we choose to evaluate its performance and compare it with usual environments. The point is to demonstrate its ability to solve complex tasks with state-of-the-art performance. We also show that it has specific exploration advantages, as it can solve problems that usually need supplementary strategies to be solved. We chose the following environment to perform our evaluation:

MountainCarContinous is a sparse deterministic environment in which the agent tries to make a car reach the top of a hill. In this environment, the agent action is a value proportional to the directional force applied to the car. The reward penalizes a value proportional to the squared of the action at each step and gives +100 if the car succeeds in reaching the top of the hill. While the agent failed this task, the reward will suggest more minor actions, leading to a poor policy. To succeed, the agent must explore the environment before reaching the local optimum. This environment shows the ability of an agent to solve a simple problem with poor information.

HalfCheetah is based on the work by P. Wawrzynski in "A Cat-Like Robot Real-Time Learning to Run". The HalfCheetah is a 2-dimensional robot with nine links and eight joints connecting them (including two paws). The goal is to apply torque on the joints to make the cheetah run forward (right) as fast as possible, with a positive reward allocated based on the distance moved forward and a negative reward allocated for moving backwards. The torso and head of the cheetah are fixed, and the torque can only be applied on the other six joints over the front and back thighs (connecting to the torso), shins (connecting to the thighs) and feet (connecting to the shins). Observations consist of positional values of different body parts of the cheetah, followed by the velocities of those individual parts (their derivatives), with all the positions ordered before all the velocities. The reward consists of two parts: A reward of moving forward, which is positive if the cheetah runs forward (right). A cost for penalising the cheetah if it takes too large actions.

LunarLanderContinuous is a starship trajectory optimization problem. The goal is to land the starship on a pad without crashing. The landing pad is always at coordinates (0,0). The coordinates are the first two numbers in the state vector. The action space has two dimensions, the first determines the throttle of the main engine, while the second specifies the throttle of the lateral boosters. The main engine will be turned

off entirely if the primary is negative and the throttle scales from 50% to 100% for 0 <= main <= 1. Similarly, if -0.5 < lateral < 0.5, the lateral boosters will not fire. If lateral < -0.5, the left booster will fire ; if lateral > 0.5, the right booster will fire. Again, the throttle scales affinely from 50% to 100% between -1 and -0.5 (and 0.5 and 1, respectively). State space has eight dimensions: the coordinates of the lander in x and y, its linear velocities in x and y, its angle, its angular velocity, and two booleans representing whether each leg is in contact with the ground. The reward for landing at the landing pad and coming to rest is about 100-140 points. The starship loses its reward if it moves away from the landing pad. If the lander crashes, it loses 100 points. If it comes to rest, it receives an additional +100 points. Each leg with ground contact is +10 points. Firing the main engine is -0.3 points in each frame. Firing the side engine is -0.03 points in each frame. We solve the problem if we obtain at least 200 points.

2.5.2 Compared Models

TRPO : Trust Region Policy Optimisation is a policy gradient algorithm authored by Schulman and al. [75]. They have constructed their policy update on the following constraint with C, a value discussed in the original paper:

$$J(\pi_{\theta+\Delta\theta}) - J(\pi_\theta) \geq \mathcal{L}_{\pi_{\theta+\Delta\theta}} - CD_{\mathrm{KL}}(\pi_\theta|\pi_{\theta+\Delta\theta}).$$

The novelties of this algorithm can be summarized as follows:

— The conjugate gradient is used to avoid computing the Hessian, and the Fisher information matrix can be used without inversion

— The constraint on the update $\mathrm{D}_{\mathrm{KL}}(\theta|\theta_k) \leq \delta$ is enforced by reducing the step size until the condition is fulfilled

— When updating the actor to a new policy, the improvement is checked: if $\mathcal{L}(\theta) < 0$, then the improvement is rejected

PPO : Proximal Policy Optimisation [76] has various implementations, and we choose to focus on the clipped version. The algorithm keeps the idea of constraining the learning like the TRPO but simplifies the loss computations by clipping the loss on every update if needed:

$$\mathcal{L}^{\mathrm{CLIP}}_{\theta_k}(\theta) = \mathbb{E}_{\tau\sim\pi_k}\left[\sum_{t=0}^{T}\left[\min(r_t(\theta)\hat{A}^{\pi_k}_t, \mathrm{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}^{\pi_k}_t)\right]\right] 0$$

DDPG/TD3 : Deep Deterministic Policy Gradient [55] is an actor-critic algorithm which concurrently learns a Q-function and a policy. It uses off-policy data stored in a buffer and the Bellman equation to learn the state-action function and the approximated state-action function to learn the policy. For stability, supplementary neural networks are used to update the other network. The target actor is at the loss of the critic, and the target critic is used for the actor update. The target network is updated using the "polyak" update toward its actual network. The actor is entirely deterministic, so noise is added to the output actions to improve the exploration of the environment.

Using two critics and choosing the minimum value between them for actor evaluation, updating the actor with a lower frequency than the critic and adding noise to the action to perform the critic update lead to the Twin Delayed Deep Deterministic Policy Gradients (TD3) [31].

ARS : Augmented Random Search [58] is an algorithm proposed by Horia Mania and al. that focuses on applying noise to policy parameters. For a constant υ and a random vector of numbers drawn from a normal distribution δ, using n policies and rollouts, a real stepsize $\alpha > 0$, the gain $G(\pi_{k,j}, +)$ for the policy π_k increased with the j^e noise vector drawn and $G(\pi_{k,j}, -)$ if the noise vector j applied was multiplied by -1, $\delta_{(j)}$ the maximum of the two gain for j and their standard deviation σ_G leading to the following policy update:

$$\theta_{k+1} = \theta_k + \frac{\alpha}{n\sigma_G} \sum_{j=1}^{n} \left[G(\pi_{k,j}, +) - G(\pi_{k,j}, -) \right] \delta_{(j)}.$$

Moreover, they proposed a version with a normalisation of the policy noise based on the mean and covariance of the states encountered.

A2C : A2C [59] focus on using various actors asynchronously to explore more efficiently. They based their work on using asynchronous gradient descent for deep neural network optimisation. They found that ensembling the actors asynchronously is empirically robust in learning.

SAC : Soft Actor-Critic is a model-free DRL algorithm developed by Tuomas Haarnoja and al. [36]. The algorithm is an off-policy actor-critic DRL algorithm having an actor that maximises the expected reward and the entropy. For this, they consider the following objective for the actor, with $\mathcal{H}$ the entropy:

$$J(\pi) = \sum_{t=0}^{T} \mathbb{E}_{(s_t, a_t) \sim \rho_\pi} \left[r(s_t, a_t) + \alpha \mathcal{H}(\pi(\cdot|s_t)) \right].$$

They update the critic using the following Bellman operator:

$$\mathcal{T}Q(s_t, a_t) = r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim p} \left[V(S_{t+1}) \right].$$

with

$$V(S_{t+1}) = \mathbb{E}_{a_t \sim \pi}[Q(s_t, a_t) - \log \pi(a_t|s_t)].$$

Then they update the policy according to the following:

$$\pi_{\text{new}} = \arg\min_{\pi} D_{\text{KL}} \left(\pi(\cdot|s_t) | \frac{\exp(Q^{\pi_{\text{old}}}(s_t, \cdot))}{Z^{\pi_{\text{old}}}(s_t)} \right).$$

TQC : The Truncated Quantile Critics [50] tackles overestimation bias as one of the most significant impediments to accurate off-policy learning. This algorithm blends three ideas: distributional representation of a critic, truncation of critics' prediction, and ensembling of multiple critics. They use distributional representation and truncation for arbitrary granular overestimation control. TQC values sample efficient off-policy RL with an accurate approximation of the Q-function.

2.5.3 Results

Table 2.1 – Estimated rewards from two hundred runs for different state-of-art agents and our agent on the environments of HalfCheetah-v3, LLC-v2 and MCC-v0

Model	HalfCheetah-v3	LLC-v2	MCC-v0
TQC	**12102.81 +/- 119.24**	**279.66 +/- 13.79**	76.40 +/- 38.49
TRPO	1785.07 +/- 55.05	273.95 +/- 23.86	92.59 +/- 0.08
PPO	5836.27 +/- 171.68	274.47 +/- 24.37	94.57 +/- 0.45
DDPG	/	223.87 +/- 80.41	93.51 +/- 0.05
TD3	9709.01 +/- 104.84	/	93.46 +/- 0.05
ARS	4046.14 +/- 2253.39	/	**96.50 +/- 0.78**
A2C	3096.61 +/- 82.49	83.21 +/- 135.70	91.17 +/- 0.26
SAC	9564.23 +/- 79.21	251.89 +/- 71.75	94.53 +/- 1.26
DBP	10611.30 +/- 137.90	231.30 +/- 60.79	91.90 +/- 1.62

The results of our experiments are summarized in Table 2.1. This table shows the rewards estimated on HalfCheetah-v3, LLC-v2 and MCC-v0 environments. Our agent was tested on two hundred runs while other agents' results were obtained from stable-baseline-3 [70], all taken at 1M steps. Using public results ensures reproducibility and results validated by the community avoiding error introduction and counter-performance from the state-of-art algorithms. It is also a way of avoiding unnecessary computations. Indeed, the stability and reproducibility of DRL [42] is a serious matter. Still, the need for power computation to tune, evaluate the stability and experiment deeply with an agent lets us only show how promising Bayesian Neural Network Actors are for DRL. To evaluate our model, we take the centre of the actor distribution. For this, we fix all the η_ρ parameters to 0. Our model is evaluated on a single training, and then we run 200 episodes with the trained agent to demonstrate the ability of our agent to solve a task. The results on the mountainCar-continuous-v0 show how the model can make coherent exploration. Indeed mountainCar-continuous-v0 is hard to solve without reward modification or specific strategy [12]. Models from stable-baseline-3 use state-dependent exploration [74] and generalized state-dependent exploration [71] noise to solve this environment. Our model includes coherent exploration in the model conception and only needs a particular prior specification to solve it. Indeed the prior needs to be "wide" enough to include a policy that can be successful. The LunarLander-continuous environment has particular action and state spaces. The actions activate the boosters only over a threshold, making exploring such an environment difficult. Our agent seems to be more flexible when it comes to environment exploration.

2.5.4 Limitation and perspective

Deep Bayesian policy is straightforward to implement, as it means replacing standard policies with a deep neural network, making little modifications. Then parallelisation is also easy to implement as it is part of the framework as a Bayesian neural network ensemble model.

The difficulty with Bayesian methods is to choose a prior correctly. For the Bayesian neural networks, the prior plays primarily the role of penalisation. The second point is to select the model initialisation. In our case, we initialise the model to the prior. In our experiments, we observe using α equal to zero still gives good results and may converge faster in some cases. However, using a larger value of α is a way of containing the parameters and slowing the convergence to control exploration. the original paper highlighted that the likelihood cost and the compression cost should be balanced when using Bayesian inference over the parameters of a neural network. Perhaps using decreasing series as our α may be interesting. In practice, we observe good results even with an α equal to zero. With no contribution from the prior, the agent can still converge but tends to get a more significant value for the parameters. The prior can be studied deeply to improve the exploration strategy. For example, considering a mix of models of prior and posterior. Other RL approaches, such as entropy objectives, may benefit the algorithms. New approaches in the Bayesian learning field may also improve such methods. For example, using normalising flow [56] would improve understanding and usage of the probabilities inside the network. Other losses and Bayesian inference approaches are promising to train neural networks (see [57]). One major perspective for RL is to develop "curiosity" (see [86] Section 17.6). Probabilities may have a card to play by using a cost based on entropy or considering different kinds of processes for the parameters. Bayesian Neural Network is a hot topic in Deep Learning. We can expect that the improvement of the field can be used in DRL. Through these experiments, we didn't add action noise to the action generation to concentrate on the policy noise generation. But both may be used simultaneously. We could consider using a Bayesian neural network for the critic to improve the performance as considered in [41]. Still, the point of this chapter was to show the improvement of using a Bayesian neural network as policy encoding.

2.6 Conclusion

This chapter introduces how to implement a Bayesian neural network in a DRL concept with a theoretical foundation and an algorithm. The framework encapsulates the policy exploration concept and furnishes the possibility of using coherent exploration schemes. Sampling the actor allows tuning the exploration by leveraging the number of networks sampled. We obtain reproducible results on our physical control application by improving the procedure's stability. We compared the Bayesian neural network agent and state-of-the-art algorithms in the Lunar-Lander, mountain-car-continuous and Halfcheetah environments. Our agent benefits from a robust exploration due to its definition, making it unnecessary to use a supplementary exploration strategy for an environment like the mountain car. We first presented the agent and its behaviour with a plain environment to furnish intuition on how α and the initialisation impact the learning. Then we realised a comparison showing the benefit of our approach and highlighting the benefit of using deep Bayesian methods for encoding the policy in DRL.

2.7 Appendix

2.7.1 Proof of Theorem 3

We use the assumptions A1, A2, A3 and A4 to use Fubini's theorem and Leibniz's integration rule along with the following proof.

Proof.
Starting with the Equation of the value function (2.15) and the Bellman Equation (2.16):

$$\begin{aligned}\nabla_\theta V^\theta(s_{t_0}) &= \nabla_\theta \mathbb{E}_{q(w|\theta)}\left[Q^\theta(s_{t_0}, \mu_w(s_{t_0}))\right] \\ &= \nabla_\theta \int_\Omega q(w|\theta)\left[r(s_{t_0}, a)|_{a=\mu_w(s_{t_0})}\right. \\ &+ \left.\int_S \gamma p(s_{t_1}|s_{t_0}, \mu_w(s_{t_0})) \int_\Omega q(w'|\theta) Q^\theta(s_{t_1}, a)|_{a=\mu_{w'}(s_{t_1})} dw' ds_{t_1}\right] dw.\end{aligned}$$

Then we use the reparametrisation and the result given in Equation (2.21):

$$\begin{aligned}\nabla_\theta V^\theta(s_{t_0}) &= \nabla_\theta \int_\mathcal{E} q(\epsilon)\left[r(s_{t_0}, a)|_{a=\mu_\theta(s_{t_0},\epsilon)}\right. \\ &+ \left.\int_S \gamma p(s_{t_1}|s_{t_0}, \mu_\theta(s_{t_0}, \epsilon)) \int_\mathcal{E} p(\epsilon') Q^\theta(s_{t_1}, a)|_{a=\mu_\theta(s_{t_1},\epsilon')} d\epsilon' ds_{t_1}\right] d\epsilon.\end{aligned}$$

Now that the integral does not depend on θ anymore, we use the Leibniz integral rule:

$$\begin{aligned}\nabla_\theta V^\theta(s_{t_0}) &= \int_\mathcal{E} q(\epsilon)\left[\nabla_\theta \mu_\theta(s_{t_0}, \epsilon)\nabla_a r(s_{t_0}, a)|_{a=\mu_\theta(s_{t_0},\epsilon)}\right. \\ &+ \int_S \gamma \nabla_\theta \mu_\theta(s_{t_0}, \epsilon)\nabla_a p(s_{t_1}|s_{t_0}, a)|_{a=\mu_\theta(s_{t_0},\epsilon)} \int_\mathcal{E} p(\epsilon') Q^\theta(s_{t_1}, a)|_{a=\mu_\theta(s_{t_1},\epsilon')} d\epsilon' ds_{t_1} \\ &+ \left.\int_S \gamma p(s_{t_1}|s_{t_0}, a)|_{a=\mu_\theta(s_{t_0},\epsilon)} \nabla_\theta \int_\mathcal{E} p(\epsilon') Q^\theta(s_{t_1}, a)|_{a=\mu_\theta(s_{t_1},\epsilon')} d\epsilon' ds_{t_1}\right] d\epsilon.\end{aligned}$$

Now we notice in the first part of the state-action value function, and we start to gather what's left to retrieve the state-action visitation measure under the policy:

$$\begin{aligned}\nabla_\theta V^\theta(s_{t_0}) &= \int_\mathcal{E} q(\epsilon)\left[\nabla_\theta \mu_\theta(s_{t_0}, \epsilon)\nabla_a \left(r(s_{t_0}, a)|_{a=\mu_\theta(s_{t_0},\epsilon)} + \int_S \gamma p(s_{t_1}|s_{t_0}, a)\mathbb{E}_{q(\epsilon)}[Q^\theta(s_{t_1}, a)|_{a=\mu_\theta(s_{t_1},\epsilon)}]\right)\right. \\ &+ \left.\int_S \gamma p(s_{t_1}|s_{t_0}, a)|_{a=\mu_\theta(s_{t_0},\epsilon)} \nabla_\theta \int_\mathcal{E} p(\epsilon') Q^\theta(s_{t_1}, a)|_{a=\mu_\theta(s_{t_1},\epsilon')} d\epsilon' ds_{t_1}\right] d\epsilon \\ &= \mathbb{E}_{q(\epsilon)}\left[\nabla_\theta \mu_\theta(s_{t_0}, \epsilon)\nabla_a Q^\theta(s_{t_0}, a)|_{a=\mu_\theta(s_{t_0},\epsilon)}\right] \\ &+ \int_\mathcal{E} q(\epsilon)\left[\int_S \gamma p(s_{t_1}|s_{t_0}, a)|_{a=\mu_\theta(s_{t_0},\epsilon)} \nabla_\theta \int_\mathcal{E} p(\epsilon') Q^\theta(s_{t_1}, a)|_{a=\mu_\theta(s_{t_1},\epsilon')} d\epsilon' ds_{t_1}\right] d\epsilon \\ &= \mathbb{E}_{q(\epsilon)}\left[\nabla_\theta \mu_\theta(s_{t_0}, \epsilon)\nabla_a Q^\theta(s_{t_0}, a)|_{a=\mu_\theta(s_{t_0},\epsilon)}\right] \\ &+ \int_\mathcal{S} p(s_{t_0\to t_1}|\pi)\gamma \nabla_\theta \int_\mathcal{E} q(\epsilon) Q^\theta(s_{t_1}, a)|_{a=\mu_\theta(s_{t_1},\epsilon)} d\epsilon d\tau.\end{aligned}$$

