ELTE
EÖTVÖS LORÁND
UNIVERSITY

CoLocation Center for Academic and Industrial Cooperation
Faculty of Informatics
Eötvös Loránd University

Advancing Data-Driven Robotics with Transfer and Curriculum Learning

written by

Dániel Horváth

Supervisor: Dr. Zoltán Istenes, PhD
Industrial Supervisor: Dr. Ferenc Gábor Erdős, PhD
Campus France Internship Supervisor: Prof. Fabien Moutarde, PhD

Doctoral School of Informatics
Head: Prof. Zoltán Horváth, PhD

Doctoral Program of Information Systems
Head: Prof. András Benczúr, PhD

A thesis submitted in fulfilment of the requirements
for the degree of Doctor of Philosophy in Computer Science

DOI: 10.15476/ELTE.2024.375

Budapest, 2024

"La science n'a pas de patrie, parce que le savoir est le patrimoine de l'humanité, le flambeau qui éclaire le monde."

"Science has no homeland, because knowledge is the heritage of humanity, the torch that illuminates the world."

Louis Pasteur

Abstract

The deep learning revolution has fundamentally reshaped numerous fields, including robotics. However, as in other fields, certain challenges must be overcome to exploit the power of deep learning algorithms and create truly adaptive intelligent robots. The difficulty lies less in adult-level intelligence than in the skills of perception and mobility, also referred to as Moravec's paradox. In this context, the key issues are transferability and universality. This thesis addresses data-driven robotics, with a focus on transfer and curriculum learning. My main contributions are as follows.

Robots operating in unstructured environments need to effectively sense and interpret their surroundings. A major challenge for deep learning models in the field of robotics is the lack of domain-specific labelled data for various industrial applications. To bridge the reality gap, I developed a sim2real transfer learning method based on domain randomization for object detection (S2R-ObjDet), enabling automatic generation of labelled synthetic data. In addition, I propose the generalised confusion matrix (GCM) which addresses the limitations of the classical precision-recall-based metrics. I also introduce a public and annotated real-world dataset of industrial objects (InO-10-190) for evaluating sim2real object detection methods.

In object manipulation, it is essential to estimate not only object positions but also their poses. Thus, I propose two vision-based, multi-object grasp pose estimation models – the real-time MOGPE-RT and the high-precision MOGPE-HP – as well as the extension of the S2R-ObjDet method to pose estimation (S2R-PosEst). This framework provides an industrial tool for rapid data generation and model training while requiring minimal data from the target distribution.

Reinforcement learning – inspired by human learning – aims to offer a universal solution to various problems. Nevertheless, the field of robotics poses significant challenges. To facilitate the exploration of reinforcement learning robot agents, I propose a data exploitation curriculum learning method, called highlight experience replay (HiER). The experimental results demonstrate that HiER significantly improves the performance of the state-of-the-art, exhibiting stochastic dominance over them. To further enhance HiER, I introduce HiER+, which integrates an arbitrary data collection curriculum learning method for which I propose the easy2hard initial state entropy method (E2H-ISE).

Although the results presented in this thesis are my own, henceforth, I will use plural wording for stylistic purposes. The implementations, the qualitative results, the video presentations, and further materials are available on the project site: www.danielhorvath.eu/thesis.

Kivonat

A mély tanulás forradalma alapjaiban változtatta meg számos tudományterületet, beleértve a robotikát is. Azonban, ahhoz, hogy a mélytanulási algoritmusok lehetőségeit kiaknázzuk és adaptív, intelligens robotokat hozzunk létre, bizonyos kihívásokat le kell küzdeni. A nehézség elsősorban az érzékelési és mozgási képességek elsajátításában rejlik mintsem a felnőtt szintű intelligencia elérésében (Moravec-paradoxon). A legfőbb kihívások a transzferabilitás és az univerzalitás. A dolgozatban ezen kihívásokra koncentrálva, a transzfer- és a curriculum tanulás (CL) segítségével kívánok megoldásokat adni. Fő eredményeim a következők.

Strukturálatlan környezetben működő robotoknak képeseknek kell lenniük észlelni és értelmezniük a környezetüket. A mélytanulási modellek alkalmazásának egyik fő akadálya a feladat-specifikus címkézett adatok hiánya. A valóság és a szimuláció közötti különbség áthidalása érdekében kifejlesztettem egy domén randomizáción alapuló sim2real tudástranszfer módszertant objektum detektáláshoz (S2R-ObjDet), amely lehetővé teszi címkézett szintetikus adatok automatikus generálását. Továbbá, javaslom a generalizált zavarmátrixot (GCM) ami a klasszikus precizitás-szenzitivitás alapú metrikák hiányosságaira kínál megoldást. Ezenkívül készítettem egy nyilvános és annotált, valós ipari tárgyakból álló adatbázist (InO-10-190).

