

Piete Laiti

# **NVIDIA ISAAC LAB: STRUCTURE, CAPABILITIES AND LIMITATIONS FOR MACHINE LEARNING IN ROBOTICS**

NVIDIA Isaac Lab:  
Rakenne, Mahdollisuudet ja Rajoitteet  
Robottiikan Koneoppimisessa

Bachelor's thesis  
Faculty of Engineering and  
Natural Sciences  
Examiner: Roel Pieters  
November 2025

# ABSTRACT

Piete Laiti: NVIDIA Isaac Lab: Structure, Capabilities and Limitation for machine learning in robotics  
Bachelor's thesis  
Tampere University  
Degree Programme in Engineering Sciences, BSc  
November 2025

---

The thesis evaluates how effectively NVIDIA Isaac Lab functions as a simulation and reinforcement learning framework for academic robotics research. The study investigates if the framework can provide an efficient and adaptable environment for developing and training robotic control policies without a strong dependence on physical testing. The work combines a review of machine learning in general, reinforcement learning, and digital twin concepts with practical experimentation conducted in the Isaac Lab environment.

The theoretical background outlines how reinforcement learning enables robots to acquire control behaviour through continuous interaction with their surroundings by guiding them with reward signals, rather than manually designed rules. Digital twin models and sim-to-real transfer are examined as a key mechanism that helps to bridge simulation-based methods with real-world robotics. These ideas establish the foundation for analysing Isaac Lab within the context of modern data-driven robotics.

The practical component analyses the structure of Isaac Lab, which is built on NVIDIA Omniverse and Isaac Sim. Isaac Lab follows a modular manager-based design that separates control, observation, reward and task logic. This structure supports reusability and makes it possible to develop more advanced learning tasks. The Isaac Lift Cube Franka v0 example task was used to evaluate Isaac Lab in practice. The results demonstrated that a Franka Emika Panda robot could be trained to lift a cube in two minutes by running more than 2000 parallel simulation environments on a single computer.

Several challenges were also identified. These were hardware requirements and restrictions, an installation process that could cause difficulties on shared computers, a steep learning curve for new users and the need to keep up with software updates.

The findings suggest that Isaac Lab can serve as a strong tool for reinforcement learning research in robotics when sufficient resources and expertise are available. At this stage, it is useful for both research and teaching, but its practical limitations should be managed before and during wider academic adoption.

Keywords: robotics, machine learning, reinforcement learning, simulation, digital twins, Omniverse, Isaac Sim, Isaac Lab

The originality of this thesis has been verified using the Turnitin Originality Check service.

# USE OF AI IN THESIS

I have utilised AI tools in my thesis:

- No
- Yes

The AI tools utilised in my thesis and their purposes are described below:

Names and versions of AI tools: GPT-4o, GPT-5, Sonnet 4

Purpose of using AI tools: large language models were used to troubleshoot errors during setup, python coding and to help with general usage of Ubuntu.

Sections where AI tools were used: 3.3

I acknowledge that I am fully responsible for the entire content of my thesis, including the parts generated by AI, and accept accountability for any violations of ethical standards in publications.

# PREFACE

I would like to thank Veli-Pekka Pyrhönen, Roel Pieters, Suvi Pellinen and Osama Tasneem for helping me with my thesis. I would like to also send special thanks to anonymous users in NVIDIA Omniverse Discord channel that used their personal time to help me.

Tampere, 25 November 2025

Piete Laiti

# CONTENTS

USE OF AI IN THESIS .....	ii
1.INTRODUCTION .....	1
2.THEORETICAL BACKGROUND.....	2
2.1 Machine learning in robotics.....	2
2.2 Fundamentals of reinforcement learning .....	3
2.3 Digital twin and sim-to-real .....	3
3.RESEARCH METHODOLOGY AND MATERIALS.....	4
3.1 Research approach and methods .....	4
3.2 Overview of NVIDIA Isaac ecosystem .....	5
3.3 System setup and operating environment .....	6
4.FRAMEWORK ANALYSIS .....	8
4.1 System architecture and modular design .....	8
4.2 Reinforcement learning integration.....	9
4.3 RL demonstration: Franka Emika Panda Lift Cube.....	11
5.DISCUSSION.....	14
5.1 Opportunities and strengths of Isaac Lab .....	14
5.2 Limitations and challenges in practice .....	15
5.3 Comparison to intended goals.....	15
5.4 Future prospects and development needs .....	16
5.5 Recommendation for further work .....	17
6.CONCLUSIONS.....	18
REFERENCES.....	20

## LIST OF SYMBOLS AND ABBREVIATIONS

AI	Artificial Intelligence
API	Application Programming Interface
CPU	Central Processing Unit
DL	Deep Learning
DOF	Degree of Freedom
DT	Digital Twin
GPU	Graphics Processing Unit
GB	Gigabyte
LLM	Large Language Model
MDP	Markov Decision Process
ML	Machine Learning
PPO	Proximal Policy Optimization
RAM	Random-access memory
RL	Reinforcement Learning
SB3	Stable Baselines3
TD	Temporal-Difference
USD	Universal Scene Description
VRAM	Video random-access memory

# 1. INTRODUCTION

During the past decade, the rapid growth of automation, artificial intelligence (AI) and robotics has driven researchers to seek new tools and platforms that can make robot learning more efficient and scalable. Simulation environments and machine learning (ML) frameworks have become essential in developing, testing, and optimising robotic control strategies. Reinforcement learning relies on iterative interaction between an agent and its environment [1, pp. 7].