Then we have from the value function definition:

$$\begin{aligned}
\nabla_\theta V^\theta(s_{t_0}) &= \mathbb{E}_{q(\epsilon)}\left[\nabla_\theta \mu_\theta(s_{t_0},\epsilon)\nabla_a Q^\theta(s_{t_0},a)|_{a=\mu_\theta(s_{t_0},\epsilon)}\right] \\
&+ \int_{\mathcal{S}} p(s_{t_0\to t_1}|\pi)\gamma\nabla_\theta \int_{\mathcal{E}} q(\epsilon) r(s_{t_1},a)|_{a=\mu_\theta(s_{t_1},\epsilon)} \\
&+ \gamma \int_S p(s_{t_2}|s_{t_1},a)|_{a=\mu_\theta(s_{t_1},\epsilon)} \int_{\mathcal{E}} p(\epsilon') Q^\theta(s_{t_2}|s_{t_1},a)|_{a=\mu_\theta(s_{t_1},\epsilon')} d\epsilon' d\epsilon d\tau.
\end{aligned}$$

We keep expanding on the last part:

$$\begin{aligned}
\nabla_\theta V^\theta(s_{t_0}) &= \mathbb{E}_{q(\epsilon)}\left[\nabla_\theta \mu_\theta(s_{t_0},\epsilon)\nabla_a Q^\theta(s_{t_0},a)|_{a=\mu_\theta(s_{t_0},\epsilon)}\right] \\
&+ \int_{\mathcal{S}} p(s_{t_0\to t_1}|\pi)\gamma \int_{\mathcal{E}} q(\epsilon)\nabla_\theta \mu_\theta(s_{t_1},\epsilon)\nabla_a r(s_{t_1},a)|_{a=\mu(s_{t_1},w)} \\
&+ \gamma \int_S \nabla_\theta \mu_\theta(s_{t_1},\epsilon)\nabla_a p(s_{t_2}|s_{t_1},a)|_{a=\mu_\theta(s_{t_1},\epsilon)} \int_{\mathcal{E}} p(\epsilon') Q^\theta(s_{t_2}|s_{t_1},a)|_{a=\mu_\theta(s_{t_1},\epsilon')} d\epsilon' \\
&+ \gamma \int_S p(s_{t_2}|s_{t_1},a)|_{a=\mu(s_{t_1},w} \int_{\mathcal{E}} p(\epsilon')\nabla_\theta Q^\theta(s_{t_2}|s_{t_1},a)|_{a=\mu_\theta(s_{t_1},\epsilon')} d\epsilon' d\epsilon d\tau \\
&= \mathbb{E}_{q(\epsilon)}\left[\nabla_\theta \mu_\theta(s_{t_0},\epsilon)\nabla_a Q^\theta(s_{t_0},a)|_{a=\mu_\theta(s_{t_0},\epsilon)}\right] \\
&+ \int_{\mathcal{S}} p(s_{t_0\to t_1}|\pi)\gamma \int_{\mathcal{E}} q(\epsilon)\nabla_\theta \mu_\theta(s_{t_1},\epsilon)\nabla_a Q^\theta(s_{t_1},a)|_{a=\mu_\theta(s_{t_1},\epsilon)} \\
&+ \gamma \int_S p(s_{t_2}|s_{t_1},a)|_{a=\mu_\theta(s_{t_1},\epsilon)} \int_{\mathcal{E}} p(\epsilon')\nabla_\theta Q^\theta(s_{t_2}|s_{t_1},a)|_{a=\mu_\theta(s_{t_1},\epsilon')} d\epsilon' d\epsilon d\tau \\
&= \mathbb{E}_{q(\epsilon)}\left[\nabla_\theta \mu(s_{t_0},\epsilon)\nabla_a Q^\theta(s_{t_0},a)|_{a=\mu_\theta(s_{t_0},\epsilon)}\right] \\
&+ \int_{\mathcal{S}} p(s_{t_0\to t_1}|\pi)\gamma \int_{\mathcal{E}} q(\epsilon)\nabla_\theta \mu_\theta(s_{t_1},\epsilon)\nabla_a Q^\theta(s_{t_1},a)|_{a=\mu_\theta(s_{t_1},\epsilon)} d\epsilon d\tau \\
&+ \int_{\mathcal{S}} p(s_{t_0\to t_2}|\pi)\gamma^2 \int_{\mathcal{E}} q(\epsilon)\nabla_\theta \mu_\theta(s_{t_1},\epsilon)\nabla_a Q^\theta(s_{t_2},a)|_{a=\mu_\theta(s_{t_1},\epsilon)} d\epsilon d\tau \\
&\vdots \\
&= \int_{S\times\mathcal{E}} \sum_{t=t_0}^{\infty} \gamma^{t-t_0} p(s_{t_0\to t})\nabla_\theta \mu_\theta(s_t,\epsilon)\nabla_a Q^\theta(s_t,a)|_{a=\mu_\theta(s_t,\epsilon)} ds_t d\epsilon.
\end{aligned}$$

We notice $Q^\theta(s_t,a)|_{a=\mu_\theta(s_t,\epsilon)}$ is not dependant of t. So we obtain by injecting the previous

result into Equation (2.19) and using the Leibniz integral rule:

$$\begin{aligned}
\nabla_\theta J(\theta) &= \nabla_\theta \int_S p_0(s) V^\theta(s) ds - \alpha \nabla_\theta KL(q(w|\theta)|p(w|\theta_{\text{prior}})) \\
&= \int_S p_0(s_{t_0}) \nabla_\theta V^\theta(s_{t_0}) ds_{t_0} - \alpha \nabla_\theta KL(q(w|\theta)|p(w|\theta_{\text{prior}})) \\
&= \int_S p_0(s_{t_0}) \int_{S\times\mathcal{E}} \sum_{t=t_0}^{\infty} \gamma^{t-t_0} p(\tau_{t_0\to t}) \nabla_\theta \mu_\theta(s_t, \epsilon) \nabla_a Q^\theta(s_t, a)|_{a=\mu_\theta(s_t,\epsilon)} ds_t d\epsilon ds_{t_0} \\
&\quad -\alpha \nabla_\theta KL(q(w|\theta)|p(w|\theta_{\text{prior}})) \\
&= \int_{S\times\mathcal{E}} \sum_{t=t_0}^{\infty} \gamma^{t-t_0} \int_S p_0(s_{t_0}) p(\tau_{t_0\to t}) \nabla_\theta \mu_\theta(s, \epsilon) \nabla_a Q^\theta(s, a)|_{a=\mu_\theta(s,\epsilon)} ds d\epsilon \\
&\quad -\alpha \nabla_\theta KL(q(w|\theta)|p(w|\theta_{\text{prior}}))
\end{aligned}$$

We have the state-visitation measure from Equation (2.12):

$$\begin{aligned}
&= \int_S \rho^\pi(s) \nabla_\theta \mu_\theta(s, \epsilon) \nabla_a Q^\theta(s, a)|_{a=\mu_\theta(s,\epsilon)} d\tau - \alpha \nabla_\theta KL(q(w|\theta)|p(w|\theta_{\text{prior}})) \\
&= \mathbb{E}_{\rho^\pi}\left[\nabla_\theta \mu_\theta(s, \epsilon) \nabla_a Q^\theta(s, a)|_{a=\mu_\theta(s,\epsilon)}\right] - \alpha \nabla_\theta KL(q(w|\theta)|p(w|\theta_{\text{prior}})) \\
&= \mathbb{E}_{q(w|\theta),\rho^\pi}\left[\nabla_\theta \mu_\theta(s, \epsilon) \nabla_a Q^\theta(s, a)|_{a=\mu_\theta(s,\epsilon)} ds + \alpha \left(\nabla_\theta \log p(w|\theta_{\text{prior}}) - \nabla_\theta \log q(w|\theta)\right)\right].
\end{aligned}$$

□

2.7.2 Likelihood and Compression Cost

The parameters are drawn from the posterior to compute this gradient. In the set $\theta = (W_\mu^l, W_\rho^l, \epsilon_W^l, B_\mu^l, B_\rho^l, \epsilon_B^l)_{l\in L}$ we write $\eta = (\eta_\mu, \eta_\rho)$, with η_μ, η_ρ the corresponding parameters in the parameters' matrix either from W or from B. We consider the gradients of the loss with respect to the parameters given by:

$$\begin{aligned}
\log(q(w|\theta)) &= \sum_{\eta^*\in w} \left[-\frac{1}{2}\left(\log(2\pi) + 2\log(\log(1+\exp(\eta_\rho))) + \frac{(\eta^* - \eta_\mu)^2}{\log^2(1+\exp(\eta_\rho))}\right)\right], \\
&= \sum_{\eta^*\in w} \left[-\frac{1}{2}\left(\log(2\pi) + 2\log(\log(1+\exp(\eta_\rho))) + \epsilon^2\right)\right],
\end{aligned}$$

$$\frac{\partial \log(q(w|\theta))}{\partial \eta_\mu} = 0,$$

$$\frac{\partial \log(q(w|\theta))}{\partial \eta_\rho} = -\frac{1}{(1+\exp(-\eta_\rho))\log(1+\exp(\eta_\rho))}.$$

Moreover, we have the prior:

$$\log(p(w|\theta_{\text{prior}})) = \sum_{\eta^*\in w} \left[-\frac{1}{2}\left(\log(2\pi) + 2\log(\log(1+\exp(\eta_\rho^{\text{prior}}))) + \frac{(\eta^* - \eta_\mu^{\text{prior}})^2}{\log^2(1+\exp(\eta_\rho^{\text{prior}}))}\right)\right],$$

When injecting the reparametrisation trick, it comes from ((2.7)):

$$\frac{\partial p(w|\theta_{\text{prior}})}{\partial \eta_\mu} = \frac{\eta_\mu^{\text{prior}} - \eta_\mu - \epsilon \log(1+\exp(\eta_\rho))}{\log^2(1+\exp(\eta_\rho^{\text{prior}}))},$$

It comes that:

$$\frac{\partial p(w|\theta_{\text{prior}})}{\partial \eta_\rho} = \frac{\epsilon(\eta_\mu^{\text{prior}} - \eta_\mu) - \epsilon^2 \log(1+\exp(\eta_\rho))}{(1+\exp(-\eta_\rho))(\log^2(1+\exp(\eta_\rho^{\text{prior}})))}.$$

Then we have the gradients:

$$\nabla \eta_\mu = \frac{\alpha}{\log^2(1+\exp(\eta_\rho^{\text{prior}}))} \left[\eta_\mu^{\text{prior}} - \eta_\mu - \epsilon \log(1+\exp(\eta_\rho))\right] - \frac{1}{|S|} \sum_{s \sim S} \frac{\partial Q(s, \pi(s,w))}{\partial \pi(s,w)} \frac{\partial \pi(s,w)}{\partial \eta_\mu}.$$

and for the $\nabla \eta_\rho$:

$$\nabla \eta_\rho = \frac{\alpha}{1+\exp(-\eta_\rho)} \left[\frac{\epsilon(\eta_\mu - \eta_\mu^{\text{prior}})}{\log^2(1+\exp(\eta_\rho^{\text{prior}}))} + \frac{\epsilon^2 \log^2(1+\exp(\eta_\rho)) - \log^2(1+\exp(\eta_\rho^{\text{prior}}))}{\log^2(1+\exp(\eta_\rho^{\text{prior}})) \log(1+\exp(\eta_\rho))}\right] - \frac{1}{|S|} \sum_{s \sim S} \frac{\partial Q(s, \pi(s,w))}{\partial \pi(s,w)} \frac{\partial \pi(s,w)}{\partial \eta_\rho}$$

Considering the reparametrisation, $\sigma = \log(1+\exp(\eta_\rho))$ and $\sigma_{\text{prior}} = \log(1+\exp(\eta_\rho^{\text{prior}}))$:

$$\nabla \eta_\rho = \frac{\alpha(\exp(\sigma)-1)}{\exp(\sigma)} \left[\frac{\epsilon(\eta_\mu - \eta_\mu^{\text{prior}})}{\sigma_{\text{prior}}^2} + \frac{\epsilon^2\sigma^2 - \sigma_{\text{prior}}^2}{\sigma_{\text{prior}}^2 \sigma}\right] - \frac{1}{|S|} \sum_{s \sim S} \frac{\partial Q(s, \pi(s,w))}{\partial \pi(s,w)} \frac{\partial \pi(s,w)}{\partial \eta_\rho}.$$

By taking the expectation over the random variable ϵ:

$$\mathbb{E}_{q(\epsilon)}\left[\nabla \eta_\mu\right] = \frac{\alpha}{\log^2(1+\exp(\eta_\rho^{\text{prior}}))} \left[\eta_\mu^{\text{prior}} - \eta_\mu\right] - \frac{1}{|S|} \sum_{s \sim S} \mathbb{E}_{q(\epsilon)} \left[\frac{\partial Q(s, \pi(s,w))}{\partial \pi(s,w)} \frac{\partial \pi(s,w)}{\partial \eta_\mu}\right],$$

and

$$\mathbb{E}_{q(\epsilon)}\left[\nabla \eta_\rho\right] = \frac{\alpha(\exp(\sigma)-1)}{\exp(\sigma)} \left[\frac{\frac{1}{2\sqrt{2}}\sigma^2 - \sigma_{\text{prior}}^2}{\sigma_{\text{prior}}^2 \sigma}\right] - \frac{1}{|S|} \sum_{s \sim S} \mathbb{E}_{q(\epsilon)} \left[\frac{\partial Q(s, \pi(s,w))}{\partial \pi(s,w)} \frac{\partial \pi(s,w)}{\partial \eta_\rho}\right].$$

The gradients inform us of the importance of α in the gradients. We can directly observe its impact on the parameters trajectories of our agent in our plain environment presented in Section 3.3. The parameter trajectories are shown in Figure 3.7

Architecture Considered

In DRL many configurations are used for actors and critics. For the different experiments, these are the architectures we considered:

parameter	HalfCheetah-v3	LLC-v2	MCC-v0
Actor	400-300	250 - 250	50 - 50
Critic	(64 - 64) - 350 - 350	(64 - 64) - 256 - 256	(50 - 50) - 512 - 512
rho_init	-3.	-2.	-1.5
batch size	256	256	256
learn rate	0.0005	0.0003	0.003
alpha	0.000001	0.00005	0.000001
batch length	1 000 000	500 000	100 000
activation (actor-critic)	tanh-relu	tanh-relu	tanh-relu

The main idea is to use the best configurations for the algorithm. The results presented in table 2.1 show the performance of the algorithms kept as references. The performances we obtained with our model show the possibilities of performance. For the applications, we used the following configurations:

parameter	Plain Environment	Physical Application
Actor	20 - 20	30 - 30
Critic	(16 - 16) - 64 - 64	(32 - 32) - 400 - 400
rho_init	-3.	-3.5
batch size	64	1024
learn rate	0.001	0.0001
alpha	0.0001	0.00003
batch length	100 000	10 008 000
activation (actor-critic)	tanh - relu	tanh - relu

Those architectures were chosen by starting using commonly used parameters such as those found on the implementations of stable-baseline [70]. They have published the parameters associated with the runs they made, giving standard parameters to use. Then closely watching the learning gives much information on the performance of the parameters, but diagnosing the source of an issue is still complicated.