Objektumok mozgatásához nem csak a pozíciójuk, hanem az orientációjuk ismerete is szükséges. Ennek érdekében, két, gépi látáson alapuló, több objektum egyidejű megfogási helyzetének becslésére alkalmas modellt javaslok – a valós idejű MOGPE-RT és a nagy pontosságú MOGPE-HP modelleket. Továbbá az S2R-PosEst módszertant mely az S2R-ObjDet metódus kiterjesztése orientáció becslésre. Ez a keretrendszer ipari eszközt biztosít a gyors adatgeneráláshoz és modelltréninghez, miközben minimális valós adatot igényel.

A megerősítéses tanítás (RL) – az emberi tanulást mintául véve – univerzális megoldást kíván nyújtani különféle problémákra. Ugyanakkor a robotika jelentős kihívásokat állít. A robotágensek hatékony tanításának érdekében javaslom a kiemelt tapasztalati puffer (HiER), adatkiaknázó CL módszeremet, mely jelentősen javítja a state-of-the-art módszerek teljesítményét. Ennek továbbfejlesztésére, bevezetem a HiER+ módszert, amelyben a HiER egy tetszőleges adatgyűjtési CL módszerrel egészül ki, például az általam javasolt easy2hard kezdeti állapot entrópia módszerrel (E2H-ISE).

Bár a disszertációban bemutatott eredmények a saját munkám, a továbbiakban stilisztikai okokból a többes szám első személyt fogom használni. Az implementációk, kvalitatív eredmények, videó prezentációk és további anyagok elérhetők a disszertáció weboldalán: www.danielhorvath.eu/thesis.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisors, Zoltán Istenes, Gábor Erdős, and Fabien Moutarde for their invaluable guidance and continuous support throughout this journey. I am also immensely thankful for the fruitful collaboration with my article co-authors, Jesús Bujalance, Sándor Földi, Kristóf Bocsi, and Tomáš Horváth.

I sincerely appreciate the members of the jury and reviewers for their time, effort, and thoughtful evaluation. Your expertise and feedback are invaluable, and I am grateful for your important contributions to this process.

I extend my heartfelt appreciation to József Váncza and László Monostori for providing me with the tools and freedom to pursue my research. I am also indebted to Richard Beregi for his mentorship during my student's projects. Additionally, I deeply appreciate the support of Zsolt Kemény, András Kovács, Emma Takács, Tamás Cserteg, Bence Tipary, Gergely Horváth, Mátyás Hajós, Markó Horváth, Ádám Juniki, Nelli Nyisztor, Tekla Tóth, Kristóf Abai, Peter Smejkál, Bálint Laza, Péter Dobrovoczki, Ambrus Tamás, Julia Bergmann, János Nacsa, Imre Paniti, Judit Megyery, and everyone else who has supported me in one way or another.

Engaging in a nine-month research internship at Mines Paris was an amazing experience. I am deeply grateful to Fabien Moutarde for the opportunity, and to Bassam Abdallah for his assistance. I would also like to extend my appreciation to Thomas Gilles, Camille Truong-Allié, Joseph Gesnouin, Arthur Moreau, Sascha Hornauer, Raphael Chekroun, Louis Soum-Fontez, Hugo Blanc, Fábio Elnecave Xavier, Jules Sanchez, Sofiane Horache, Sami Jouaber, and everyone else who welcomed me into the lab and for the engaging discussions.

I also appreciate the support of my friends throughout this journey: Sára Fejes, Nick Durrant, Márton Grósz, Mathilde Jacques, Paloma Tannous, Gergely Hunyady, Péter Gönczöl, Gergely Juhász, Dániel Uzseka, Bálint Borosnyay, Bálint Prohászka, Gábor Deme, András Mészáros, Péter Szirtes, Nicolas Cavaillès-Wurmser, Eduarda Kleinsorge, Zoltán Remecz, Virgilia Sanguedolce, Louis Guisnet and to the rest of my friends!

I am deeply grateful to my amazing girlfriend, Caroline, whose love, support, and belief in me have been my greatest source of strength. Thank you for always being by my side! Je remercie également Françoise, Phong, Céline et Chris pour tout leur soutien.

Végül, de nem utolsó sorban szeretném megköszönni a családomnak az egész életemen át tartó folyamatos támogatást, anyukámnak, apukámnak, nagyszüleimnek és a család többi tagjának egyaránt. Nélkületek nem sikerült volna!