Reinforcement learning (RL) allows robots to learn control strategies through continuous interaction with their environment, guided by reward feedback instead of explicit programming rules [1, pp. 1–2]. Training these systems directly on physical robots is slow, hardware-intensive, and sometimes even risky. This training may require thousands of iterations, which makes real-world applications impractical. For this reason, researchers continue to look more practical approaches to train robots in this way.

This thesis investigates the extent to which NVIDIA Isaac Lab can be used as a tool for reinforcement learning in robotics in academic setting. It is a new simulation framework built on Omniverse and derived from the earlier Orbit project, [2, pp. 3740].

The scope of this thesis is limited to the analysis and evaluation of the Isaac Lab simulation framework for reinforcement learning in robotics. Introduction of key applications will be also included. The study does not involve any physical implementation of robotic systems nor does develop new reinforcement learning algorithms.

The Tampere University library's search service Andor and NVIDIA documentation provided the majority of the references and information used in this thesis. Large language models were used to help troubleshoot problems that arose during coding and Ubuntu setup of the Isaac Lab environment.

## 2. THEORETICAL BACKGROUND

The theoretical underpinnings of the thesis are presented in this chapter, along with the background information required to comprehend machine learning-based robotics techniques. Reinforcement learning as a subfield of machine learning is covered in more detail because of its crucial importance in this work.

### 2.1 Machine learning in robotics

In contemporary robotics, machine learning has emerged as one of the most revolutionary technologies. Robots may adapt to changing settings and unpredictable conditions by learning the patterns and decision-making principles from data and interaction, rather than only explicit programming. In robotics, ML techniques are widely used in perception, motion planning and manipulation. This enables robots to perform tasks that were previously difficult to model accurately with traditional analytical control methods. [3, pp. 314–315]

A subfield of machine learning, deep learning (DL) is based on artificial neural networks. It has played a significant role in improving how robots process sensory and visual data. Earlier attempts to use deep learning for robotic control faced major challenges because non-differentiable dynamics and numerical instability made direct end-to-end control impractical. Levine et al. addressed these limitations by introducing the guided policy search algorithm, which reformulates policy learning as a supervised learning problem. This method makes it possible to train deep convolutional neural networks for visuomotor control in more stable and efficient way, allowing policies to map raw camera observation directly to joint torques. The results demonstrates that such end-to-end training can produce robust and generalisable manipulation skills. This training presents one of the first successful integrations of deep learning and robotic control. [4, pp. 2–4]

As robotic systems continue to integrate with ML-based methods, reinforcement learning has become one of the most prominent approach for achieving autonomous and adaptive robot behaviours. In contrast to supervised learning, RL optimises control policies through interaction with the environment and reward-driven feedback through iterations [1, pp. 2–3].

## 2.2 Fundamentals of reinforcement learning

Reinforcement learning starts with the idea of an agent that interacts continuously with its environment, takes actions and observes the changes that results from those actions. This interaction is what drives the learning process, where the agent attempts to improve its behaviour based on the previous outcomes it receives. The basic cycle consists of action, observation, reward and adjustment and form the foundation for how intelligent systems can improve autonomously. [1, pp. 1–3]

To model this process, RL relies on the framework provided by Markov Decision Processes (MDPs). These processes define how an environment responds to an agent's actions using probabilities, rewards and state transitions [1, pp. 47–49]. A MDP formulation includes the specification of states, available actions, stochastic transitions, a reward signal and a discount factor for future outcomes of the process [1, pp. 55].

A fundamental learning method within this framework is Temporal Difference (TD) learning. This approach allows an agent to improve its value estimates based on partial interaction with the environment, without the need to complete episodes of experience. It achieves this by comparing predictions of future outcomes. Using these results, it updates the agent's internal model accordingly. [1, pp. 118–121]

Usage of TD approach in simulation-based robotics training is essential, as it enables learning even when an agent has limited experience within a given task. TD methods serve as the foundation for many algorithms used in systems such as Isaac Lab.

## 2.3 Digital twin and sim-to-real

A digital twin (DT) is a virtual representation of a physical system developed for monitoring, simulation and control [5 pp. 2405–2407]. It is commonly used in industrial and automation applications. In the context of robotic learning, DT models provide structured high-fidelity environments for developing and testing reinforcement learning policies before deployment to physical hardware. This sim-to-real transfer is supported by aligning parameters and dynamics of the environment between virtual and real systems [6, pp. 1–2]. Domain randomisation and dynamics randomisation approaches can enhance robustness to variations in perception and physical properties during real-world deployment. [7]

## 3. RESEARCH METHODOLOGY AND MATERIALS

The methodological structure of this study and the materials used for the analysis are presented in this chapter. It describes the research approach, presents the structure and functionality of the NVIDIA Isaac Lab framework. It also outlines the system setup within the simulation environment and discusses the reliability and limitations of the findings.

### 3.1 Research approach and methods

Based on the theoretical framework presented in Chapter 2, this section presents the research approach and methods used to evaluate the NVIDIA Isaac Lab simulation environment. The study adopts a qualitative analytical methodology, integrating theoretical review and hands-on experimentation to investigate the framework's structure, functionality and limitations in the context of academic robotics research.