Chapitre 3

Application to Data Center Simulation

3.1 Introduction

Energy consumption due to air conditioning systems in DCs and buildings is a hot topic. With the emergence of Reinforcement Learning (RL) and Deep Learning (DL), we have dynamic optimisation tools to reduce infrastructure energy consumption (see [92] [106] [18]). This work assesses the difficulty of using RL to control building air conditioning simulation and discusses the application in actual conditions. We expect to find better regulation policies to diminish the consumption of air conditioning systems. Data-driven approaches have the advantage of not depending on the definition of the physical model and can have a high capacity to model reality. Modelling continuous state-action environments, like HVAC control, is a challenge that RL has recently started tackling. Building's thermodynamics tend to act as a filter for high frequencies ventilation variation due to physical properties [4]: "There are some processes (e.g., dynamics of zone temperature and humidity) in the HVAC system that are known as slow-moving due to the substantial thermal inertia of the system". As highlighted in another paper [93], the inertia phenomenon impacts energy consumption to control the temperature: "The thermal inertia effect can also directly influence the heating and cooling demand of a building and the resulting energy consumption". We focus on on-policy algorithms because it is easier to obtain stable results. Still, off-policy agents can be trained from previous data and seem more adapted to real-world applications. The learning can start as soon as we have data, and we can learn the policy before deploying the model. Improving the quality of control of a system that has already been in service for a while and has generated a lot of Data is very interesting. However, the current methods are inefficient, mainly due to the difficulty of assessing a counterfactual policy. Training on-policy models in actual life conditions is a real issue as the Data generation process is much longer, and the consideration for the training must be different. Also, the usual exploration strategies are impracticable as the high-frequency variations of fan inputs may damage the material. We analyse a state-dependent noise (see [61]) to explore the environment and overcome inertia filtering. We use DL with Bayesian consideration for encoding the policy during the learning. To learn models, we built an environment for RL on a nodal model of a building heated room with a fan system. Physical behaviours are computed through differential equations between the nodes of the system. The model is parameterised to reproduce complex behaviour such as inertia. The purpose is to estimate the gain of machine learning models over standard

methods and assess the current issues to learn policies for fan systems. The performance of DRL is encouraging, we are getting closer to applying models in actual conditions (see [32]). But there are also supplementary constraints that are not observable in simulations, such as the delays between an action and its impact on the state and the fact that a measure of the temperature is a proxy for the actual state of the system. Moreover, the actual conditions are far more complex as weather data can be chaotic. RL is very promising, but understanding the possibilities and how to improve the framework and algorithms is necessary to scale to larger environments and deploy them in genuine conditions.
The main contributions of this chapter are the following:

— We developed and analysed an environment to simulate thermodynamics in a room with differential equations on a nodal system. This environment assesses the response to ventilation and how to control the temperature under temperature inertia.
— We use our algorithm to optimise this environment using state-dependant exploration and discuss its application in real conditions.

The issues of using artificial intelligence in physical control applications are not clear, and they need to be more exposed to go further with RL applications. We first use numerical simulation to understand how our model converges and the impact of the parameters on learning. Then we focus on the physical simulation and compare our results with optimized baseline models. Moreover, we discuss the difficulties of implementing DRL in real life. Section 3.4 describes the modelled environment using a nodal system, while Section 3.5 presents the results and discusses the application in real conditions.

3.2 Related Work

Applications that use RL in a physical and environmental context have become appealing. The research on nuclear fusion and plasma control has shown impressive results [16]. Some research contributions already employ RL to optimise air distribution in DCs (see [92] [106] [18]). The possibilities with RL and DL seem very appealing, but in reality, using those algorithms is not easy and has several limitations. We focused on analysing how to do RL on plain application and what the limits are. The knowledge of the DRL applications issues is poor, whereas the instability and the difficulties in reproducing previous work are well-known in the research [42]. When using DRL, the DL concerns add up to the RL issues. Nevertheless, research contributions (see [32]) make applications more realistic. Many environments deal with robotics, but only some models are available for physical system control. HVAC energy-plus models are great tools for experimenting [39], but it is hard to master. Its complexity makes it hard to understand the causes of the issues encountered. DL Meta-modelization is a hot topic, but explainability and the dependence on the data support are also critical limitations [99].

3.3 Numerical simulation

To explore the behaviour of our model, we develop a plain environment. The environment contains a ball, which an actor can move left or right at each step. We use a Gaussian function for the reward evaluation depending on the ball position. Figure 3.5 represents

this environment. The reward is associated with the height of the ball. The point of this environment having only one continuous state and one continuous action is to be easy to visualize in its plainest form. The environment is easy to solve but can be complexified by changing the reward function. For example, using a mix of Gaussians instead of a single Gaussian. The purpose of this environment is to describe how our model works and the impact of the α temperature parameter and the initialisation on learning. For convenience, we initialize the prior and posterior on similar configurations.

FIGURE 3.1 – Simple continuous environment for RL, the point is to train an agent to move the ball to the highest position. We maximise the reward when the ball is uphill.

Figures 3.2 and 3.3 show the behaviour of our actors under a high and low α value. As shown in Equation (2.19), α impacts the loss and accentuates the compression or likelihood cost. As previously shown, $\frac{1}{\alpha}$ could be directly used as a reward scaler. As we use an exponential distribution over the critic estimation, the scaling sharpens the values making the gradient higher, while lower $\frac{1}{\alpha}$ smooths the estimation and the gradients. We directly observe the effect of the α on the figures. Low α makes the network parameters find good values for the critic, favouring the left and right movements. High values impact the model dispersion, making prior and posterior closer and diminishing the ρ parameters associated with the uncertainty of the network. Higher or lower α might be preferred depending on the environment. Larger α tends to benefit more from exploration, while lower α should learn faster ; this impacts the exploration and exploitation dilemma [86]. From this example, we can also benefit from the parallelisation of the environment. Several balls are controlled by the neural networks generated from our Bayesian neural network actor. For every episode, the experience is run with different policies generating more data and exploring the environment faster.

Deep Bayesian Actor
$\alpha = 0$, $\rho = -2$, layers $= [20, 20]$

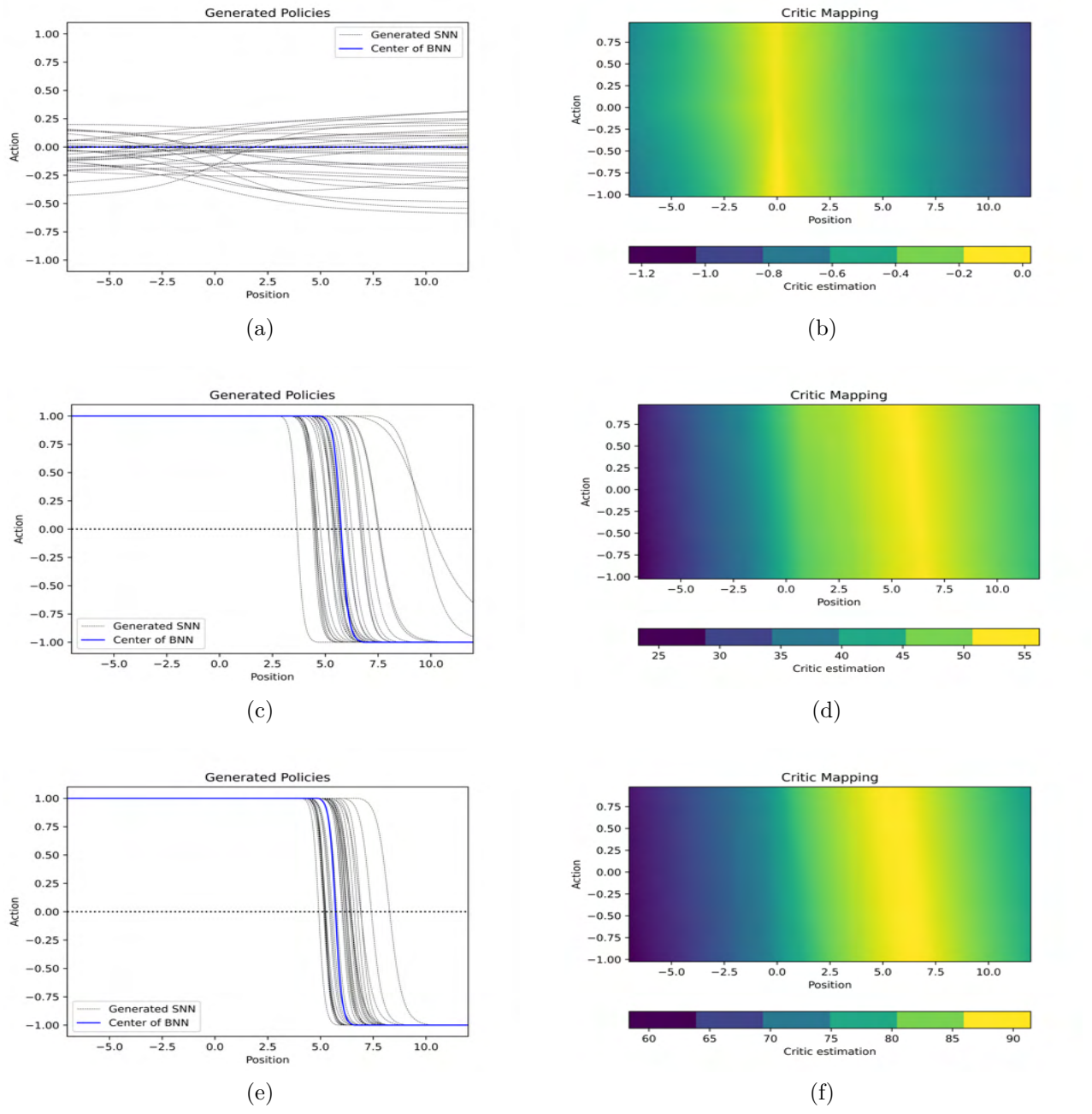


FIGURE 3.2 – Learning of the Bayesian Neural Network agent in our plain environment. The left part represents the agent and the policy generated for the exploration, while the right part represents the gain estimated by the critic depending on the states and actions.

Deep Bayesian Actor
$\alpha = 10^{-3}$, $\rho = -2$, layers $= [20, 20]$

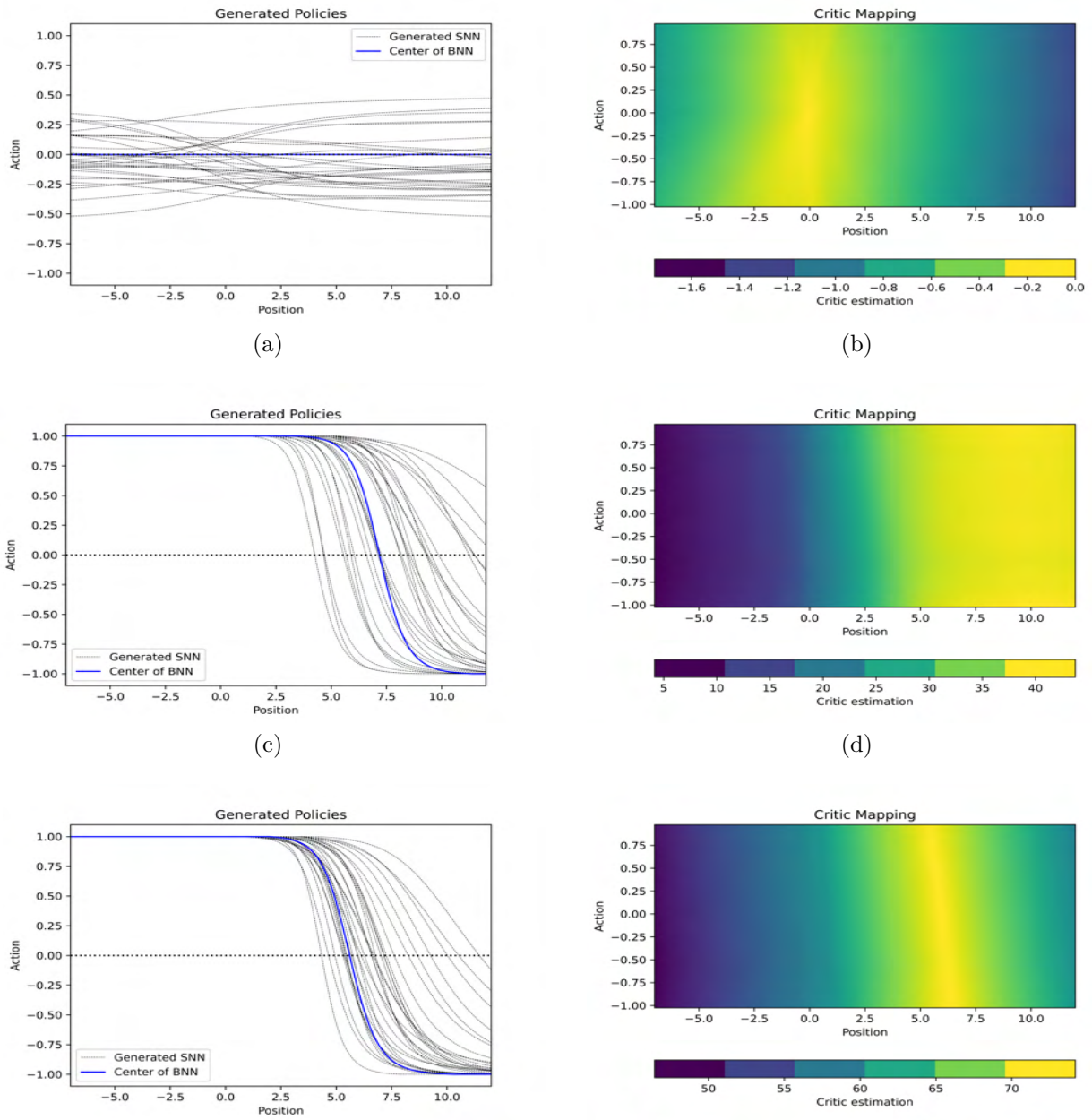


FIGURE 3.3 – Learning of the Bayesian Neural Network on our plain environment for alpha equals 10.

Deep Bayesian Actor
$\alpha = 0$, $\rho = -3$, layers $= [20, 20]$

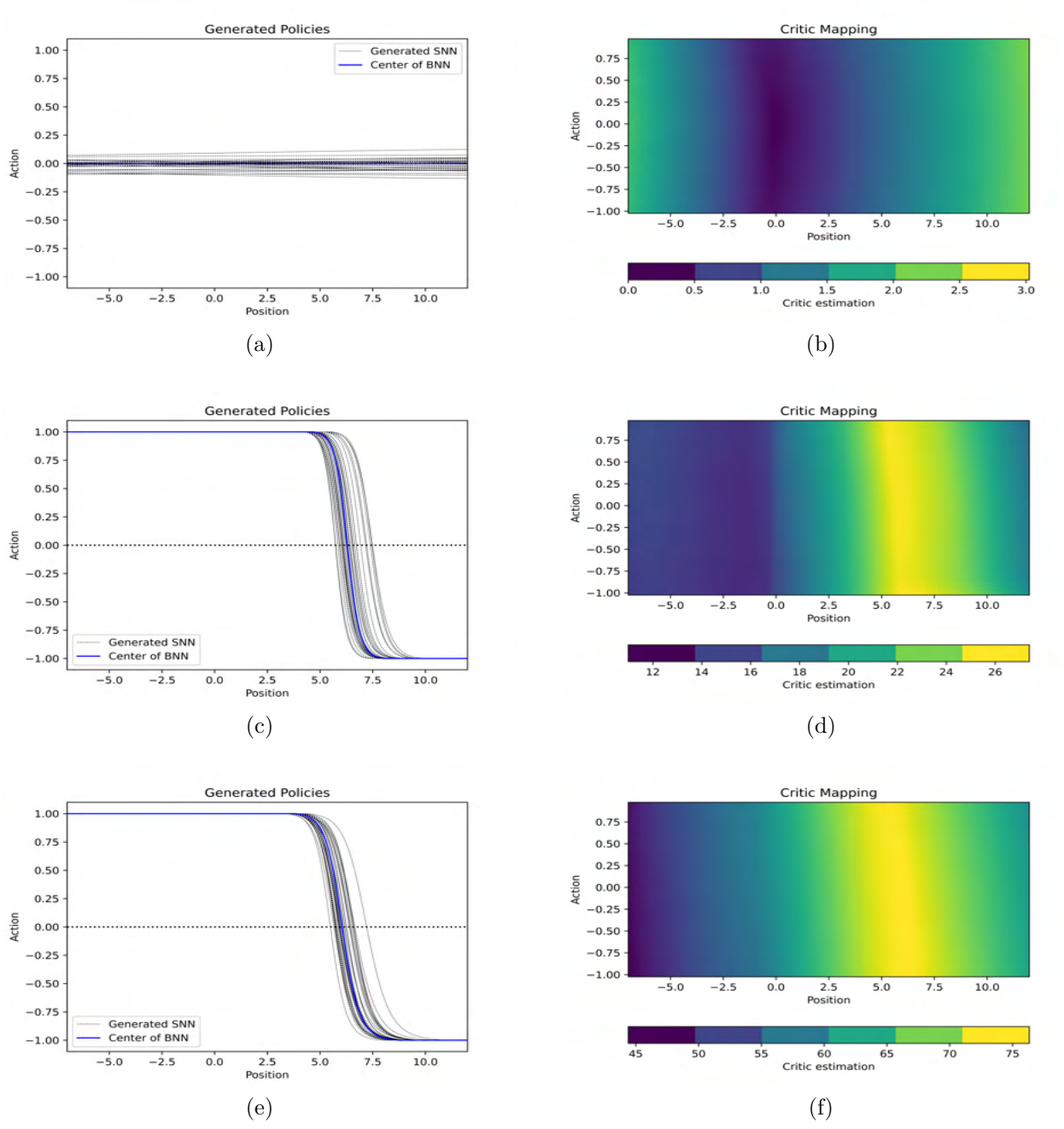


FIGURE 3.4 – Learning of the Bayesian Neural Network on our plain environment for alpha equals 0.