Acknowledgement of research projects and funding

This work was supported by European Union within the framework of the Artificial Intelligence National Laboratory under Project RRF-2.3.1-21-2022-00004.

My internship at the Centre for Robotics at the École Nationale Supérieure des Mines de Paris, University Paris Sciences & Lettres in the framework of "Campus France Bourse du Gouvernement Français – Bourse Excellence Hongrie" was funded by the Government of France.

Table of Contents

Abstract ii

Kivonat iii

Acknowledgements iv

List of Figures xi

List of Tables xvii

List of Algorithms xx

Abbreviations xxi

Summary of Notation xxvi

1 Introduction 1

1.1 Context: Adaptive robots 2

1.2 Problem statement: Transferability and universality 4

1.3 Contribution 5

1.4 Outline 8

2 Theoretical background **9**

2.1 Computer vision . . . 11

2.1.1 Problem formulation . . . 11

2.1.2 Convolutional neural networks . . . 13

2.1.3 Vision transformers . . . 15

2.1.4 Image classification . . . 16

2.1.5 Object detection and pose estimation . . . 18

2.2 Transfer learning . . . 22

2.2.1 Definitions and notations . . . 22

2.2.2 Sim2real object detection . . . 23

2.3 Reinforcement learning . . . 24

2.3.1 Markov decision process . . . 25

2.3.2 Multi-goal tasks . . . 25

2.3.3 Reward functions . . . 26

2.3.4 Demonstrations . . . 28

2.3.5 Categorization of RL algorithms . . . 28

2.3.6 Tabular reinforcement learning . . . 30

2.3.7 Reinforcement learning in robotics . . . 30

2.3.8 Evaluation of RL algorithms . . . 33

2.4 Curriculum learning . . . 34

2.4.1 Easy2hard curriculum learning . . . 35

2.4.2 Generalised curriculum learning . . . 36

2.4.3 CL in supervised learning . . . 37

2.4.4 CL in reinforcement learning . . . 37

3 Sim2real knowledge transfer for object detection **38**
3.1 Introduction . . . 40
3.2 Related work . . . 43
3.3 The S2R-ObjDet method . . . 47
3.3.1 Sim2real knowledge transfer . . . 47
3.3.2 Data generation . . . 49
3.3.3 Training . . . 57
3.4 Evaluation protocol . . . 57
3.5 Generalised confusion matrix for object detection . . . 58
3.6 The InO-10-190 dataset . . . 61
3.7 Results . . . 64
3.7.1 Zero-shot transfer (ZST) . . . 66
3.7.2 One-shot transfer (OST) . . . 69
3.8 Ablation study . . . 74
3.8.1 Seed . . . 74
3.8.2 Texture and post-processing . . . 75
3.8.3 Data size . . . 76
3.8.4 Gravity, positional disturbance, and bounding-box calculation . . . 77
3.8.5 Cutouts . . . 79
3.8.6 Faster R-CNN . . . 79
3.9 Robotic application . . . 80
3.10 Conclusion . . . 81

4 Sim2real grasp pose estimation **85**
4.1 Introduction . . . 86
4.2 Problem statement . . . 88
4.3 Related works . . . 89
4.4 Approach . . . 90

4.4.1 Object detection with S2R-ObjDet . . . 90
4.4.2 ROI cropping . . . 91
4.4.3 Orientation estimation with S2R-PosEst . . . 91
4.4.4 Pattern matching (optional) . . . 93
4.5 Robot control architecture . . . 94
4.6 Results . . . 96
4.6.1 Setting of the robotic experiments . . . 96
4.6.2 Object detection . . . 97
4.6.3 Orientation estimation . . . 97
4.6.4 Robotic grasping . . . 98
4.7 Conclusion . . . 99

5 Highlight experience replay **102**
5.1 Introduction . . . 103
5.2 Related works . . . 105
5.2.1 Data exploitation . . . 106
5.2.2 Data collection . . . 106
5.3 Method . . . 109
5.3.1 HiER . . . 109
5.3.2 E2H-ISE . . . 112
5.3.3 HiER+ . . . 115
5.4 Results . . . 115
5.4.1 Evaluation protocol . . . 118
5.4.2 Aggregated results across all tasks . . . 119
5.4.3 Panda-Gym . . . 122
5.4.4 Gymnasium-Robotics Fetch . . . 123
5.4.5 Gymnasium-Robotics PointMaze . . . 123
5.4.6 Qualitative evaluation . . . 128