The analysis draws on two primary sources:

1. academic and technical literature, including *The Reinforcement Learning: An Introduction* by Sutton and Barto (2020) and prior work related to Isaac Lab and its predecessor Orbit, *Orbit: A Unified Simulation Framework for Interactive Robot Learning Environments* by Mittal et al. (2023) [1, 2]
2. official documentation of NVIDIA, which provides step-by-step configuration and an overview of the framework's technological architecture, including its simulation and reinforcement learning components. [8, 9]

The practical phase involved installing and running Isaac Lab on an Ubuntu operating system, following official documentation and available configuration guidelines. A robotic manipulation task using Franka Emika Panda robot was selected as the demonstration case to evaluate functionality and observe the behaviour of the environment in practice. Emphasis on this was placed on identifying system strengths, usability challenges and technical limitations relevant to academic use.

The author declares no affiliation, sponsorship or financial relationship with NVIDIA or any other commercial entity. The thesis was conducted independently within an academic environment, and all analyses were based solely on publicly available documentation, the author's own practical experimentation with the Isaac Lab framework, and

informal troubleshooting assistance obtained from the NVIDIA community forums and NVIDIA Omniverse Discord channel.

### **3.2 Overview of NVIDIA Isaac ecosystem**

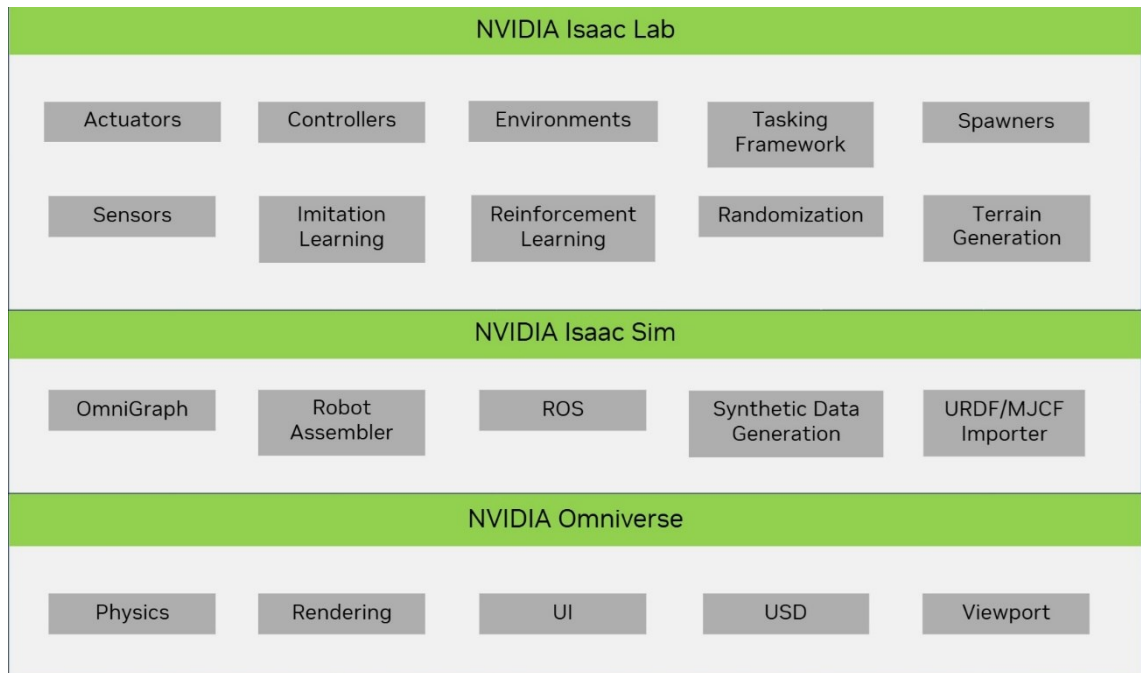
NVIDIA Isaac Lab is a modern, open-source framework for robotic learning and simulation, developed as part of the larger Isaac ecosystem. It is built on top of Isaac Sim and NVIDIA Omniverse. Isaac Lab provides a unified environment for training, testing and evaluating robot control algorithms under realistic physical and visual conditions. Isaac Lab serves as the successor to NVIDIA's earlier simulation frameworks including Orbit, IsaacGymEnvs and OmnisaacGymEnvs. It consolidates their capabilities into a single extensible framework. [10]

The framework is designed to support reinforcement learning, imitation learning, and other data-driven control techniques. It allows developers to create customized training environments for tasks such as robotic manipulation, locomotion and perception. Isaac Lab integrates advanced features like physics simulation, high-fidelity rendering and domain randomisation to support the transition from simulation to real-world deployment. Its modular and research-oriented architecture makes Isaac Lab well suited for experimental applications in robotics and even AI. [2, 8]

The technological architecture of Isaac Lab is built on the NVIDIA Omniverse platform and it follows a layered structure that integrates simulation, rendering and robot learning functionalities. At the base level, Omniverse provides the core simulation infrastructure through the PhysX physics engine and the Universal Scene Description (USD) standard for representation of 3D environments. On top of this foundation Isaac Sim adds robotics-specific capabilities. These include ROS and ROS 2 integration, realistic sensor simulation, synthetic data generation and domain randomisation tools. [10]

Isaac Lab represents the top layer. It introduces a research-oriented framework that unifies reinforcement learning environments, task definitions, reward functions and configuration management into a modular system. Key components include actuator dynamics modelling, procedural terrain generation and tools for collecting human demonstration data. [10]

Through this integrated architecture, Isaac Lab inherits the simulation fidelity and scalability of Isaac Sim, while offering a flexible programming interface for creating custom learning environments and benchmark tasks [10]. The hierarchical relationship between Omniverse, Isaac Sim and Isaac Lab is illustrated in Figure 1.