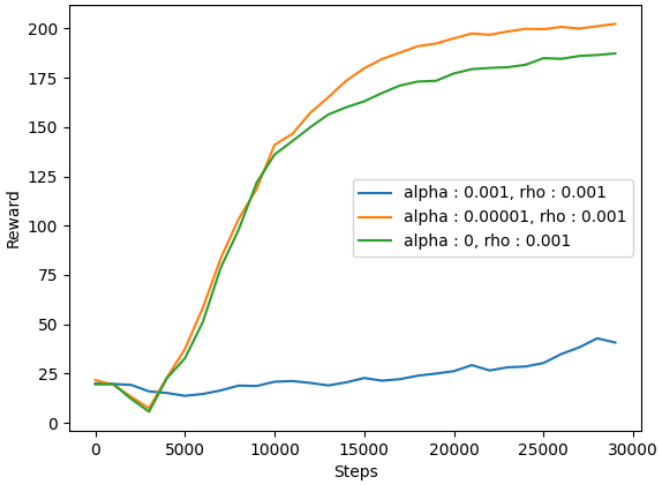


FIGURE 3.5 – Comparison for different α. We ran one hundred runs and averaged the rewards. The best configuration is for $\alpha = 0.00001$.

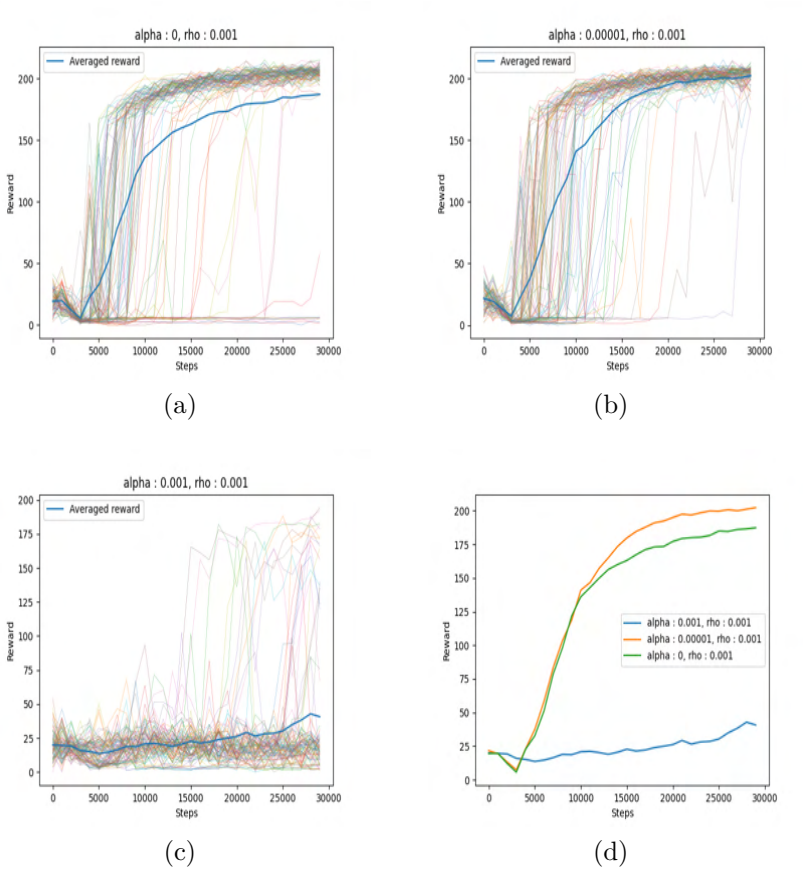


FIGURE 3.6 – The trajectories of the rewards and averaged reward of one hundred runs for different α. The last figure compares the average reward for the different α values. The best value is 10^{-5}. For α equals, some runs don't converge, while for $\alpha = 10^{-5}$, all the runs have converged.

Deep Bayesian Actor Parameter Trajectories

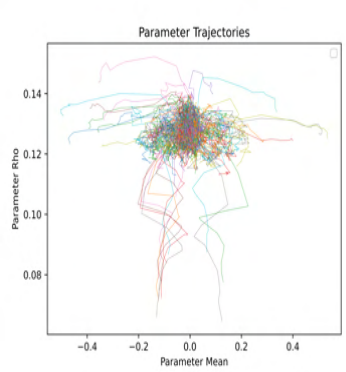
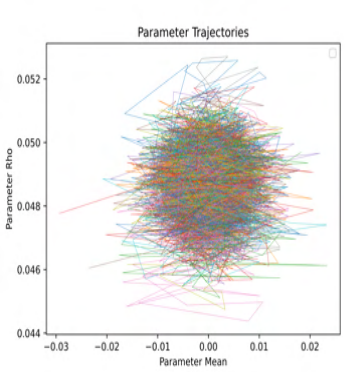
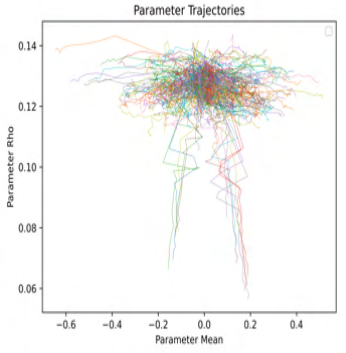
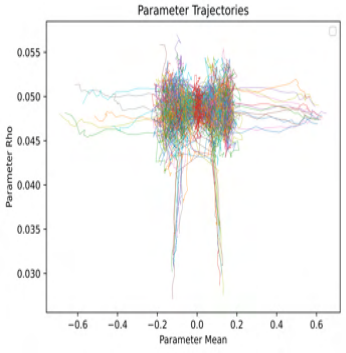
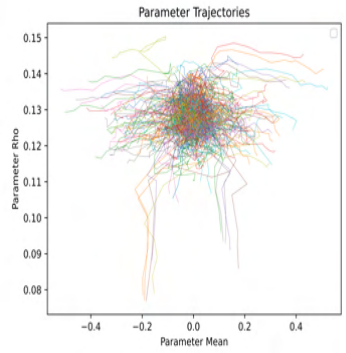
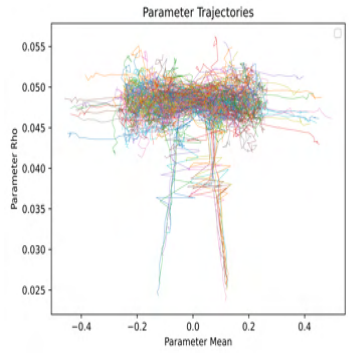
	$\rho = -2$	$\rho = -3$
$\alpha = 10^{-3}$	 (a)	 (b)
$\alpha = 10^{-5}$	 (c)	 (d)
$\alpha = 0$	 (e)	 (f)

FIGURE 3.7 – The trajectories of the Bayesian Neural Network parameters during the learning for 30,000 steps. A parameter's horizontal movements alter the parameter's mean, while vertical movements alter the deviation of the parameter. We can observe the impact of α and the η_ρ of the prior on the parameters trajectory.

3.4 Physical Nodal Environment

This chapter introduces an environment for practising RL methods based on a physical nodal system. The point is understanding how they tackle these challenges and whether we can deploy them in real situations. Applying models in actual conditions is complicated due to the cost of IT material and associated risk. Meta models of DCs are a good choice for performing RL, but they tend to be complicated to analyse the basics of thermodynamics. The physical model generates Data behaving similarly to real-world Data. The point of this environment is to reproduce physical behaviours and to raise the difficulties of handling them. We highlight the issues we encountered while solving physical environments with DRL and give our key to understanding and overcoming standard issues. The model's simplicity eases the understanding of the physical issues associated with the optimisation. We deploy our DRL algorithm based on a parameters search for the policy. We explain our choices and give the key to correctly understanding and using the different hyper-parameters. We discuss the perspective of scaling up and applying this kind of methodology to real-world applications. Nodal models are basic modelisations of thermal exchanges when cooling a building room. They are built on differential equations and energy conservation. They offer a scalable parametrisation for the room's characteristics and complexity. Heat dissipation equations depict the environment's mechanisms. A simulated IT load heats the room. We model a fan system to adjust the internal temperature using the external temperature in a conventional free-cooling approach. The nodal system is a function that computes the temperature nodes, representing the temperature at different locations within the room. We have a node for the ambient temperature inside the room and another for the external temperature. According to the building's structure, the remaining nodes represent the temperatures within the walls. This function inputs the preceding nodes' temperatures, the external temperature, the ventilation flow, and the dissipated power. This example, with a few features, helps to understand the behaviour of more complicated physical systems. It has some challenging properties for RL. When adjusting its parameters, the nodal representation may accumulate energy inside the walls, which adds inertia to the system. Those plain properties are ideal for visualizing the challenges associated with RL and physics before deploying models. When deploying a model on a more complex environment, understanding the causes of convergence issues is far more complicated. The purpose of this environment is to assess the difficulties of deploying RL to solve specific environments. The model is a partially observed environment, as the temperature of the walls' layers is unseen from the agent's perspective. The known variables from the agent's perspective are the inside and outside temperatures. In our experiments, for simplicity, we assumed the power dissipated is constant, and the outside temperature follows a sinus to represent the daily temperature variation. We also considered the temperature data measured in Mimizan in France on a telecom site, To help the agent, we utilise the cosinus and sinus of the daily hour for the agent to estimate the temperature evolution. To increase the Data volume for learning, we used a process on each episode using two random variables, the first following a normal law centred on one and a centred normal law. We drew them at each environment reset and multiplied the temperature of each step by the first one, then added the second one. Figure 3.16 represents the Data as time series and shows its distribution.

Figure 3.8 represents the physically modelled system. The physical parameters can be

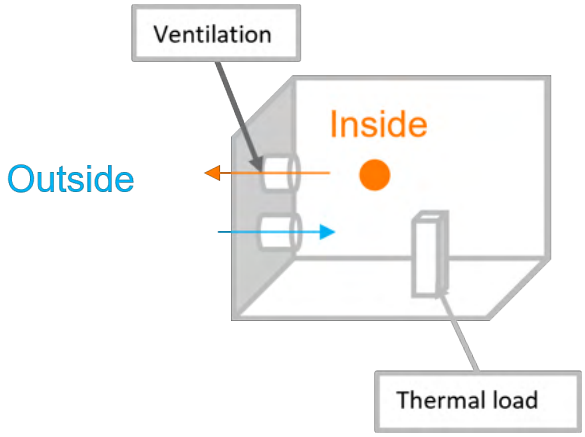


FIGURE 3.8 – Schema of a ventilated room with a thermal load.

tuned to adjust the room's physical properties. The size of the wall or the materials can change, changing thermal diffusion properties. The following system gives the physical equations representing the system $\forall i \in [0, N]$ with N the number of nodes:

$$\rho_i C_i V_i \frac{\partial T_i}{\partial t} = \begin{cases} 2\left(\frac{T_{i+1}-T_i}{R_{i+1}+R_i} + \frac{T_{ext}-T_i}{2R_{cv}+R_i}\right) & \text{for external nodes.} \\ 2\left(\frac{T_{i-1}-T_i}{R_{i-1}+R_i} + \frac{T_{i+1}-T_i}{R_{i+1}+R_i}\right) & \text{for internal nodes.} \\ 2\left(\frac{T_{i+1}-T_i}{R_{i+1}+R_i} + \frac{T_{air}-T_i}{2R_{cv}+R_i}\right) & \text{for surface nodes.} \end{cases}$$

And for the air temperature T_{air} node, we have:

$$\rho_j C_j V_j \frac{\partial T_{air}}{\partial t} = \sum_{i \in S_{int}} \left(\frac{T_i + \frac{R_i}{2R_{cv}} T_{air}}{2R_{cv} + R_i} \right) - |S_{int}| \frac{T_{air}}{R_{cv}} + P_{eq} + q_m . C_m \left(T_{ext} - T_{air}\right). \tag{3.1}$$

The parameters used in the model are:

ρ	Volumic mass	kg/m^3
C	Specific heat	J/kg
V	Volume	m^3
R	Thermal resistance	$°C/W$
P_{eq}	Power used by inside equipment	W
q_m	Air mass used to ventilate	m^3/s
T_{ext}	Temperature outside the building	$°C$
T_{air}	Ambient temperature inside the room	$°C$

The action variable of the system is the air mass used to ventilate q_m. It is a continuous variable. The higher the value is, the higher the ventilation blows. Ventilation is the only way to regulate the temperature, but in most cases, we also want to minimize its utilization as it increases the energy consumption of the building. Adding nodes for the wall can be associated with increasing the wall's number of layers, which leads to delay and variation of

the heat going through the wall depending on the materials. To solve this system, we used the implicit method for stability and coherence in computations. The main drawback is the time used for computing every timestep. However, with the current computation power, it is possible to train RL agents on this model. However, the difficulties for RL agents are:

— The unobserved nodes delay the impact of the actions
— The exploration of the internal nodes is complex due to the dimension and the delay
— The stochasticity of the external temperature has a high impact on the transition of the environment
— The model has to anticipate and make counter-intuitive decisions.

Power Consumption

The profile of consumption of the fan system is usually polynomial [62], where the consumption depends on a polynomial of order 4 of the width (in mm), the depth (in mm), and the revolutions per minute of the fan. For simplicity, we assume the fan system has a consumption that depends only on the debit q_m with the following equation:

$$P(q_m) = \frac{1}{2} q_m^2. \tag{3.2}$$

We could consider a more complex and realistic consumption profile. Still, with this profile, we can appreciate the behaviour of the agent and system state evolution. The point is understanding how an agent tackles inertia to optimize non-linear energy consumption.

3.4.1 Reward

We consider the following objective: The policy must keep the temperature inside the authorised range and limit the distance to the range as best as possible when out. When the temperature constraint is respected, energy consumption must be minimised. Concerning the reward, we choose to create a penalisation in two terms. The first one penalises energy consumption, as we assume the consumption is a function of the debit. Then the second term is linked to the inside temperature, and if the inside temperature is in the authorised temperatures, it takes the value zero. Otherwise, it comprises a polynomial term of the difference between the temperature and the closest temperature limit. The first term takes the maximal penalisation possible as we expect to respect the temperature constraint before minimising the energy consumption. The sparsity of the reward is due to the discrete issue of being or not in the acceptable range of temperatures. The model is not wrong until the temperature gets out of range. When optimizing energy consumption, the temperature has to get close to the maximum authorized temperature. Sparsity in rewards is known to be an issue [40].

We use the following function as a reward, with the temperature $T \in \mathbb{R}$, the power used for the fan $P \in \mathbb{R}$, the minimal and maximal temperature (T_{min}, T_{max}) to not damage IT material, and the maximal power for the fan system P_{max}:

$$\begin{aligned} R_\tau\colon \quad \mathbb{R} &\longrightarrow \mathbb{R} \\ T &\longmapsto -\left([T - T_{max}]_+ + [T_{min} - T]_+\right)^k, \end{aligned}$$

$$
\begin{aligned}
R_{\mathcal{P}}\colon \quad \mathbb{R}^2 &\longrightarrow \mathbb{R} \\
T, P &\longmapsto -\mathbb{1}_{\{R_\tau(T)>0\}} P_{max} - \mathbb{1}_{\{R_\tau(T)=0\}} P,
\end{aligned} \tag{3.3}
$$

$$
\begin{aligned}
R\colon \quad \mathbb{R}^2 &\longrightarrow \mathbb{R} \\
T, P &\longmapsto R_\tau(T) + R_{\mathcal{P}}(T, P).
\end{aligned}
$$

This reward always ensures temperature control before energy consumption. When out of bounds, the energy consumption penalty is maximal. More common approaches consider a sum of temperature and energy components. We consider that a violation of the temperature constraint is always worse than any over-consumption. For the baseline, we add a term based on the variation of the action. It is penalised when the variation changes three times of sign consequently. We introduce this term because otherwise, the model uses steep action variation leading to instability. In actual conditions, this kind of behaviour may damage the fans.