5.4.7 HiER λ, HiER ξ, and E2H-ISE c versions . . . 130
5.4.8 TD3 and DDPG . . . 130
5.5 Conclusion . . . 130

6 Conclusions **137**
6.1 Summary of thesis achievements . . . 138
6.2 Future work . . . 139

APPENDICES **141**

A Machine learning **142**
A.1 Context . . . 142
A.2 Formulation . . . 143
A.3 Neural networks . . . 144
A.4 Deep learning . . . 145

B Control theory and reinforcement learning **147**

C Safe reinforcement learning **150**

D Tabular reinforcement learning **156**

E Detailed results of HiER and HiER+ **158**

References **161**

List of Figures

2.1 Comparison of classification, object detection, and instance segmentation from [105]. . . . 12

2.2 Architecture of VGG16 [111]: The feature extractor is composed of successive convolutional layers (with ReLU activations) and max pooling layers, while the classification head comprises fully connected layers with ReLU activations. The image has been adapted from [112]. . . . 15

2.3 Confusion matrix. . . . 18

2.4 The comparison of axis-aligned bounding boxes and oriented bounding boxes. 19

2.5 An example of the precision-recall curve from [116]. The colour bar indicates the τ_{con} threshold values. . . . 21

2.6 The agent–environment interaction in a Markov decision process [73]. The done flag is not depicted. . . . 25

2.7 **Top.** The architecture of the traditional actor-critic model. The actor and the critic are trained separately. **Bottom.** The architecture of DDPG. The actor and the critic are trained together. As the value function $Q(s_t, a_t)$ is the target for the actor, the policy can be deterministic. For training, the policy must be deterministic because sampling would disrupt the continuity of the gradient. . . . 33

2.8 Illustration of the easy2hard CL from [98]. . . . 35

2.9 Illustration of the continuation method from [163]. . . . 36

3.1 **Top.** Pipeline of knowledge transfer. **Bottom.** Flowchart diagram of our data generation, training, and evaluation process. The picture of the Boston bull is from ImageNet [70]. 41

3.2 Some examples of the textures used from [222]–[224]. 51

3.3 Post-process transformation on a blank image. 56

3.4 Generalised confusion matrix (GCM). In the cyan frame, the traditional 10×10 confusional matrix for the 10 classes. The correct detections are in the diagonal, marked with red dotted lines. An extra row is added, marked in green, to the predictions that do not belong to any ground truth object (false positives). Furthermore, an extra column is added, marked in yellow, for the objects that were not found (false negatives). Finally, for simplicity, marked in orange, the extra square at the bottom right of the matrix which is zero by definition. In this example, it can be seen at a glance, that several (62) bonnet objects were misclassified as body objects, and only 16 bonnet objects were classified correctly. 60

3.5 The selected industrial parts in the InO-10-190 dataset. Their names in order of their identifier numbers are the following: 1. L-bracket, 2. U-bracket, 3. angle bracket, 4. seat, 5. pipe clamp, 6. handle, 7. bonnet, 8. body, 9. ball, 10. cable shoe. The letter 'F' designates the camera holder frame. The green dashed lines show the borders of the cropped images. . . 62

3.6 The 3D models of the selected industrial parts in the InO-10-190 dataset. Their names in order of their identifier numbers are the following: 1. L-bracket, 2. U-bracket, 3. angle bracket, 4. seat, 5. pipe clamp, 6. handle, 7. bonnet, 8. body, 9. ball, 10. cable shoe. Their scaling factors are different for better visualization. 62

3.7 The similarity of the body and the bonnet objects. 63

3.8 Samples of our public and annotated InO-10-190 dataset (cropped version). 64

3.9 Class distributions of the InO-10-190 dataset. 65

3.10 Two examples of synthetic images with the automatically generated annotations. The bounding boxes are shown here for illustration purpose only. . 66

3.11 Results of the `ZST_BEST` models. **Left.** The precision-recall curves. **Right.** The F_1 scores. The train and valid scores overlap and are relatively close to the perfect 100% score. 67

3.12 The average mAP_{50} scores of the `ZST_BEST` models in the different classes. 69

3.13 Class specific results of the ZST_BEST$_1$ model. **Left.** The precision-recall curves on the cropped images. **Right.** The GCM is evaluated on the cropped images with $\tau_{\text{con}} = 0.8$. 70

3.14 Qualitative evaluation. **Left.** An accurate example. **Right.** An inaccurate prediction. Both are the results of the ZST_BEST$_1$ model with $\tau_{\text{con}} = 0.8$. The colour-coding follows Fig. 3.9. 71