**Figure 1.** Isaac ecosystem hierarchy [10]

### 3.3 System setup and operating environment

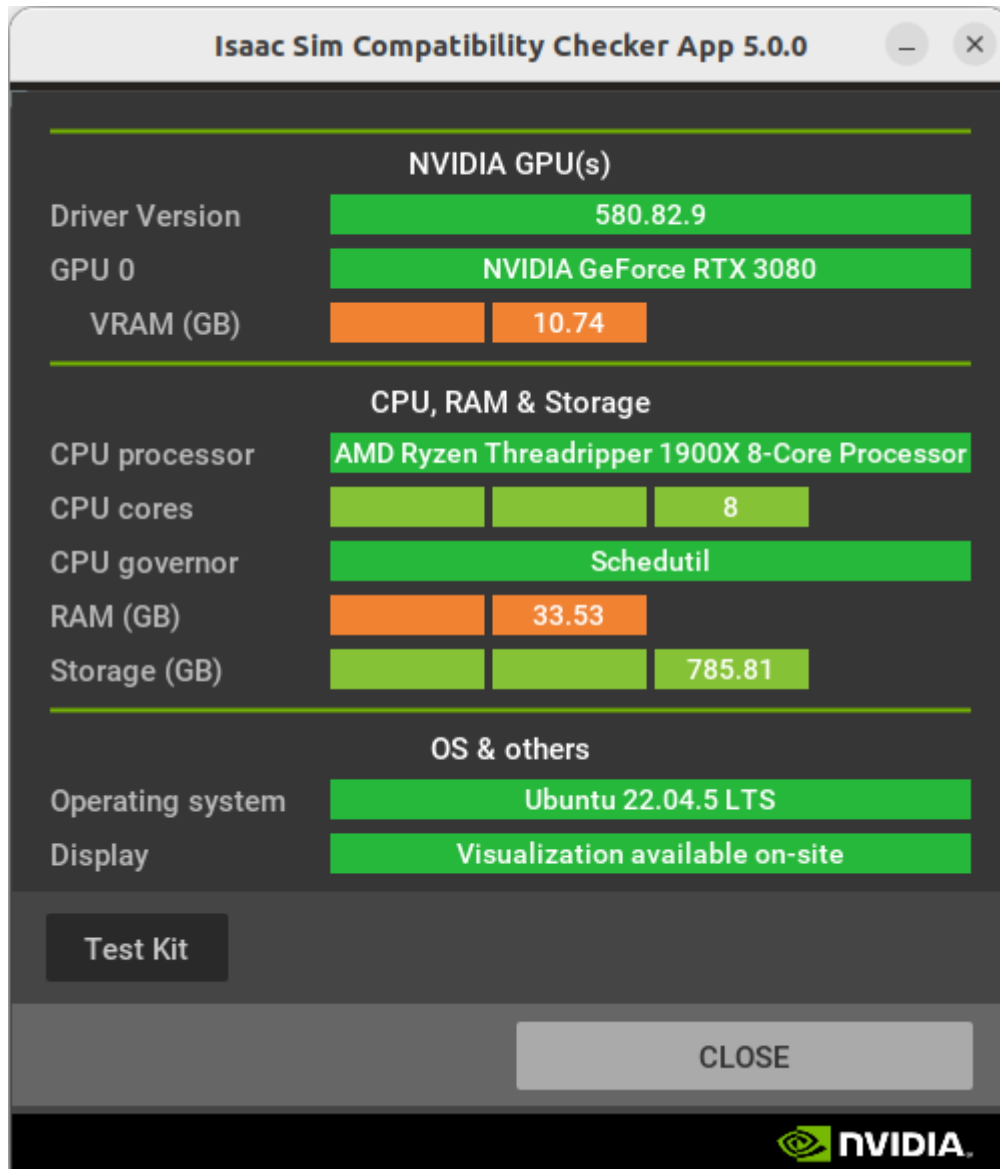
The experiments and configuration procedures described in this thesis were conducted at RoboLab, which is located in the Hervanta campus of Tampere University. The computer used was running Ubuntu 22.04 operating system and the hardware consist of an NVIDIA GeForce RTX 3080 GPU, an AMD Ryzen Threadripper 1900X CPU and 32 GB of installed RAM.

Before beginning of the installation process, the Isaac Sim Compatibility Checker was downloaded and run to ensure that the system met all hardware and software requirements [11]. Figure 2 shows the key hardware components used in the setup and confirms that all critical elements were compatible with the software. However, the available RAM and VRAM may limit performance in large-scale simulation scenarios.

Isaac Sim was installed using NVIDIA's official Quick Install guide for version 5.0 [12]. Once the Isaac Sim installation was verified by a test run, Isaac Lab was set up according to the official pre-built binaries installation instructions [13].