To work with the assumptions of the policy gradient theorems 2.3.3, we have to add a ramp to limit the derivative of the reward at the authorized temperature limit. We can use a real $\xi > 0$ to obtain the following expression from Equation (3.3):

$$
R_{\mathcal{P}}(T, P) = \begin{cases}
-(1 - \frac{T_{max} - T}{\xi}) P_{max} - \frac{T_{max} - T}{\xi} P, & \text{if} \quad T \in [T_{max} - \xi, T_{max}] \\
-(1 - \frac{T - T_{min}}{\xi}) P_{max} - \frac{T - T_{min}}{\xi} P, & \text{if} \quad T \in [T_{min}, T_{min} + \xi] \\
\mathbb{1}_{\{R_\tau(T)>0\}} P_{max} - \mathbb{1}_{\{R_\tau(T)=0\}} P, & \text{otherwise,}
\end{cases}
$$

3.4.2 Baselines

The usual rule for fan systems is a linear model of the temperature getting as parameters minimal and maximal values for the fan usage and starting and maximal temperatures. We decided to create a baseline model which will be data optimized. We optimize a linear model with four parameters the maximal temperature, the minimal and maximal temperatures and the minimal and maximal actions. The action is considered constant outside the minimal and maximal values.

$$
\zeta(T) = clip(\frac{T - T_{min}}{T_{max} - T_{min}}, 0, 1), \tag{3.4}
$$

$$
a(T) = \zeta(T) \times (A_{max} - A_{min}) + A_{min}. \tag{3.5}
$$

With $A_{inf} \leq A_{min} \leq A_{max} \leq A_{sup}$, and A_{inf}, A_{sup} are the bounds of the action.
We also considered the following model:

$$
\zeta(T) = clip(\frac{T - T_{min}}{T_{max} - T_{min}}, 0, 1), \tag{3.6}
$$

$$
\tau(T, T_{ext}) = clip(\frac{T - T_{ext}}{C}, 0, 1). \tag{3.7}
$$

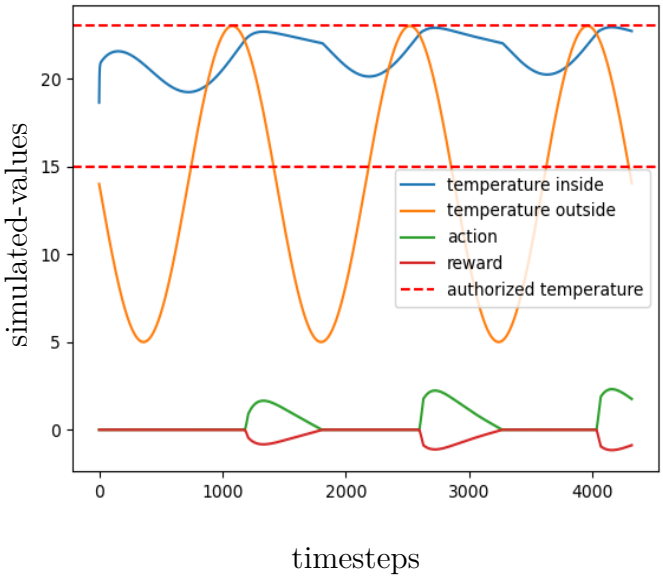


FIGURE 3.9 – The optimised ventilation based on the parametric exploitation of the temperature delta. This model ventilates when the temperature difference between outside and inside is negative, and the room requires cooling.

with C a constant chosen to maximize the reward. The policy is defined by:

$$a(T, T_{ext}) = \zeta(T) \times \tau(T, T_{ext}). \tag{3.8}$$

We use stochastic proximal optimisation to choose the parameters as the number of parameters is low, and the system is deterministic. We generate parameters randomly around a starting configuration and then update the parameter with an iterative mean or max operator based on a reward for each generated parameter. Once the model has converged, we check the model to challenge the physical possibility and realise correction if needed. This method depends on the low dimension of the parametric model and the deterministic environment with a short-term horizon. For a policy, we obtain stability in a few days at worst. Then we have a baseline model better than most regulations used in practice due to more knowledge of the environment and its deterministic behaviour. These baseline models can be overcome because of the non-linear energy consumption profile, and their parameterised form may be challenging to obtain the optimal policy.

3.4.3 Inertia and differential impact

Inertia is a physical phenomenon happening in building modelling. Due to heat transfer and their time counterpart, the heat may be absorbed by walls and then heated back to the inside temperature. This is dependent on the physical properties of the building. Specifically the density ρ, the specific heat C or the thermal resistance R of materials. This leads to delayed responses to variations. The air temperature variation in our model comes from Equation (3.1). We can estimate the following:

$$\rho_j C_j V_j \frac{\partial T_{air}}{\partial q_m} = \partial t . C_m \left(T_{ext} - T_{air} \right). \tag{3.9}$$

We notice the impact of actions on the system depends on the time-step size. Indeed, if we choose for the simulation a time-step very small, the ventilation will impact less the system state but will allow more precise regulation. Conversely, with a large time step, actions have a significant impact on the system, but the policy is less flexible. Considering the wall nodes, unobserved from the agent for surface nodes in contact with the air temperature:

$$\rho_i C_i V_i \frac{\partial T_i}{\partial t} = 2 \left(\frac{T_{i+1} - T_i}{R_{i+1} + R_i} + \frac{T_{air} - T_i}{2R_{cv} + R_i} \right),$$

and for internal nodes in contact with other wall nodes ;

$$\rho_i C_i V_i \frac{\partial T_i}{\partial t} = 2 \left(\frac{T_{i-1} - T_i}{R_{i-1} + R_i} + \frac{T_{i+1} - T_i}{R_{i+1} + R_i} \right),$$

we notice how the density ρ, the specific heat C or the thermal resistance R of material are impacting the latent state of the system. Changing materials and architecture affects the building's ability to stack and return energy. So the impact of the unobserved variables is mainly due to the building architecture and materials.

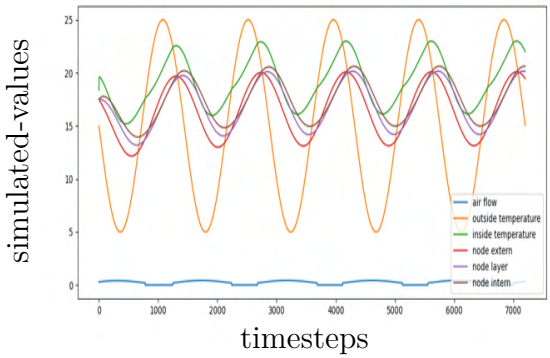
Figure 3.11 shows how the unobserved variables play a role in the state transition of the system. The main idea is that in an environment with inertia, an action impacts states far from the current state. The policy changes the unobserved variable, modifying the transition of the environment. So even if reaching an equivalent state at some point, the trajectory will differ from the expected one. Many environments are not entirely observable or Markovian, but using the usual RL algorithms still works in most cases.

When the unobserved variables are hidden, the Markovian property is broken. Indeed, there is a path between an action and the next states not expressed in the model through the unobserved variables. It induces a lag between the state-action pair and the response in the state.

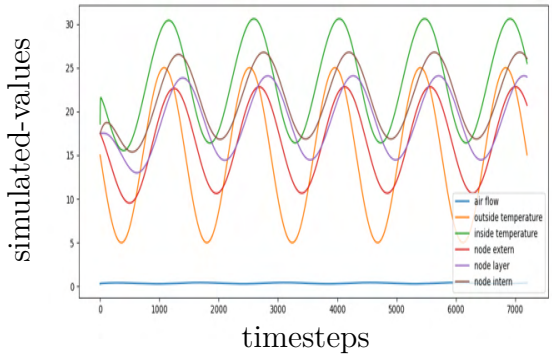
The non-stationarity and non-markovian transitions

The Bellman error is known to make a temporal smoothing instead of an accurate prediction, leading to, in some cases, poor approximations. Some situations may be hard to solve with function approximation and bellman error. The policy updates impact the reward in a non-markovian way by changing the transition probability. The latent states depend on the previous actions, and the transitions depend on the latent states. So, actions impact the reward with large delays. The issue can be summarized as follows, from a state, the following trajectory depends on previous states. Then as RL uses only the current state to estimate the return, the error will be high if the previous states have a high impact. If the critic estimation has a high variance or error, the actor will be led to the wrong policies. The more delayed the answer is, the more time the critic needs to be correctly updated. This is because

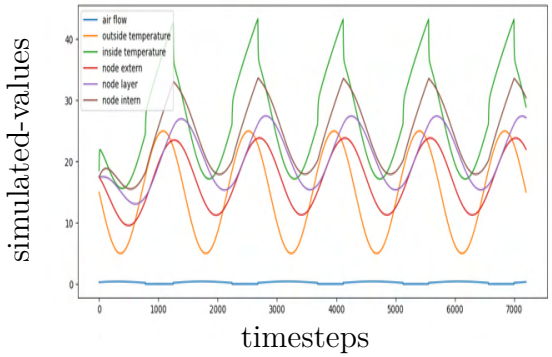
Interaction between the model's nodes and ventilation activation



(a) Time coherent noise



(b) No noise high debit

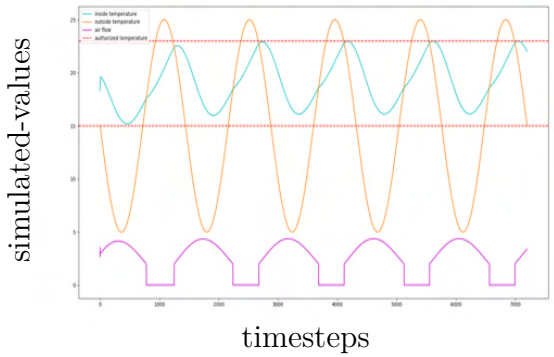


(c) No noise low debit

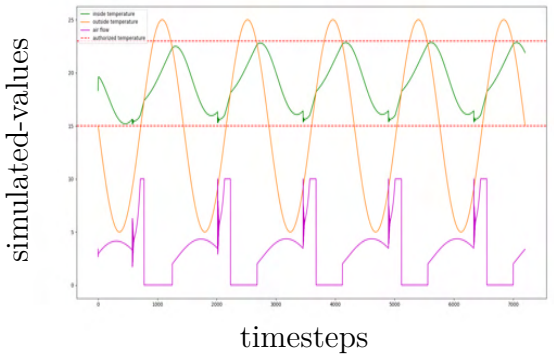
FIGURE 3.10 – The impact of wall material on the temperature regulation system of a room. The unseen nodes of the model are represented in the figure and show how the material's characteristics alter the temperature behaviour.

the return estimates with the Bellman objective are propagated from one state to the previous one.

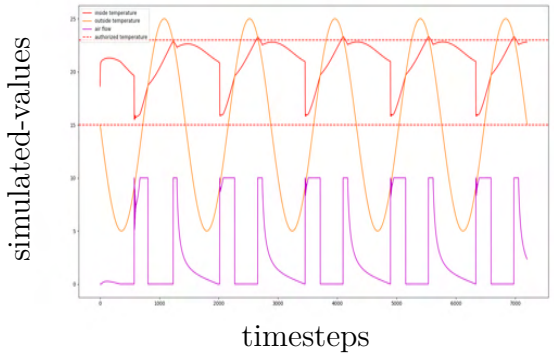
To illustrate this issue, we assume a simple environment with 4 time steps. It goes $T0 \rightarrow T1 \rightarrow T2 \rightarrow T3$. The environment starts at $T0$ in the state $S1$ with a null reward. To each other time step a state is associated $\{S1, S2, S3\}$ with their associated reward $\{0, 1, -10\}$. This example aims to find a policy avoiding the state $S3$. The agent can at $T0$ and $T1$ choose the next state, but at $T2$ the next state is randomly chosen with equal probability in one of the following collections: $\{S1, S2, S3\}$, $\{S2, S3\}$, $\{S3\}$, depending on the state at the timesteps $T1$ and $T2$:



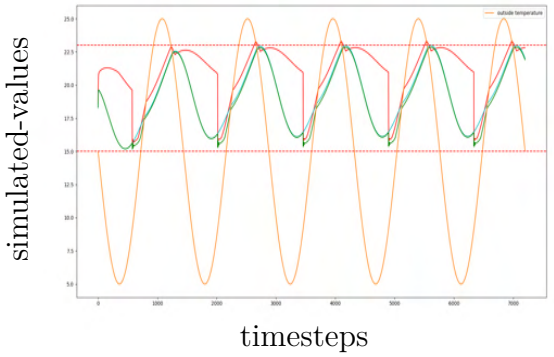
(a) Air flow policy cooling the room and the wall during the night



(b) Air flow policy cooling the room and the wall when outside temperature is at its lowest values, with an airflow increasing before the outside temperature exceeds the inside temperature



(c) Air flow policy neglecting the inertia phenomenon



(d) This figure compares the room temperature for all the models used. We can observe the impact of the previous states and actions on the temperature transition

FIGURE 3.11 – These policies show the impact of inertia on the state transitions of the system. The environment is not Markovian, as we can observe a policy that is not cooling over the night but lowering the temperature before the temperature starts increasing won't succeed in keeping the temperature in the authorized range as the temperature increases faster than the policies on the two other policies

$s_{(T1,T2)}$		Possible state at $T3$	Expected Return
$(S1, S1)$	$\rightarrow$	$\{S1, S2\}$	.5
$(S1, S2)$	$\rightarrow$	$\{S2, S3\}$	-3.5
$(S2, S1)$	$\rightarrow$	$\{S2, S3\}$	-3.5
$(S2, S2)$	$\rightarrow$	$\{S3\}$	-8

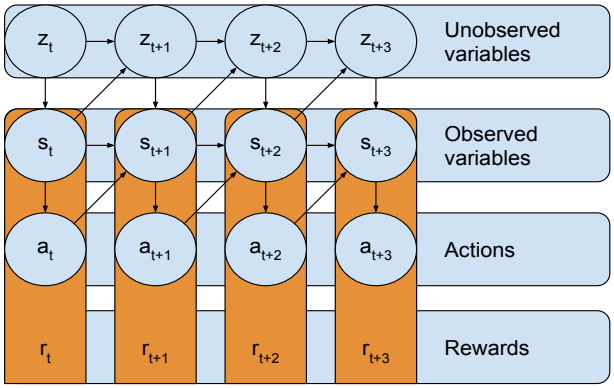


FIGURE 3.12 – The directed acyclic graph (DAG) represents the transition in a partially observable Markovian decision process. We assume the reward is computed only from the state and action at the same timestep, even if other setups exist. The arrows represent conditional dependencies of the variables

The point is to maximize the expected return. It is easy to see that the only trajectory avoiding the maximal penalty of -10 is by choosing $S1$ for the next state at time step $T1$ and $T2$, leading to randomly having between $S1$ and $S2$ at $T3$, the expected return is 0.5. Otherwise, the expected return is -3.5 if $T1$ or $T2$ is chosen to be $S2$ otherwise, the expected return is -8. Using an approximation method such as the TD error is tricky in this situation: Indeed, when choosing the state at $T2$, the conditional expected returns are given by:

$$\begin{aligned}\mathbb{E}\left[R|S_{T2}=S1\right] &= P(S_{T1}=S1)\times 0.5 - P(S_{T1}=S2)\times 3.5\\ \mathbb{E}\left[R|S_{T2}=S2\right] &= -P(S_{T1}=S1)\times 3.5 - P(S_{T1}=S2)\times 8.\end{aligned}$$

So whatever the state at $T1$, the best action is to choose $S1$ for $T2$. But when estimating the best choice at $T0$ as we use the Bellman equation, which assumes Markov property when computing the expected return, we have:

$$\mathbb{E}\left[R|S_{T1}=S1\right]+1=\mathbb{E}\left[R|S_{T1}=S2\right].$$

Thus the algorithm favours choosing $S2$ for $T1$, which is not the optimal policy.

This example is crucial to understand what is happening in our environment. It is an oversimplification of the system but gives intuitions on what is happening for critic estimation. Assuming it is related to our problem, the state can be considered as follows:

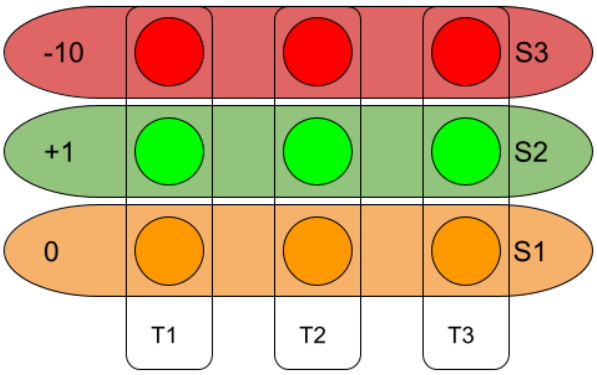
$S1$: The fans blow air and consume energy.

$S2$: The fans do not blow air and don't consume energy, but this leads to a temperature increase.

$S3$: The temperature overcomes the maximum authorized temperature, and the material is affected.

Even if the best solution is always to activate the fans, the model will converge to a wrong policy leading to material damage. This is due to the impact of the action on the

FIGURE 3.13 – The states of our environment with time and state dependence transitions. The environment has three states with associated rewards.



transition probabilities, which in the real world is a consequence of inertia in buildings. In practice, algorithms may give good policies that do not converge to optimal ones. With some luck, practitioners may encounter policies with good performance and stop learning at that point. The learning seems unsatisfactory in those conditions. Indeed the learning will not be reproducible. We lose the possibility of continuous learning, one possibility offered by RL. In our continuous setting, there are latent variables z due to the wall's ability to stack energy. The impact of state-action pairs is diluted in the future through the unobserved states. If the impact is significant, convergence becomes hard to achieve. Because the small policy changes involve more changes in trajectories, their consequences can be dramatic if the reward signal admits some sparsity because it leads to high variance in the return estimation.