3.15 Results of the OST_BEST models. **Left.** The precision-recall curves. **Right.** The F_1 scores. The train and valid scores overlap and are relatively close to the perfect 100% score. 72

3.16 The average mAP_{50} scores of the OST_BEST models in the different classes. 74

3.17 Class specific results of the OST_BEST$_3$ model. **Left.** The precision-recall curves on the cropped images. **Right.** The GCM on the cropped images with $\tau_{\text{con}} = 0.8$. 75

3.18 Qualitative evaluation. **Left.** An accurate example. **Right.** An inaccurate prediction. Both are the results of the OST_BEST$_3$ model with $\tau_{\text{con}} = 0.8$. The colour-coding follows Fig. 3.9. 76

3.19 Results of the ablation study on the original images (ZST models). Model without added textures (-T), without post-processing methods (-PP), and without both (-TPP). 77

3.20 The setup of the robotic application. 81

3.21 Information flow in the robotic application. The computer vision module, which is the main topic of this chapter, is highlighted in orange. In this application, the force sensor was not used (marked with the dashed line). The following software resources were used: [232]–[236]. 82

4.1 **Top.** Illustration of our S2R-ObjDet and S2R-PosEst methods. **Bottom.** The flowchart diagram of our multi-object grasp pose estimation (MOGPE) methods. 87

4.2 The data flow of the ROI cropping method. 92

4.3 The proposed CNN architecture for orientation estimation. Abbreviations are Conv: convolutional layer, FC: fully connected layer K: kernel size, FM: number of feature maps, A: activation function. 93

4.4 Examples of the generated synthetic training dataset. 94

4.5 The robot control architecture. With blue colour, the version of the MOGPE-RT model, while with orange colour, the version of the MOGPE-HP model. 95
4.6 Experimental setup. 96
4.7 An accurate (a) and an inaccurate (b) prediction. The orientation of the arm is slightly tilted in the latter case. Regarding the object detection, both examples are accurate. 99

5.1 Overview of HiER and HiER+. For every episode, the initial state is sampled from μ_0. After every episode, the transitions are stored in $\mathcal{B}_{\text{ser}}$, and in case the λ condition is fulfilled then in $\mathcal{B}_{\text{hier}}$ as well. For training, the transitions are sampled from both $\mathcal{B}_{\text{ser}}$ and $\mathcal{B}_{\text{hier}}$ according to the ratio ξ. For a detailed description, see Algorithm 1 and 2. 105
5.2 Visualization of the effect of parameter c on μ_0 in a 2D case where state $s = [s_x, s_y]$. The initial state $s_0 = [s_{0,x}, s_{0,y}]$ is sampled from the probability distribution $\mu_0(c)$. 114
5.3 HiER compared to the state-of-the-art across all tasks with 95% CIs. Both HiER version outperform their corresponding baseline. HiER [HER] yields the best performance in all metrics. The point estimates are presented in Tab. 5.2. 120
5.4 Performance profiles across all tasks with 95% CIs. **Left**: run-score distribution, **right**: average-score distribution. The red-dotted line shows the median values while the areas under the performance profiles correspond to the mean values (comparing with Tab. 5.2, the average-score distribution needs to be examined). Both HiER and HiER [HER] have stochastic dominance over their corresponding baselines. 121
5.5 Probability of improvement of HiER versions compared to their corresponding baselines and themselves across all tasks with 95% CIs. The average probabilities from top to bottom are the following: 0.76, 0.88, and 0.85. . . 121
5.6 Learning curves of HiER and HiER+ with E2H-ISE compared to the state-of-the-art based on success rates on the push, slide, and pick-and-place tasks of the Panda-Gym robotic benchmark with 95% CIs. 124
5.7 Aggregate metrics on the push, slide, and pick-and-place tasks of the Panda-Gym robotic benchmark with 95% CIs. HiER (blue) and both versions of HiER+ (purple and magenta) significantly outperform the baselines (gray). E2H-ISE alone could slightly improve the performance of the baseline. . . . 124

5.8 Performance profiles (run-score distribution) on the push, slide, and pick-and-place tasks of the Panda-Gym robotic benchmark with 95% CIs. . . . 125

5.9 Probability of improvement on the push, slide, and pick-and-place tasks of the Panda-Gym robotic benchmark with 95% CIs. The average probabilities from top to bottom: 0.625, 0.857, 0.86, 1.0, 1.0, 0.99, 0.99, 1.0, 1.0, and 1.0. 125