Both installations were carried out within a user virtual (UV) environment, since the computer used for the experiments was part of shared robotics' research laboratory at

Tampere University. This use of UV also supported the use of Python 3.11, which is required for compatibility with Isaac Sim version 5.0. After successful installation and verification that both Isaac Lab and Isaac Sim launched without errors, the framework was prepared for use in the subsequent experimental and analytical phases of the study.



**Figure 2.** Isaac Sim Compatibility Checker results

## 4. FRAMEWORK ANALYSIS

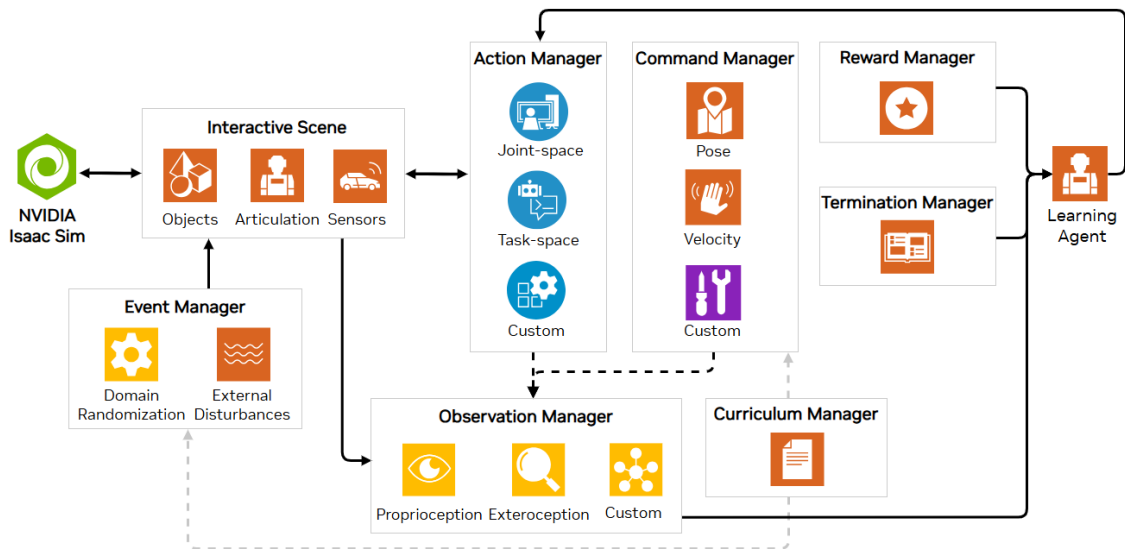
This chapter presents the technical analysis and practical evaluation of the NVIDIA Isaac Lab framework. At first, the analysis examines Isaac Lab’s internal architecture and modular design, describing how the system coordinates simulation control with the learning process. It explores how reinforcement learning algorithms are integrated within the framework, emphasising the interaction within the ecosystem. To demonstrate this in practice, the case study using the Franka Emika Panda robotic manipulator presents the framework’s practical implementation in a reinforcement learning task.

### 4.1 System architecture and modular design

The internal structure of NVIDIA Isaac Lab is based on a modular design. This separates key components of simulation control, observation processing and learning management into independent elements known as managers. [14]

The core managers include the Action Manager, which computes robot control actions, and the Command Manager that issues actuation and motion commands. The framework also uses the Reward Manager, which is responsible for evaluating task performance and generating feedback, as well as the Observation Manager, which gathers sensory and state information from the simulated environment. Additional to these, there are modules, such as the Event Manager and Curriculum Manager, supporting task-level randomisation, domain variability and the development of progressive training strategies. [14]

Figure 3 presents this manager-based architecture, showing how control and feedback modules are integrated into a complete reinforcement learning environment. Each manager can be independently replaced or extended. This design approach also reflects core software engineering principles such as abstraction, interface consistency and component reusability.



**Figure 3.** Manager-based architecture for robot learning task in NVIDIA Isaac Lab [14]

## 4.2 Reinforcement learning integration

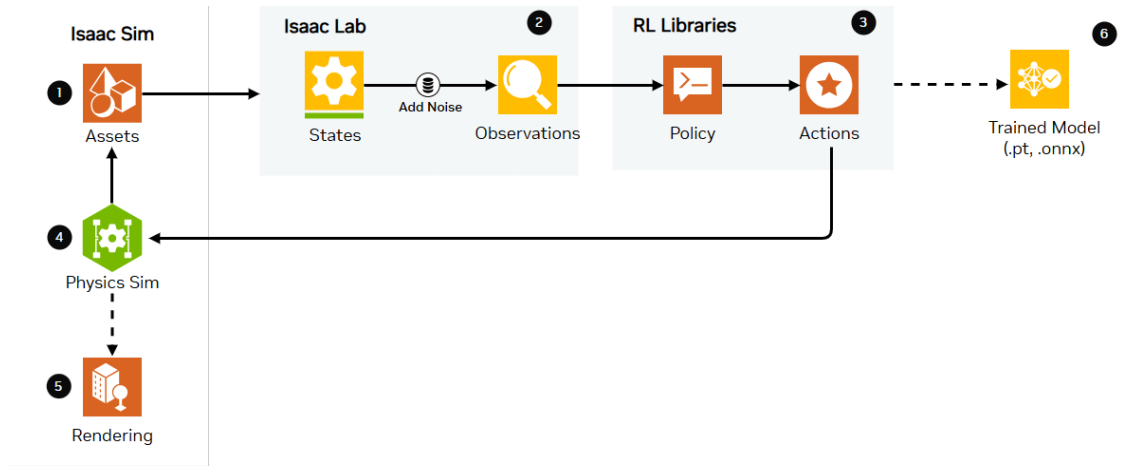
At the operational level, Isaac Lab connects these modular components to the physical simulation engine in Isaac Sim and to external reinforcement learning libraries. The framework relies on GPU-accelerated processing to ensure high simulation throughput.