From Figure 3.14 we can understand the behaviour of the unobserved nodes. The behaviour of the unobserved node influences the air temperature, but if the policy is fixed, the impact is considered in the transition function. The environment is partially observed, but if we know previous iterations, the environment can be entirely described. Alternatively, the environment can be considered stochastic as the transition function is stochastic, which can be associated with policy uncertainty. We know the system is time-dependent and has hidden variables. To mitigate the latent effect would use memory or latent models. However, the poor understanding of neural networks coupled with RL needs a better understanding of how they deal with information and what matters. As the latent state influences the transition, adding the states and actions of previous steps integrates the latent effect into the learning. The idea is to let the agent implicitly estimate the latent state through the variations of states and actions. Assuming previous steps, we deduce that estimating the heat transfer with the nodes in contact with the air is possible from Equation (3.1), reducing the uncertainty about the unobserved variables.

$$\sum_{i \in S_{\text{int}}} \frac{T_i}{2R_{\text{cv}}R_i} = \rho_j C_j V_j \frac{\partial T_{air}}{\partial t} + |S_{int}| \frac{T_{air}}{R_{cv}} - q_m.C_m \left(T_{ext} - T_{air}\right).$$

Another possibility is to consider Monte Carlo estimation methods of the return by sampling whole trajectories. The current policy is evaluated on the whole trajectory, but this

approach has poor scaling capacity, as it needs several trajectories to estimate the return of a policy. With a stochastic and high dimensional environment, the learning will be slow or too much approximated.

3.5 Results and Discussion

We trained efficient DRL models that can outperform baseline models in some conditions:

Energy consumption for the policy considered	
Baseline	249.26
Baseline delta T	221.12
DRL	205.12

TABLE 3.1 – The DRL model performs better than the baseline optimized for this problem. We have a gain of 17% for the plain baseline and 7% for the optimised baseline. None of the models gets out of the authorized range.

When learning a model in a deterministic environment, we observe that the model has random behaviour outside the learning domain and cannot perform in slightly different settings.

In real-world applications, the external temperature is more inconsistent. The sinus function needs to be a proper way of modelling it. The outside temperature can be very noisy and uncertain. The trained models must consider the uncertainty when estimating the best actions. The policies are subject to the environment's stochasticity. This consideration makes usual optimisations far more complicated than in the previous case. The main concern is how DRL scales to this complexity. The stochasticity of the environment influences the probability of getting to a state knowing the current state and action $p(s_{t+1}|s_t, a_t)$ if the confidence of the transition of the environment is low, a lot of Data is needed to compensate. It motivates to favorise off-policy methods as on-policy requires to sample with the current policy.

We couple the nodal model with actual data presented in Figure 3.16. For more uncertainty and to increase the Data, use two supplementary variables following the normal distribution drawn before each simulation. One variable is added to the temperature, and the other multiplies it. When doing such, we add the time in the variables for the agents. The time is encoded with a sinus and cosines to keep the cyclic aspect. The purpose is to help the agent to deal with day and night cycles to anticipate heat peaks.

3.5.1 Discussion

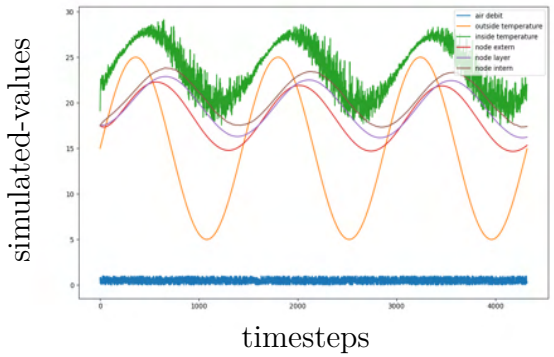
The results obtained in this section give the motivation to use RL to optimise physical systems, even if the studied environment was purely deterministic. At first, our primary goal was to obtain results close to the baseline as they are optimised on this particular environment using the fact that it is deterministic. The Learned model can outperform the baseline for the reason that the baseline parametric model with linear form. Neural networks learned are more flexible. But even if good results can be obtained, the learning of agents able to control

fan systems is harder than it seems. The stability of learning is not granted and it is a real problem for applications
When the outside temperature is stochastic, the model should consider the risk. The learning is more complex and the inertia is more difficult to deal with. More complex models using memory or latent might be needed to find good policies. When using simulated cyclic data, the model has no issue estimating the risk of action, but the complexity is much higher if the environment is stochastic. In RL and DRL choosing the hyperparameters for solving an environment is a pain as the optimisation of a complex scenario need a huge computation capacity due to the number of possible configuration, the time of running a simulation and the instability of RL. Starting with an already-tuned parameter for similar environments and algorithms is a good start. Using implementation that the community has recognized is a significant concern to ease the reproduction of the results, as the complexity of the algorithms leads to errors that are challenging to detect.

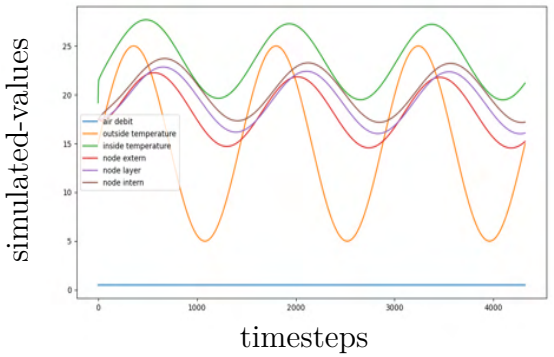
3.5.2 Deploying a model in actual conditions

In actual conditions, deploying an agent will face more complex situations. Simulated environments for physical applications tend to be perfect. But in reality, we know that measured values with sensors can be misleading, the air is not perfectly mixed, and sensors are a proxy for the real state of the system. So when observing values, there are still uncertainties about the real state of the system. Moreover, agents have to deal with delays. When picking an action with the agent, there will be delays between the decision and the impact on the state. It is due to the fact physical limit of the system but also due to the engineering pipeline. So using the step objective will be misleading. Adding n-step methods [86] to the objective is necessary. It considers the impact of an action on the n next steps. The agent will have to deal with unexpected events due to implementation (like coding errors) or external conditions (like a storm). From the agent's perspective, those unrecognised events may lead him to execute random actions. This is why security management must take control in case of incidental behaviour. But one major issue is the data generation process of an agent in actual conditions is far more important. Training agents in RL is time-consuming. Depending on the computer's performance and the environment's difficulty, teaching an agent to behave as expected can take hours or days. In actual conditions, the data generation process is even longer. Still, progress is realized in improving off-policy learning, and some strategies may be employed to train agents more efficiently. This is a good reason to use off-policy RL. If we have data generated previously by some policies, we can start learning an agent. Also, the trained agent does not need to be the one producing the action. Yet it is preferable to have an agent close to the target agent to learn the behaviour on which it will act. The second advantage is that the learning steps are asynchronous from the data generation, so the learning speed is not entirely dependent on the exploration. From our perspective to deploy RL, the main considerations are:

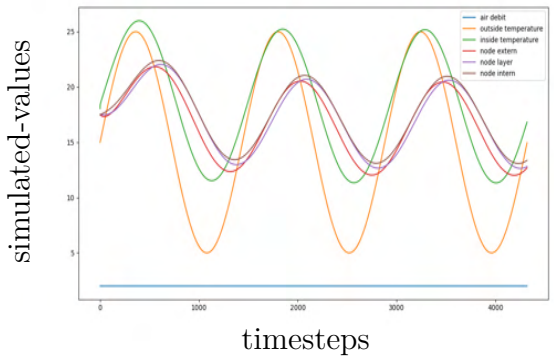
— Using off-policy RL.
— Ensuring a security system to take the lead in case of an issue.
— Using a proper objective like n-step methods.
— Maintaining a very clean engineering pipeline.



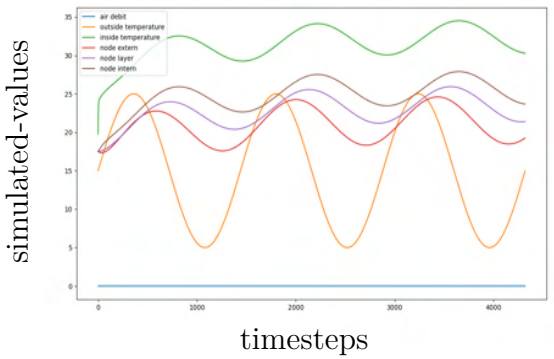
(a) action noise



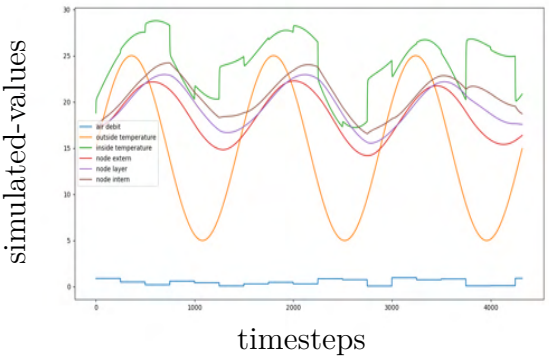
(b) No noise medium debit



(c) No noise high debit

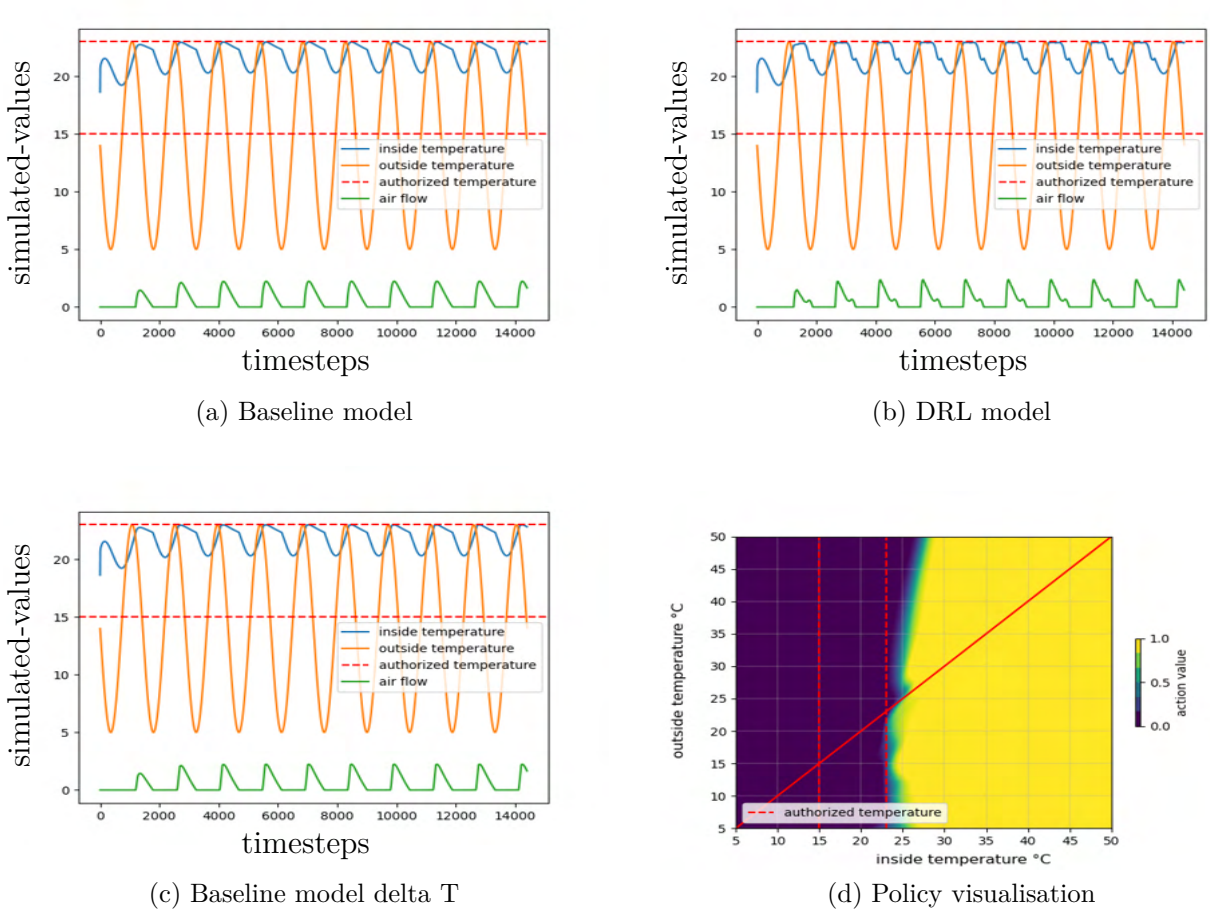


(d) No noise low debit



(e) Time coherent noise

FIGURE 3.14 – Comparison of the system's action profile shows that coherent schemes of exploration are necessary to explore the intern states of the system correctly. High-frequency action variations are not perceptible on the intern nodes. In the left top figure, the behaviour of the intern nodes is very close to the one in the right top figure, while the last figure shows variation in the intern nodes.



(a) Baseline model

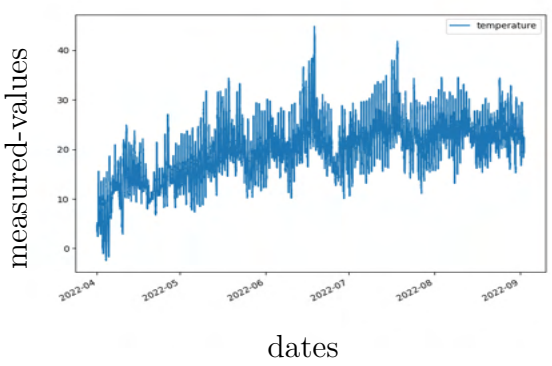
(b) DRL model

(c) Baseline model delta T

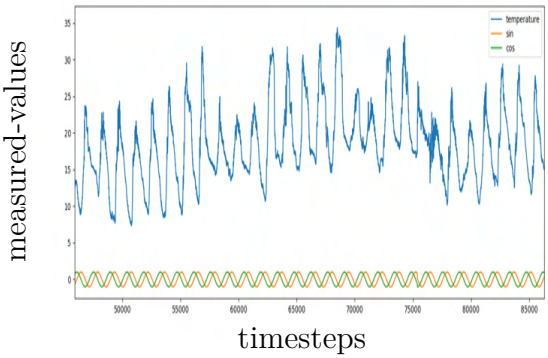
(d) Policy visualisation

FIGURE 3.15 – We compare the baseline models to the policies found with DRL. The airflow was scaled by a factor of ten. The last figure represents the policy using colour mapping. From the outside and inside temperature to the action chosen. From this figure, we observe when the model would choose to ventilate. The results for the consumption are in Table 3.1

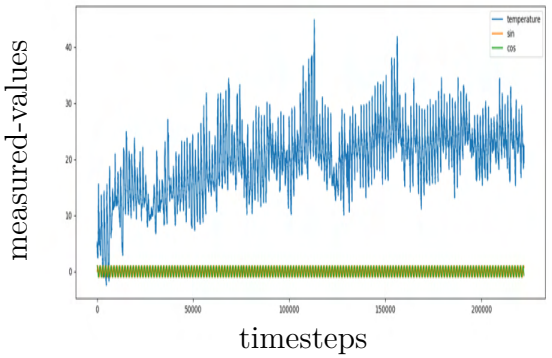
Real Data temperature at Mimizan in the summer of 2021



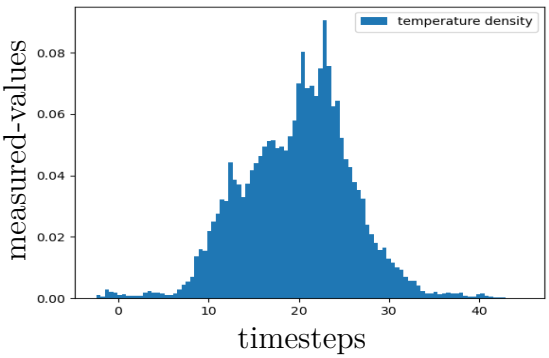
(a) The temperature between June and September 2021



(b) Temperature sequence with sinus and cosines of time in seconds to model time as a loop on a sample



(c) Temperature sequence with sinus and cosines of time in seconds to model time as a loop



(d) Density of temperatures in the dataset

FIGURE 3.16 – Data used to add uncertainty and get closer from actual conditions to the model.