5.10 Learning curves of HiER compared with its baselines on push, slide, and pick-and-place tasks of the Gymnasium-Robotics Fetch benchmark with 95% CIs. 126

5.11 Aggregate metrics on the push, slide, and pick-and-place tasks of the Gymnasium-Robotics Fetch benchmark with 95% CIs. Both HiER (blue) and HiER [HER] (magenta) significantly outperform the baselines (light blue and purple). 127

5.12 The tasks of Gymnasium-Robotics PointMaze environment [246]. The mazes were custom-made, thus we named them accordingly. The layouts (b) and (d) show the placement of the walls and the possible start and target positions from a top view. The environment is based on the MuJoCo simulator [188]. 129

5.13 Learning curves of HiER compared with its baselines on the Gymnasium-Robotics PointMaze environment with 95% CIs. 129

5.14 The analysis of HiER λ versions (a) and (b), and HiER ξ versions (c). ξ is fixed at 0.5 for (a) and (b). HiER λ `ama` parameters: $\lambda_0 = -50$, $\lambda_{\max} = -10$ $M = 0$ and $w = 20$. The `without` version indicates that HiER was not used. 131

5.15 Comparison of different HiER λ methods on the slide task of the Panda-Gym benchmark with 95% CIs. The `predefined` λ method is seemingly superior, although the CIs with the `fix` λ method overlap. HiER λ `ama` parameters: $\lambda_0 = -50$, $\lambda_{\max} = -10$ $M = 0$ and $w = 20$. The profiles of HiER λ are depicted on Fig. 5.14 (b). 132

5.16 Comparison of different HiER ξ methods on the slide task of the Panda-Gym benchmark with 95% CIs. The `fix` $\xi = 0.25$, $\xi = 0.5$, and the `prioritized` appear to be the best versions in this order, although their CIs overlap. . . 132

5.17 Comparison of different E2H-ISE c methods on the slide task of the Panda-Gym benchmark with 95% CIs. The parameters of the methods and the point estimates are presented in Tab. 5.6. 132

5.18 Comparison of the TD3 and DDPG versions of HiER+ with their baselines on the push, slide, and pick-and-place tasks of the Panda-Gym benchmark with 95% CIs. The point estimates are presented in Tab. 5.7. 134

A.1 Field of artificial intelligence, following [108]. 143

C.1 A comparison of model-driven, data-driven, and combined approaches from [265]. 151

C.2 Safety levels from [265]. 152

C.3 The SQRL approach from [273]. 153

C.4 The recovery RL approach from [275]. 154

C.5 The predictive safety filter from [278]. 154

C.6 Illustrations of three different notions of state-wise safety from [271]. **Left.** safety after convergence. **Middle.** safety during training with hard constraints. **Right.** safety during training with progressive safe exploration. . 155

List of Tables

3.1 Summary of related works. Abbreviations are the following Sim: Simulator, Synt: Synthetic, Img: Images, P&P: Pick-and-place, Segm: Segmentation, Class: Classification, ObjDet: Object detection, Nav: Navigation, FrR-CNN: Faster R-CNN, MJC: MuJoCo [188], Gaz: Gazebo [190], PyB: PyBullet [192], V-R: V-REP [216], UE4: Unreal Engine [197], OGL: OpenGL [217], DFP: DART [218], FleX [219], Pyrender [220], Blen: Blender [137], acc: accuracy, [a]: domain adaptation (otherwise domain randomization), [u]: unlabelled. The AP and mAP scores are with IoU=0.5. YCB [221]. 48

3.2 The most relevant input parameters of the data generator module in terms of object generation. Param: Parameter. 52

3.3 The most relevant input parameters of the data generator module in terms of image rendering. Param: Parameter. 54

3.4 The types of noises in post processing. 56

3.5 The most relevant advanced data augmentation tools in the training process. 57

3.6 Summary of the InO-10-190 dataset. 65

3.7 The mAP_{50} scores of the `ZST_BEST` models in the different test groups. . . 68

3.8 The mAP_{50} scores of the `ZST_BEST` models for the different classes. 70

3.9 Training datasets. 71

3.10 The mAP_{50} scores of the `OST_BEST` models in the different test groups. . . 73

3.11 The mAP_{50} scores of `OST_BEST` models for the different classes. 73

3.12 The mAP_{50} scores of `ZST_BEST` models with different seeds. 76

3.13 The mAP_{50} scores of different `ZST` models. `-T`: without texture, `-PP`: without post processing, `-TPP`: without post processing and texture. 78