Isaac Lab integrates reinforcement learning into its simulation pipeline through a unified interface that supports third-party RL libraries, such as RSL-RL, RL Games, SKRL and SB3 [14]. These integrations are managed through Isaac Lab’s modular configuration system, which allows users to switch between learning libraries and adjust hyperparameters with minimal changes to the code [14].

In practice, Isaac Sim handles physics simulation and rendering, Isaac Lab computes observations and rewards, and the reinforcement learning library, such as Proximal Policy Optimization (PPO), performs the optimisation. PPO improves training stability and sample efficiency by constraining policy updates within a predefined trust region [15, pp. 2–8].

This interaction between simulation, task design and the learning back end enables Isaac Lab to serve as a complete platform for robotic policy training and evaluation. The practical component of this thesis was done using single-GPU training setup. Therefore, only that will be here presented. Figure 4 illustrates the data flow in a single-GPU training model. The diagram highlights how Isaac Lab mediates between Isaac

Sim and the learning algorithm, handling data transfer, observations and step synchronization. [14]



**Figure 4.** Workflow for single GPU training [14]

### 4.3 RL demonstration: Franka Emika Panda Lift Cube

To evaluate the reinforcement learning capabilities of Isaac Lab a default training task “Isaac Lift Cube Franka v0” was selected. The task involves training a Franka Emika Panda robotic manipulator to grasp and lift a cube using reinforcement learning methods.

The experiment was conducted using Isaac Lab’s integrated RL pipeline where RL Games works as the backend library. Training was run from the Isaac Lab root directory on Ubuntu running the following script in terminal:

```
./isaacclab.sh -i rl_games
./isaacclab.sh -p scripts/reinforcement_learning/rl_games/train.py \
  --task Isaac-Lift-Cube-Franka-v0 \
  --num_envs 2048 \
  --max_iterations 2000
```

**Program 1.** RL Games installation and running Isaac-Lift-Cube-Franka-v0 [16]

Program 1 initializes 2048 parallel simulated environments using GPU-accelerated computation to speed up the training. Each environment contains one Panda manipulator and a cube placed on a flat surface. The objective was to train the robot to lift the cube and holding it by optimising its control policy through trial-and-error interactions.

Figure 5 shows the initial state of the training environments, where all agents begin with randomized joint configurations and untrained control policies. Figure 6 illustrates an intermediate training stage (after about one minute), where some agents perform partially successful grasps. Figure 7 shows the final stage (approximately two minutes into training), where most agents have successfully learned to lift the cube using the trained policy. This script was run multiple times; therefore, the data is approximations

of 10 repeated fresh simulations.



**Figure 5.** Initial state of training environments before policy optimisations [8]



**Figure 6.** Intermediate stage (around 1.5 minutes): some agents have achieved successful grasps [8]



**Figure 7.** Final stage (around 2 minutes): most agents have learned to lift the cube successfully [8]

The RL demonstration confirmed that the Isaac Lab framework was properly installed and fully operational for large-scale reinforcement learning experiments. The results indicated stable GPU performance and demonstrated efficient parallel processing across 2048 simulated environments.

## 5. DISCUSSION

The chapter discusses the findings of the analysis by reflecting on the framework's strengths and practical behaviour in relation to the objectives of the thesis. The results presented in Chapter 4 are examined from a broader perspective to evaluate how effectively Isaac Lab supports reinforcement learning research in academic environments and how it could be enhanced for that.

### 5.1 Opportunities and strengths of Isaac Lab

The analysis and experiments conducted in this study demonstrate that NVIDIA Isaac Lab provides a powerful and flexible platform for reinforcement learning research in robotics. Its primary strengths include its modular architecture, computational scalability, and strong integration with both NVIDIA's simulation ecosystem and widely used open-source learning libraries.

One of the most significant advantages of Isaac Lab is its modular and extensible design. The manager-based architecture divides the system into functional components that handle specific tasks including reward generation, observation processing and task control. Each component can be configured or replaced independently, which makes the framework easier to modify and extend. This structure supports rapid experimentation, since researchers can modify a single element of the system without affecting the entire framework. Such modularity enhances maintainability and promotes repeatability, both which are essential in academic research.

Another significant strength of the framework is its computational scalability. Isaac Lab efficiently utilizes GPU acceleration to simulate thousands of environments in parallel reducing training time for reinforcement learning models. In the chapter 4.3 demonstration, the framework maintained stable and consistent GPU performance across more than 2000 parallel environments. This capability enables researchers to train complex control policies in a fraction of the time required by traditional CPU-based simulators.

Additionally, Isaac Lab also benefits from strong ecosystem integration and compatibility. It is built on top of Isaac Sim and the Omniverse platform, from which it inherits advanced physics, rendering and sensor simulation capabilities. Besides this, it supports several major reinforcement learning libraries through a unified configuration interface. This flexibility allows users to test and benchmark algorithms with minimal modification to core code supporting comparative studies and cross-framework validation.