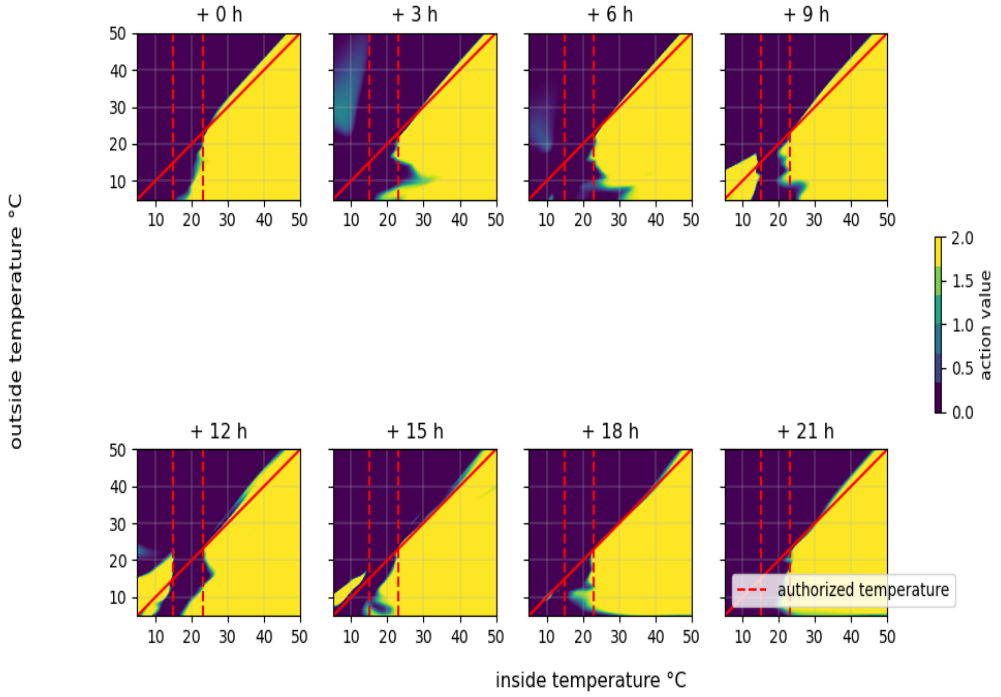


FIGURE 3.17 – In this figure, we can observe how time impacts the policy. It shows the model's ability to adapt the policy for the moment of the day and anticipate.

Chapitre 4

Conclusion and Perspective

4.1 Conclusion

Artificial Intelligence has much potential for improving the functioning of Data Centers and other industrial applications. Reinforcement learning seems the ideal candidate for artificial intelligence tools to design autonomous agents for providing optimal decisions in an operating environment. Reinforcement learning was first designed as a tabular method, with the limitation of dealing with discrete states and actions. But with Deep Learning and function approximation, Reinforcement Learning gains a lot of potential. It can tackle complex environments. It can deal with continuous states and actions, and having variables such as temperature or hygrometry is not an issue anymore. Moreover, Deep Learning is known for scaling relatively well with the dimensions of the input/output, making it an ideal candidate for dealing with structures that can scale up. The results presented by the research community are very promising. Many papers are improving the theory and showing results in applications. However, we must keep in sight the actual limit of the current methods. Even if Deep Learning is appealing, its theoretical foundation is poor compared to traditional methods like linear models. In Reinforcement Learning with function approximation, there is also the risk of divergence and non-reproducibility. So combining the technology increases the risks during learning. Sutton, in his book [86] is talking about "The Deadly Triad", composed of Function approximation, Bootstrapping and Off-policy training. But Function approximation is necessary for dealing with complex continuous environments. Bootstrapping diminishes the computation a lot. Off-policy is a way of learning from experiences that are not generated from the policy, which is learning. It is a major asset for safe learning. When evaluating Reinforcement Learning, the time needed to train models makes the comparison fastidious, and the number of hyperparameters is too big to find the fittest collection. It is challenging to evaluate the impact of the hyperparameters individually. It is common to choose them by trying different values until an acceptable value is found. The hyperparameters are often chosen as typical values in the field. This is an issue regarding comparing agents. The comparison cannot be valid if one is better tuned than the other. If the results are not reproducible, the comparison needs many runs, making it hard to conduct. The current state of the art of Deep Reinforcement Learning compels practitioners to implement a lot of code to produce models. The community is working on creating reliable packages to assess algorithms and techniques and experiment and develop environments associated with use cases.

Solving specific tasks also brings challenges. Many environments are not perfect Markovian Decision Processes. In actual conditions (not in simulations), most environments are partially observed, and some do not satisfy the Markov property or the stationarity of the chain. Even if, in many cases, it is possible to trick to find a setup in which the Markov property is enough by, for example, integrating several previous steps in the state. Those methods are not ideal, and the research on approximate methods is ongoing. Even if results start to be shown in simulations, there are more possibilities for demonstration than actual application successes. Until more understanding of convergence and robustness is furnished, these methods won't be able to tackle the current environment safely. The dependency on initialisation and randomness makes them currently not practical in most applications. Moreover, when stepping into actual conditions, the data face new issues, such as the chaotic weather data, leading to high variation and stochasticity in system state transitions. The observed data undergo measurement uncertainties, the physical behaviours are more complex than what we simulated, and gazes obey more complex theories, such as the mechanics of fluids. Fortunately, most IT material currently has a wider temperature range than what we could experiment with.

4.2 Perspective

Reinforcement learning is an ongoing research subject, and approximate methods, specifically when associated with deep learning methods, need to be more understood and explored to be used efficiently. The difficulties and the solutions are not well assessed. Many frameworks must be assessed and answer some difficulties encountered when solving environments with Deep Reinforcement Learning. Some of these frameworks' impact may leverage Reinforcement Learning's expectations for high-dimensional environments with complex transitions and observations. Among those technologies, we have variational methods for discovering latent space or encoding high dimensional spaces. Recurrent modelling works with environments that do not satisfy the Markov property due to response delays. However, the complexity of these models is a burden added to the reinforcement learning complexity. Understanding the issue inherent in the framework is necessary before introducing more complex methods.

4.2.1 VRAE/LSTM

To perform better in environments with no Markovian properties on the state transitions, VRAE [26] and recurrent models [19] are popular deep-learning models with high potential [10], [38], [52]. The issue is their complexity and need for large data consumption to perform. LSTM use memory to improve the prediction. Previous information is consumed through a memory path inside the network.

4.2.2 Bayesian Methods

Bayesian methods have a lot of perspectives [98], especially to deal with the understanding and transparency of deep learning models. Bayesian methods seem a good choice to counterbalance the over-parametrization and black box drawbacks. Normalizing

flow [56] for Bayesian neural networks is an exciting perspective to improve our actor as it has been shown to improve the prediction ability and uncertainty measure of Bayesian neural networks. Moreover, it is a key to new strategies such as empirical methods for the prior.

4.2.3 Offline Methods

Developing offline (which has no access to the environment and uses only a dataset of interactions) and off-policy (the exploration policy and learning policies are different) methods is a requirement to deploy agents in more concrete situations. Offline configuration returns to more traditional supervised learning, as the environment mechanics are drawn from data. This configuration is more complex as the learning agent cannot try on the environment and must be satisfied with counterfactual experiments [54]. The current limitations are very hard to overcome, but successful offline algorithms would be a great achievement for many applications.

4.2.4 Non-Parametric Recursive Regression

Combining deep learning with a more traditional stochastic learning algorithm is also a promising perspective. When using deep learning approximations, we sacrifice the convergence insurance and control over learning compared to techniques with more theoretical backgrounds. Non-parametric regression is a good compromise with non-parametric regression flexibility and its theoretical knowledge [82], [83], [81]. Recursive non-parametric regression has the advantage of having the possibility to be learned online, which has the same spirit as reinforcement learning. Using this kind of model for critic representation may help solidify the ground of reinforcement learning. The distributional representation of the framework is a tool for tackling the counterfactual issue of off-policy learning.

Bibliographie

[1] Cisco Annual Internet Report - Cisco Annual Internet Report (2018–2023) White Paper.

[2] Cooling buildings sustainably in Europe: exploring the links between climate change mitigation and adaptation, and their social impacts — European Environment Agency.

[3] Bayesian Inference and Decision Theory. In Jayanta K. Ghosh, Mohan Delampady, and Tapas Samanta, editors, *An Introduction to Bayesian Analysis: Theory and Methods*, Springer Texts in Statistics, pages 29–63. Springer, New York, NY, 2006.

[4] Zakia Afroz, GM Shafiullah, Tania Urmee, and Gary Higgins. Modeling techniques used in building HVAC control systems: A review. *Renewable and Sustainable Energy Reviews*, 83:64–84, March 2018.

[5] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory Aware Synapses: Learning what (not) to forget, October 2018. Issue: arXiv:1711.09601 arXiv:1711.09601 [cs, stat].

[6] Sheena Angra and Sachin Ahuja. Machine learning and its applications: A review. In *2017 International Conference on Big Data Analytics and Computational Intelligence (ICBDAC)*, pages 57–60, March 2017.

[7] Verena M. Barthelmes, Yeonsook Heo, Valentina Fabi, and Stefano P. Corgnati. Exploration of the Bayesian Network framework for modelling window control behaviour. *Building and Environment*, 126:318–330, December 2017.

[8] Michael Beetz. Knowledge Representation and Reasoning. pages 413–432. May 2022.

[9] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight Uncertainty in Neural Network. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 1613–1622. PMLR, June 2015. ISSN: 1938-7228.

[10] Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron Courville, and Yoshua Bengio. A Recurrent Latent Variable Model for Sequential Data. *arXiv:1506.02216 [cs]*, April 2016. arXiv: 1506.02216.

[11] Pierre Clavier, Stéphanie Allassonière, and Erwan Le Pennec. Robust Reinforcement Learning with Distributional Risk-averse formulation, June 2022. Issue: arXiv:2206.06841 arXiv:2206.06841 [cs, math, stat].

[12] Cédric Colas, Olivier Sigaud, and Pierre-Yves Oudeyer. GEP-PG: Decoupling Exploration and Exploitation in Deep Reinforcement Learning Algorithms. In *Proceedings of the 35th International Conference on Machine Learning*, pages 1039–1048. PMLR, July 2018. ISSN: 2640-3498.

[13] Carlos Cruz, Juan R. González, and David A. Pelta. Optimization in dynamic environments: a survey on problems, methods and measures. *Soft Computing*, 15(7):1427–1448, July 2011.

[14] Miyuru Dayarathna, Yonggang Wen, and Rui Fan. Data Center Energy Consumption Modeling: A Survey. *IEEE Communications Surveys & Tutorials*, 18(1):732–794, 2016. Conference Name: IEEE Communications Surveys & Tutorials.

[15] Jeffrey Dean, Greg S. Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Quoc V. Le, Mark Z. Mao, Marc'Aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, and Andrew Y. Ng. Large scale distributed deep networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pages 1223–1231, Red Hook, NY, USA, 2012. Curran Associates Inc.

[16] Jonas Degrave, Federico Felici, Jonas Buchli, Michael Neunert, Brendan Tracey, Francesco Carpanese, Timo Ewalds, Roland Hafner, Abbas Abdolmaleki, Diego de las Casas, Craig Donner, Leslie Fritz, Cristian Galperti, Andrea Huber, James Keeling, Maria Tsimpoukelli, Jackie Kay, Antoine Merle, Jean-Marc Moret, Seb Noury, Federico Pesamosca, David Pfau, Olivier Sauter, Cristian Sommariva, Stefano Coda, Basil Duval, Ambrogio Fasoli, Pushmeet Kohli, Koray Kavukcuoglu, Demis Hassabis, and Martin Riedmiller. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature*, 602(7897):414–419, February 2022. Number: 7897 Publisher: Nature Publishing Group.

[17] Thomas Degris, Martha White, and Richard S. Sutton. Off-Policy Actor-Critic, June 2013. arXiv:1205.4839 [cs].

[18] Xianzhong Ding, Wan Du, and Alberto Cerpa. OCTOPUS: Deep Reinforcement Learning for Holistic Smart Building Control. In *Proceedings of the 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, BuildSys '19, pages 326–335, New York, NY, USA, November 2019. Association for Computing Machinery.

[19] Andreas Doerr, Christian Daniel, Martin Schiegg, Duy Nguyen-Tuong, Stefan Schaal, Marc Toussaint, and Sebastian Trimpe. Probabilistic Recurrent State-Space Models. *arXiv:1801.10395 [stat]*, February 2018. arXiv: 1801.10395.

[20] Brian Dougherty, Jules White, and Douglas C. Schmidt. Model-driven auto-scaling of green cloud computing infrastructure. *Future Generation Computer Systems*, 28(2):371–378, February 2012.

[21] Carlo D'Eramo, Andrea Cini, Alessandro Nuara, Matteo Pirotta, Cesare Alippi, Jan Peters, and Marcello Restelli. Gaussian Approximation for Bias Reduction in Q-Learning. page 51.

[22] Chance Elliott, Vipin Vijayakumar, Wesley Zink, and Richard Hansen. National Instruments LabVIEW: A Programming Environment for Laboratory Automation and Measurement. *JALA: Journal of the Association for Laboratory Automation*, 12(1):17–24, February 2007. Publisher: SAGE Publications Inc.

[23] ETSI. Environmental Engineering (EE) ; Environmental conditions and environmental tests for telecommunications equipment ; Part 1-3: Classification of environmental conditions ; Stationary use at weatherprotected locations.

[24] ETSI. Environmental Engineering (EE) ; European telecommunications standard for equipment practice ; Thermal Management Guidance for equipment and its deployment.

[25] ETSI. ETSI GS OEU 001 V1.2.4 (2014-10) Operational energy Efficiency for Users (OEU) ; Global KPIs for Data Centres.

[26] Otto Fabius and Joost R. van Amersfoort. Variational Recurrent Auto-Encoders. *arXiv:1412.6581 [cs, stat]*, June 2015. arXiv: 1412.6581.

[27] Haneef A Fatmi and RW Young. A definition of intelligence. *Nature*, 228:97–97, 1970. Publisher: Springer.

[28] Robert Ferret, Françoise Berthoud, and Laurent Lefèvre. Consommation energetique des datacentres: quand la politique Europeenne s'en mêle. page 9, 2011.

[29] H. Flandorfer, F. Gehringer, and E. Hayer. Individual solutions for control and data acquisition with the PC. *Thermochimica Acta*, 382(1):77–87, January 2002.

[30] Karl Friston, Jérémie Mattout, Nelson Trujillo-Barreto, John Ashburner, and Will Penny. Variational free energy and the Laplace approximation. *NeuroImage*, 34(1):220–234, January 2007.

[31] Scott Fujimoto, Herke Hoof, and David Meger. Addressing Function Approximation Error in Actor-Critic Methods. In *Proceedings of the 35th International Conference on Machine Learning*, pages 1587–1596. PMLR, July 2018. ISSN: 2640-3498.

[32] Scott Fujimoto, David Meger, and Doina Precup. Off-Policy Deep Reinforcement Learning without Exploration. In *Proceedings of the 36th International Conference on Machine Learning*, pages 2052–2062. PMLR, May 2019. ISSN: 2640-3498.

[33] Mohammad Ghavamzadeh, Shie Mannor, Joelle Pineau, and Aviv Tamar. Bayesian Reinforcement Learning: A Survey. *Foundations and Trends® in Machine Learning*, 8(5-6):359–483, 2015. Number: 5-6 arXiv: 1609.04436.

[34] Alex Graves. Practical Variational Inference for Neural Networks. In *Advances in Neural Information Processing Systems*, volume 24. Curran Associates, Inc., 2011.

[35] Steve Greenberg and William Tschudi. Self Benchmarking Guide for Data Center Energy Performance Version 1.0. page 33.

[36] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. *arXiv:1801.01290 [cs, stat]*, August 2018. arXiv: 1801.01290.

[37] Ali Habibi Khalaj and Saman K. Halgamuge. A Review on efficient thermal management of air- and liquid-cooled data centers: From chip to the cooling system. *Applied Energy*, 205:1165–1188, November 2017.

[38] Dongqi Han, Jun Tani, and Kenji Doya. VARIATIONAL RECURRENT MODELS FOR SOLVING PARTIALLY OBSERVABLE CONTROL TASKS. page 19, 2020.

[39] Vinay Hanumaiah and Sahika Genc. Distributed Multi-Agent Deep Reinforcement Learning Framework for Whole-building HVAC Control, October 2021. Issue: arXiv:2110.13450 arXiv:2110.13450 [cs, eess].

[40] Joshua Hare. Dealing with Sparse Rewards in Reinforcement Learning, November 2019. arXiv:1910.09281 [cs, stat].

[41] Nicolas Heess, Gregory Wayne, David Silver, Timothy Lillicrap, Tom Erez, and Yuval Tassa. Learning Continuous Control Policies by Stochastic Value Gradients. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[42] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep Reinforcement Learning that Matters. *arXiv:1709.06560 [cs, stat]*, January 2019. arXiv: 1709.06560.

[43] Geoffrey E. Hinton and Drew van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory*, COLT '93, pages 5–13, New York, NY, USA, 1993. Association for Computing Machinery.

[44] Zhewei Huang, Shuchang Zhou, BoEr Zhuang, and Xinyu Zhou. Learning to Run with Actor-Critic Ensemble. *arXiv:1712.08987 [cs]*, December 2017. arXiv: 1712.08987.