3.14 The mAP_{50} scores of `ZST_BEST` models with different data sizes. 78

3.15 The mAP_{50} scores of `ZST` models without different factors. `-G`: no gravity, `-R`: no randomness in grid positions and no gravity, `8P`: 8-point bounding box calculation. 78

3.16 The mAP_{50} scores of `ZST` models with different cutouts at the post-processing method. 79

3.17 The mAP_{50} scores of R101-FPN Faster R-CNN model. 80

4.1 The mAP_{50} scores of the object detection model. 97

4.2 The success rate of the pose estimation model. An estimation is considered successful if it is within 10 degrees of the ground truth. 98

4.3 Results of the robotic grasping experiment. 99

5.1 Summary of related works. 107

5.2 HiER compared to the state-of-the-art across all tasks. For the reward, there is no universal desirable target, thus there is no OG value. The column-wise best results are marked in bold. Both HiER version outperform their corresponding baseline. HiER [HER] yields the best performance in all metrics. 120

5.3 Simplified summary of our results on the push, slide, and pick-and-place tasks of the Panda-Gym robotic benchmark based on success rates. The column-wise best results are marked in bold. The full table with all the configurations is presented in Tab. E.1. 126

5.4 HiER compared to the state-of-the-art based on success rates on push, slide, and pick-and-place tasks of the Gymnasium-Robotics Fetch benchmark. The column-wise best results are marked in bold. 127

5.5 HiER compared to the state-of-the-art based on success rates on the Gymnasium-Robotics PointMaze environment. The column-wise best results are marked in bold. 131

5.6 The effect of the E2H-ISE c methods on the success rates on the `Panda-Slide-v3` task. HiER parameters: λ mode `predefined` and ξ `fix` with $\xi = 0.5$. E2H-ISE parameters: `self-paced` $\Psi_{\text{low}} = 0.2$, $\Psi_{\text{high}} = 0.8$ and $\delta = 0.05$; `control`: $\psi = 0.8$ and $\delta = 0.01$; `control adaptive`: $\Delta = 0.2$, $\psi_{\max} = 0.9$, and $\delta = 0.01$. The row-wise best results are marked in bold. . . 133

5.7 HiER+ compared to the state-of-the-art based on success rates on the Panda-Gym robotic benchmark in the case of TD3 and DDPG. The column-wise best results for TD3 and DDPG separately are marked in bold. 134

A.1 Some definitions of artificial intelligence, organised into four schools of thought from [258]. 143

B.1 Comparison of the most relevant terms and their formulations of control theory and reinforcement learning. 148

E.1 HiER and HiER+ compared to the state-of-the-art based on success rates on the Panda-Gym robotic benchmark. On the left side of the header, the components of the specific algorithm are displayed (HER, PER, ISE, HiER). The column-wise best results are marked in bold. 159

E.2 HiER and HiER+ compared to the state-of-the-art based on the evaluation rewards on the Panda-Gym robotic benchmark. On the left side of the header, the components of the specific algorithm are displayed (HER, PER, ISE, HiER). The desired performance scores for the OG metric are -10, -20, and -30 for the push, slide, and pick-and-place tasks respectively. The column-wise best results are marked in bold. 160

List of Algorithms

1 HiER 110
2 HiER+ 116

Abbreviations

A2C advantage actor-critic 28, 29, 31

AABB axis-aligned bounding box 12, 18, 19, 55, 88

AGI artificial general intelligence 5

AI artificial intelligence 3, 5, 11, 142

AMI autonomous machine intelligence 5

AP average precision 20, 45, 46, 48, 68

API application programming interface 49

AVG average, arithmetic mean 68, 70, 73, 76, 78–80, 97, 98

BB bounding box 18–20, 55, 78

CI confidence interval 33, 34, 119, 130

CL curriculum learning iii, 5, 35–37, 115, 139

CMDP constrained Markov decision process 151, 152

CNN convolutional neural network 13–16, 18, 20, 43, 48, 57, 91–94, 97

CV computer vision 11, 12

DA domain adaptation 23, 50

DCNN deep convolutional neural network 41, 42, 46

DDPG deep deterministic policy gradient 28, 29, 32, 33, 104, 117, 130, 134, 135

DL deep learning 3, 4, 40, 42, 86, 142, 144, 146

DNN deep neural network 142

DoF degrees of freedom 46, 80, 88, 89, 98

DP dynamic programming 30, 156, 157

DQN deep Q-learning 28–30

DR domain randomization 23

E2H-ISE our easy2hard initial state entropy method ii, iii, xxxi, 6, 8, 104–109, 112, 114–118, 122, 130, 135, 136, 139