Isaac Lab’s open-source and research-oriented development model offers significant advantages for the machine learning community. The framework is actively developed, well-documented and supported by NVIDIA while remaining accessible for modification and extension.

## **5.2 Limitations and challenges in practice**

The analysis identified several practical challenges that users may encounter when working with NVIDIA Isaac Lab, particularly in academic environments. First of all, Isaac Lab can only utilise NVIDIA GPUs, as the framework depends on NVIDIA’s own drivers that support ray tracing. This limits flexibility in environments with varying computer hardware.

Isaac Lab has high computational requirements, as the framework depends on GPU-accelerated simulation. In the study, the computer handled the experiments reliably. Even so, the overall hardware demand remains considerably large. While many universities maintain at least a few high-performance machines focused on this kind of research, supporting larger classes working at the same time, would require several computers with comparable GPU capacity.

Another notable challenge is setting up the framework. It can be demanding, especially on shared machines. Installation was successful by following NVIDIA’s documentation, but the process requires careful attention, particularly for users new to Ubuntu. The main challenge involved dependency resolution, in which the author relied on the help of LLMs.

Alongside these setup challenges, the learning curve is steep. Basic python skills are essential, but effective use of Isaac Lab requires experience with simulation frameworks, reinforcement learning workflows and debugging multi-component systems. Creating or modifying custom scripts can be challenging.

A further concern is what the rapid software development introduces. NVIDIA ecosystem is being developed rapidly and updates occasionally may change file paths, APIs, or configuration structures. This phenomena is common among modern software development and it requires users to monitor version changes closely to avoid compatibility issues in ongoing projects.

## **5.3 Comparison to intended goals**

The primary objective of this thesis was to analyse the capabilities of NVIDIA Isaac Lab as a reinforcement learning and simulation framework and to evaluate its suitability for

academic robotics research. This work focused on understanding the framework's structure, evaluating its functionality and identifying its strengths and limitations through both literature-based analysis and practical experimentation.

These objectives were achieved by examining the internal architecture of Isaac Lab by analysing its connection with Isaac Sim and external learning libraries, and demonstrating reinforcement learning training using a predefined Franka Emika Panda task. This provided a realistic perspective on the framework's operation in practice and on its effectiveness in supporting reinforcement learning workflows.

The initial plan was to implement self-wrote cube stacking-task to evaluate the framework more extensively. Although this task was not completed, the attempt itself provided valuable insight into the practical requirements of creating new environments. The predefined Lift Cube-task was therefore used as the basis for evaluation, which remained consistent with the analytical goal of the study.

Overall, the thesis meets its intended objectives within the planned scope and presents a clear evaluation of Isaac Lab both technical and practical perspectives.

## **5.4 Future prospects and development needs**

Based on the findings of this study, several ideas for future development can be identified that would enhance Isaac Lab's usability, functionality and potential as a tool for education.

One important area for improvement is the creation and development of custom tasks and environments. The modular architecture of Isaac Lab provides a strong foundation, while developing new scenarios still requires significant technical expertise. Therefore, more beginner-friendly templates, expanded documentation and easier step-by-step examples would help lower the barrier for users who aim to design their own tasks.

Another important improvement relates to academic use. Isaac Lab has a potential to be used as a teaching platform for hands-on reinforcement learning and robot simulation, but this would require pedagogical resources. This would mean course-ready example environments, structured learning materials and simplified workflows targeted at students would support its use in university courses and laboratory exercises. Improvements like this would be particularly valuable if computational resources are limited.

From research perspective, future extensions could focus on improving support for sim-to-real pipelines, expanding domain randomisation techniques and enabling multi-

agent learning scenarios. Maintaining stable APIs and version compatibility would enhance the framework's suitability for reproducible research and collaborative development across institutions.

## 5.5 Recommendation for further work

Based on the analysis in this thesis several directions for future research can be identified. These recommendations aim to extend the present findings, address the limitations faced during experimentation and support the continued development of Isaac Lab as both research tool and a teaching platform.

### 1. Development of custom reinforcement learning tasks

Creating original manipulation or locomotion scenarios would offer a deeper evaluation of Isaac Lab's flexibility. Such work would also clarify the practical requirements of task construction, reward design and observation modelling. This would help to evaluate how easily new tasks can be implemented

### 2. Benchmarking of supported reinforcement learning libraries

Isaac Lab integrates several reinforcement learning libraries that were mentioned in this thesis including RSL-RL, RL Games, SKRL and SB3. A systematic comparison of their performance, stability and configuration requirements would help researchers select the most suitable one for their specific task.

### 3. Evaluation of sim-to-real transfer

Testing whether policies trained in Isaac Lab can generalise to physical hardware would give important information about the framework's applicability to experimental robotics and its effectiveness for real-world deployment.

### 4. Pedagogical development

Developing course-ready example scenarios, simplified student workflows, and structured instructional materials could make Isaac Lab more accessible for university courses focused on reinforcement learning and robot simulation.

## 6. CONCLUSIONS

The thesis aimed to determine how suitable NVIDIA Isaac Lab is as a reinforcement learning framework for academic robotics research. It examined the capabilities of NVIDIA Isaac Lab as a simulation and reinforcement learning framework and evaluated its suitability for use in academic environments.