[45] Tommi S. Jaakkola and Michael I. Jordan. Bayesian parameter estimation via variational methods. *Statistics and Computing*, 10(1):25–37, January 2000.

[46] Stanislaw Jastrzebski, Maciej Szymczak, Stanislav Fort, Devansh Arpit, Jacek Tabor, Kyunghyun Cho, and Krzysztof Geras. The Break-Even Point on Optimization Trajectories of Deep Neural Networks. April 2020.

[47] Laurent Valentin Jospin, Wray Buntine, Farid Boussaid, Hamid Laga, and Mohammed Bennamoun. Hands-on Bayesian Neural Networks – a Tutorial for Deep Learning Users. *arXiv:2007.06823 [cs, stat]*, September 2021. arXiv: 2007.06823.

[48] Shauharda Khadka, Somdeb Majumdar, Tarek Nassar, Zach Dwiel, Evren Tumer, Santiago Miret, Yinyin Liu, and Kagan Tumer. Collaborative Evolutionary Reinforcement Learning. In *Proceedings of the 36th International Conference on Machine Learning*, pages 3341–3350. PMLR, May 2019.

[49] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. *arXiv:1312.6114 [cs, stat]*, May 2014. arXiv: 1312.6114.

[50] Arsenii Kuznetsov, Pavel Shvechikov, Alexander Grishin, and Dmitry Vetrov. Controlling Overestimation Bias with Truncated Mixture of Continuous Distributional Quantile Critics. In *Proceedings of the 37th International Conference on Machine Learning*, pages 5556–5566. PMLR, November 2020. ISSN: 2640-3498.

[51] Nevena Lazic, Craig Boutilier, Tyler Lu, Eehern Wong, Binz Roy, MK Ryu, and Greg Imwalle. Data center cooling using model-predictive control. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[52] Alex X. Lee, Anusha Nagabandi, Pieter Abbeel, and Sergey Levine. Stochastic Latent Actor-Critic: Deep Reinforcement Learning with a Latent Variable Model. *arXiv:1907.00953 [cs, stat]*, October 2020. arXiv: 1907.00953.

[53] Joel Lehman, Jay Chen, Jeff Clune, and Kenneth O. Stanley. Safe Mutations for Deep and Recurrent Neural Networks through Output Gradients, May 2018. Issue: arXiv:1712.06563 arXiv:1712.06563 [cs].

[54] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems. *arXiv:2005.01643 [cs, stat]*, November 2020. arXiv: 2005.01643.

[55] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv:1509.02971 [cs, stat]*, July 2019. arXiv: 1509.02971.

[56] Christos Louizos and Max Welling. Multiplicative Normalizing Flows for Variational Bayesian Neural Networks, June 2017. Issue: arXiv:1703.01961 arXiv:1703.01961 [cs, stat].

[57] Stephan Mandt, Matthew D. Hoffman, and David M. Blei. Stochastic Gradient Descent as Approximate Bayesian Inference. *arXiv:1704.04289 [cs, stat]*, January 2018. arXiv: 1704.04289.

[58] Horia Mania, Aurelia Guy, and Benjamin Recht. Simple random search of static linear policies is competitive for reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[59] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous Methods for Deep Reinforcement Learning. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 1928–1937. PMLR, June 2016. ISSN: 1938-7228.

[60] Thomas M. Moerland, Joost Broekens, Aske Plaat, and Catholijn M. Jonker. Model-based Reinforcement Learning: A Survey, March 2022. arXiv:2006.16712 [cs, stat].

[61] Siddharth Mysore, Bassel Mabsout, Renato Mancuso, and Kate Saenko. Regularizing Action Policies for Smooth Control with Reinforcement Learning, May 2021. arXiv:2012.06644 [cs, eess].

[62] Olli Mämmelä, Mikko Majanen, Robert Basmadjian, Hermann De Meer, André Giesler, and Willi Homberg. Energy-aware job scheduler for high-performance computing. *Computer Science - Research and Development*, 27(4):265–275, November 2012.

[63] Andrei Paleyes, Raoul-Gabriel Urma, and Neil D. Lawrence. Challenges in Deploying Machine Learning: a Survey of Case Studies. *ACM Computing Surveys*, 2022. Just Accepted.

[64] Pingbo Pan, Siddharth Swaroop, Alexander Immer, Runa Eschenhagen, Richard E. Turner, and Mohammad Emtiyaz Khan. Continual Deep Learning by Functional Regularisation of Memorable Past, January 2021. Issue: arXiv:2004.14070 arXiv:2004.14070 [cs, stat].

[65] Michael K. Patterson. The effect of data center temperature on energy efficiency. In *2008 11th Intersociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems*, pages 1167–1174, May 2008. ISSN: 1087-9870.

[66] P. Petsagkourakis, I. O. Sandoval, E. Bradford, D. Zhang, and E. A. del Rio-Chanona. Constrained Reinforcement Learning for Dynamic Optimization under Uncertainty. *IFAC-PapersOnLine*, 53(2):11264–11270, January 2020.

[67] C. L. Philip Chen and Chun-Yang Zhang. Data-intensive applications, challenges, techniques and technologies: A survey on Big Data. *Information Sciences*, 275:314–347, August 2014.

[68] Matthias Plappert, Rein Houthooft, Prafulla Dhariwal, Szymon Sidor, Richard Y. Chen, Xi Chen, Tamim Asfour, Pieter Abbeel, and Marcin Andrychowicz. Parameter Space Noise for Exploration. March 2022.

[69] Rastin Pries, Michael Jarschel, Daniel Schlosser, Michael Klopf, and Phuoc Tran-Gia. Power Consumption Analysis of Data Center Architectures. In Joel J. P. C. Rodrigues, Liang Zhou, Min Chen, and Aravind Kailas, editors, *Green Communications and Networking*, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pages 114–124, Berlin, Heidelberg, 2012. Springer.

[70] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-Baselines3: Reliable Reinforcement Learning Implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

[71] Antonin Raffin, Jens Kober, and Freek Stulp. Smooth Exploration for Robotic Reinforcement Learning. In *Proceedings of the 5th Conference on Robot Learning*, pages 1634–1644. PMLR, January 2022. ISSN: 2640-3498.

[72] Jacques Rancière. *Le maître ignorant: cinq leçons sur l'émancipation intellectuelle*. Fayard, 2014.

[73] Huigui Rong, Haomin Zhang, Sheng Xiao, Canbing Li, and Chunhua Hu. Optimizing energy consumption for data centers. *Renewable and Sustainable Energy Reviews*, 58:674–691, May 2016.

[74] Thomas Rückstieß, Martin Felder, and Jürgen Schmidhuber. State-Dependent Exploration for Policy Gradient Methods. In Walter Daelemans, Bart Goethals, and Katharina Morik, editors, *Machine Learning and Knowledge Discovery in Databases*, Lecture Notes in Computer Science, pages 234–249, Berlin, Heidelberg, 2008. Springer.

[75] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust Region Policy Optimization. In *Proceedings of the 32nd International Conference on Machine Learning*, pages 1889–1897. PMLR, June 2015. ISSN: 1938-7228.

[76] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal Policy Optimization Algorithms. *arXiv:1707.06347 [cs]*, August 2017. arXiv: 1707.06347.

[77] Frank Sehnke, Christian Osendorfer, Thomas Rückstieß, Alex Graves, Jan Peters, and Jürgen Schmidhuber. Parameter-exploring policy gradients. *Neural Networks*, 23(4):551–559, May 2010. Number: 4.

[78] Alexander Shapiro. Monte Carlo Sampling Methods. In *Handbooks in Operations Research and Management Science*, volume 10 of *Stochastic Programming*, pages 353–425. Elsevier, January 2003.

[79] Ajay Shrestha and Ausif Mahmood. Review of Deep Learning Algorithms and Architectures. *IEEE Access*, 7:53040–53065, 2019. Conference Name: IEEE Access.

[80] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic Policy Gradient Algorithms. In *Proceedings of the 31st International Conference on Machine Learning*, pages 387–395. PMLR, January 2014. ISSN: 1938-7228.

[81] Yousri Slaoui. Wild bootstrap bandwidth selection of recursive nonparametric relative regression for independent functional data. *Journal of Multivariate Analysis*, 173(C):494–511, 2019.

[82] Yousri Slaoui. Recursive nonparametric regression estimation for dependent strong mixing functional data. *Statistical Inference for Stochastic Processes*, 23(3):665–697, 2020.

[83] Yousri Slaoui. Recursive Nonparametric Regression Estimation for Independent Functional Data. *Statistica Sinica*, 30(1):417–437, 2020.

[84] Elena Smirnova, Elvis Dohmatob, and Jérémie Mary. Distributionally Robust Reinforcement Learning, June 2019. Issue: arXiv:1902.08708 arXiv:1902.08708 [cs, stat].

[85] Felipe Petroski Such, Vashisht Madhavan, Edoardo Conti, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. Deep Neuroevolution: Genetic Algorithms Are a Competitive Alternative for Training Deep Neural Networks for Reinforcement Learning, April 2018. Issue: arXiv:1712.06567 arXiv:1712.06567 [cs].

[86] Richard S. Sutton and Andrew Barto. *Reinforcement learning: an introduction*. Adaptive computation and machine learning. The MIT Press, Cambridge, Massachusetts London, England, second edition edition, 2018.

[87] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy Gradient Methods for Reinforcement Learning with Function Approximation. In *Advances in Neural Information Processing Systems*, volume 12. MIT Press, 1999.

[88] Fei Tao, Bin Xiao, Qinglin Qi, Jiangfeng Cheng, and Ping Ji. Digital twin modeling. *Journal of Manufacturing Systems*, 64:372–389, July 2022.

[89] Philip S Thomas. Bias in Natural Actor-Critic Algorithms. page 8.

[90] Sebastian Thrun and Anton Schwartz. Issues in Using Function Approximation for Reinforcement Learning. page 9.

[91] George Tucker, Surya Bhupatiraju, Shixiang Gu, Richard E Turner, Zoubin Ghahramani, and Sergey Levine. The Mirage of Action-Dependent Baselines in Reinforcement Learning. page 10.

[92] Duc Van Le, Yingbo Liu, Rongrong Wang, Rui Tan, Yew-Wah Wong, and Yonggang Wen. Control of Air Free-Cooled Data Centers in Tropics via Deep Reinforcement Learning. In *Proceedings of the 6th ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*, pages 306–315, New York NY USA, November 2019. ACM.

[93] Stijn Verbeke and Amaryllis Audenaert. Thermal inertia in buildings: A review of impacts across climate and building use. *Renewable and Sustainable Energy Reviews*, 82:2300–2318, February 2018.

[94] Pei Wang. On Defining Artificial Intelligence. *Journal of Artificial General Intelligence*, 10(2):1–37, January 2019.

[95] Zeyu Wang and Ravi S. Srinivasan. A review of artificial intelligence based building energy use prediction: Contrasting the capabilities of single and ensemble prediction models. *Renewable and Sustainable Energy Reviews*, 75:796–808, August 2017.

[96] Mathieu Wauters and Mario Vanhoucke. Support Vector Machine Regression for project control forecasting. *Automation in Construction*, 47:92–106, November 2014.

[97] Beth Whitehead, Deborah Andrews, Amip Shah, and Graeme Maidment. Assessing the environmental impact of data centres part 1: Background, energy use and metrics. *Building and Environment*, 82:151–159, December 2014.

[98] Andrew Gordon Wilson and Pavel Izmailov. Bayesian Deep Learning and a Probabilistic Perspective of Generalization. page 12.

[99] Kaishu Xia, Christopher Sacco, Max Kirkpatrick, Clint Saidy, Lam Nguyen, Anil Kircaliali, and Ramy Harik. A digital twin to train deep reinforcement learning agent for smart manufacturing plants: Environment, interfaces and intelligence. *Journal of Manufacturing Systems*, 58:210–230, January 2021.

[100] Sungkap Yeo and Hsien-Hsin S. Lee. Peeling the Power Onion of Data Centers. In Yogendra Joshi and Pramod Kumar, editors, *Energy Efficient Thermal Management of Data Centers*, pages 137–168. Springer US, Boston, MA, 2012.

[101] Chiyuan Zhang, Oriol Vinyals, Remi Munos, and Samy Bengio. A Study on Overfitting in Deep Reinforcement Learning. *arXiv:1804.06893 [cs, stat]*, April 2018. arXiv: 1804.06893.

[102] Hainan Zhang, Shuangquan Shao, Hongbo Xu, Huiming Zou, and Changqing Tian. Free cooling of data centers: A review. *Renewable and Sustainable Energy Reviews*, 35:171–182, July 2014.

[103] xiaojing Zhang, T. Lindberg, N. Xiong, V. Vyatkin, and A. Mousavi. Cooling Energy Consumption Investigation of Data Center IT Room with Vertical Placed Server. *Energy Procedia*, 105:2047–2052, May 2017.

[104] Yijie Zhang and Herke Van Hoof. Deep Coherent Exploration for Continuous Control. In *Proceedings of the 38th International Conference on Machine Learning*, pages 12567–12577. PMLR, July 2021. ISSN: 2640-3498.

[105] Rongliang Zhou, Zhikui Wang, Alan McReynolds, Cullen E. Bash, Thomas W. Christian, and Rocky Shih. Optimization and control of cooling microgrids for data centers. In *13th InterSociety Conference on Thermal and Thermomechanical Phenomena in Electronic Systems*, pages 338–343, May 2012. ISSN: 1087-9870.

[106] Xin Zhou, Ruihang Wang, Yonggang Wen, and Rui Tan. Joint IT-Facility Optimization for Green Data Centers via Deep Reinforcement Learning. *IEEE Network*, pages 1–8, 2021. Conference Name: IEEE Network.

Optimisation of Energy Consumption of Data Center using Artificial Intelligence

Résumé

Cette thèse explore la manière dont l'intelligence artificielle peut diminuer l'énergie consommée dans les centres de données. L'intelligence artificielle est un vaste domaine de recherche très populaire. Ce domaine est souvent idéalisé, mais comment les nouveautés de la recherche peuvent-elles être appliquées à des cas d'utilisation réels ? Entre considérations physiques, théories mathématiques et technologies informatiques, cette thèse utilise différents domaines pour aider à déployer et améliorer les technologies récentes afin de répondre aux défis énergétiques actuels.
La première partie de la thèse évalue les problèmes énergétiques et les technologies déployées dans les centres de données et les bâtiments de télécommunications. Les infrastructures d'information et de communication sont de gros consommateurs d'énergie et nécessitent des systèmes de climatisation spécifiques en raison des conditions de travail du matériel informatique. L'optimisation des systèmes de climatisation et de leur consommation est une préoccupation majeure dans la réduction de la consommation d'énergie. La deuxième partie de la thèse explore l'apprentissage statistique et probabiliste pour optimiser la consommation d'énergie. Elle se concentre principalement sur les modèles d'apprentissage par renforcement profond (Deep Reinforcement Learning) pour la prise de décision automatisée basée sur la méthode axée sur les données. L'apprentissage profond est flexible dans la modélisation et peut s'adapter à de nombreux problèmes, ce qui est pratique lorsqu'une méthode doit être généralisée. La dernière partie décrit la mise en œuvre et l'application dans des environnements simples afin de soulever et de traiter les problèmes courants et de discuter de la manière d'adapter le modèle à la réalité. On sait qu'un grand nombre de projets de science des données sont très prometteurs mais ne sont pas mis en œuvre. Ce travail vise à faire un pas en avant dans l'introduction de l'IA dans des environnements sensibles tels que les centres de données.

Mots clés : Intelligence Artificielle, Apprentissage par Renforcement, Apprentissage Automatique, Apprentissage Profond, Statistiques Bayésienne, Energie, Data Centres, Optimisation Dynamique

Abstract

This thesis explores how Artificial Intelligence can diminish the energy consumed in Data centres. Artificial Intelligence is a vast research area that is very popular. This domain is often idealised, but how can the research novelties be applied to actual use cases ? Between physical considerations, mathematics theory and computer science technologies, this thesis employs various fields to help deploy and improve recent technologies to address current energy challenges.
The first part of the thesis assesses the energy issues and the technologies deployed in Data centres and telecommunication buildings. Information and communication infrastructures are massive energy consumers and need specific air conditioning systems due to the working conditions of IT material. Optimising air conditioning systems and their consumption is a primary concern in reducing energy consumption.
The second part of the thesis explores statistical and probabilistic learning to optimise energy consumption. It mainly focuses on Deep Reinforcement Learning models for automated decision-making based on the data-driven method. Deep learning is flexible in modelisation and can suit many problems, which is convenient when a method must be generalised and industrialised.
The last part describes the implementation and application in simple environments to raise and deal with common issues and discuss how to scale the model to reality. It is known that a lot of Data Science projects have great promises but fail to be implemented due to practical difficulties. This work aims to make a step in introducing AI in sensitive environments such as Data Centres.

Keywords: Artificial Intelligence, Reinforcement Learning, Machine Learning, Deep Learning, Bayesian statistic, Energy, Data Centers, Dynamic Optimisation