The theoretical part of the thesis described how machine learning, and especially reinforcement learning enables robots to learn control policies through interaction with their environment rather than relying solely on explicitly programmed rules. The role of digital twin concepts and sim-to-real transfer was also discussed as a foundation for applying simulation frameworks such as Isaac Lab in practical robotics applications.

The analysis showed that Isaac Lab is built on a layered ecosystem where it sits on top of NVIDIA Omniverse and Isaac Sim. Isaac Lab extends Isaac Sim with a manager-based architecture that separates control, observation, reward and task logic into modular components. This structure supports flexibility, reusability and the development of complex learning tasks, while also benefitting from GPU-accelerated physics and high-fidelity rendering provided by the underlying platform.

The practical case study using the Isaac Lift Cube Franka v0-task demonstrated that Isaac Lab can successfully train a Franka Emika Panda manipulator to grasp and lift a cube in a short period of time using reinforcement learning. The experiment confirmed that the framework operates reliably on a single high-performance GPU and can simulate thousands of environments in parallel. Such capability enables efficient policy optimisation within short training periods.

The study also identified several challenges relevant to academic use. These include restrictions to a single GPU brand, high computational requirements, problematic configuration process on multi-use shared systems, a steep learning curve for new users and the need to monitor software updates and how they might affect current projects. The initial plan to implement a custom cube-stacking task could not be completed for this work highlights the practical difficulty of creating new environments and scenarios without earlier experience.

In conclusion, the results indicate that Isaac Lab is a powerful and flexible framework for reinforcement learning in robotics when appropriate hardware and expertise are availa-

ble. It showed strong technical capabilities and a solid foundation for research and potentially for educational use. Even though it also presents practical challenges that should be addressed if it is considered for use in academic teaching.

## REFERENCES

- [1] R.S. Sutton, A. Barto, *Reinforcement learning: an introduction*. Second edition. Cambridge, Massachusetts London, England: The MIT Press, 2020.
- [2] M. Mittal, C. Yu, Q. Yu, J. Liu, N. Rudin, D. Hoeller, et al. Orbit: A Unified Simulation Framework for Interactive Robot Learning Environments. *IEEE Robot Autom Lett*, vol 8, 2023, pp. 3740–3747.
- [3] B. Siciliano, *Springer Handbook of Robotics*. 2nd ed. Cham: Springer International Publishing AG, 2016.
- [4] Levine S, Finn C, Darrell T, et al. End-to-end training of deep visuomotor policies. *J Mach Learn Res*, vol 17, 2016, pp. 1334–1373.
- [5] F. Tao, H. Zhang, A. Liu, A.Y.C. Nee, Digital Twin in Industry: State-of-the-Art. *IEEE Trans Ind Inform*, vol 15, 2019, pp. 2405–2415.
- [6] Y. Liu, H. Xu, D. Liu, L. Wang, A digital twin-based sim-to-real transfer for deep reinforcement learning-enabled industrial robot grasping. *Robot Comput-Integr Manuf*, vol 78, 2022.
- [7] X.B. Peng, M. Andrychowicz, W. Zaremba, P. Abbeel, Sim-to-Real Transfer of Robotic Control with Dynamics Randomization. *IEEE International Conference on Robotics and Automation (ICRA)*. Brisbane, QLD: IEEE, 2018, pp. 3803–3810.
- [8] The Isaac Lab Project Developers. Isaac Lab Documentation, <https://isaac-sim.github.io/IsaacLab/main/index.html> (2025, accessed 3 November 2025).
- [9] NVIDIA Isaac Lab Open-Source Modular Framework. *NVIDIA Developer*, <https://developer.nvidia.com/isaac/lab> (accessed 24 October 2025).
- [10] Isaac Lab Ecosystem — Isaac Lab Documentation, <https://isaac-sim.github.io/IsaacLab/main/source/setup/ecosystem.html> (accessed 4 November 2025).
- [11] Isaac Sim Requirements — Isaac Sim Documentation, <https://docs.isaacsim.omniverse.nvidia.com/5.0.0/installation/requirements.html> (accessed 12 November 2025).
- [12] Quick Install — Isaac Sim Documentation, <https://docs.isaacsim.omniverse.nvidia.com/5.0.0/installation/quick-install.html> (accessed 5 November 2025).
- [13] Installation using Isaac Sim Pre-built Binaries — Isaac Lab Documentation, [https://isaac-sim.github.io/IsaacLab/main/source/setup/installation/binaries\\_installation.html#isaaclab-binaries-installation](https://isaac-sim.github.io/IsaacLab/main/source/setup/installation/binaries_installation.html#isaaclab-binaries-installation) (accessed 5 November 2025).
- [14] Reference Architecture — Isaac Lab Documentation, [https://isaac-sim.github.io/IsaacLab/main/source/refs/reference\\_architecture/index.html](https://isaac-sim.github.io/IsaacLab/main/source/refs/reference_architecture/index.html) (accessed 3 November 2025).

- [15] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal Policy Optimization Algorithms. Epub ahead of print 28 August 2017. DOI: 10.48550/arXiv.1707.06347.
- [16] Reinforcement Learning Scripts — Isaac Lab Documentation, [https://isaacsim.github.io/IsaacLab/main/source/overview/reinforcement-learning/rl\\_existing\\_scripts.html](https://isaacsim.github.io/IsaacLab/main/source/overview/reinforcement-learning/rl_existing_scripts.html) (accessed 12 November 2025).