F_1 the harmonic mean of precision and recall 20, 40, 58, 59, 61, 67, 71, 72, 84, 138

FOV field of view 53, 66

FPS frames per second 21, 80, 81, 88, 91, 92, 100

GAN generative adversarial network 23

GCM our generalised confusion matrix ii, iii, 6, 8, 40, 42, 43, 58, 60, 61, 64, 72, 81, 82, 84, 138, 140

GIoU generalised intersection over union 20

GPU graphics processing unit 11, 40, 49, 81, 91, 92, 138

GT ground truth 11, 19, 20, 60, 143

HCPS human-cyber-physical system 40

HER hindsight experience replay 103, 104, 106, 107, 109, 110, 115–117, 119, 120, 122, 123, 126–128, 130, 131, 133, 135, 136, 139, 159, 160

HiER our highlight experience replay method ii, iii, 6, 8, 103–105, 107, 109–112, 114–120, 122, 123, 126–128, 130, 131, 133, 135, 136, 139, 140, 158–160

HiER+ our highlight experience replay method combined with a data collection curriculum learning method, eg.: with E2H-ISE ii, iii, 6, 8, 103–105, 109, 114, 115, 117–119, 122, 126, 128, 130, 134–136, 139, 158–160

i.i.d. independent and identically distributed 24, 29, 31, 32, 143, 144

ILSVRC ImageNet Large Scale Visual Recognition Challenge 13

ImageNet an online dataset for image classification 17, 41, 57

InO-10-190 our dataset of industrial objects, containing 190 images of 920 objects of 10 classes ii, iii, 6, 8, 40, 42, 43, 61, 63, 64, 81–83, 91, 138

IoU intersection over union xxvii, 19, 20, 44, 45, 48, 60, 78

IQM interquartile mean 34, 118–120, 122, 123, 126–128, 131, 134, 159, 160

ISE initial state entropy 104, 136, 139, 159, 160

KAN Kolmogorov-Arnold network 145

LLM large language model 2, 11, 140

mAP mean average precision 20, 21, 40, 43, 44, 48, 57–59, 64, 66–69, 71, 72, 75–77, 82–84, 88, 97, 100, 138

MC Monte Carlo 30, 157

MDP Markov decision process 25, 144, 148, 151, 152

ML machine learning 3, 5, 22, 35, 142–144

MLP multilayer perceptron 13, 15, 145

MOGPE our multi-object grasp pose estimation methods 6, 8, 86, 90, 138

MOGPE-HP our high-precision multi-object grasp pose estimation method ii, iii, 6, 86, 88, 90, 93, 98–100, 138

MOGPE-RT our real-time multi-object grasp pose estimation method ii, iii, 6, 86, 88, 90, 93, 95, 98–100, 138

MPC model predictive control 149, 151, 154

MPSC model predictive safety certification 154

MS COCO Microsoft Common Objects in Context, online dataset for object detection 21, 41

MSE mean square error 46, 98

NeRF neural radiance fields 23

NGIM new-generation intelligent manufacturing 40

NN neural network 13, 16, 30, 31, 144, 145

OBB oriented bounding box 19, 88

OG optimality gap 34, 118–120, 126, 127, 131, 134, 159, 160

OST one-shot transfer 65, 71, 72

PER prioritized experience replay 103, 104, 106, 107, 109–112, 115–118, 130, 135, 136, 139, 159, 160

PPO proximal policy optimization 28, 29, 32

PSF predictive safety filter 154

REINFORCE a policy-based reinforcement learning method 28–31

ReLU rectified linear unit 92, 144

RGB red, green, and blue color channels 12, 41–43, 45, 48, 51, 53, 54, 56, 63, 88

RGB-D red, green, blue, and depth channels 12, 53, 54

RL reinforcement learning iii, 5, 24, 25, 28–33, 35, 37, 103, 104, 106, 108, 115, 117, 134, 136, 139, 140, 147–151, 153, 154, 156, 157

ROI region of interest 89–91, 100

ROS robot operating system 43, 80, 81, 94

S2R-ObjDet our sim2real transfer learning method for object detection ii, iii, 6, 8, 40, 42, 43, 47, 58, 64, 65, 81–83, 88, 90, 91, 97, 99–101, 138, 140

S2R-PosEst our sim2real transfer learning method for pose estimation ii, iii, xxix, 6, 8, 86, 88, 90, 97, 99, 101, 138, 140

SAC soft actor-critic 28, 29, 32, 33, 104, 117, 130, 135

SARSA state–action–reward–state–action reinforcement learning method 28, 29

SCMDP state-wise constrained Markov decision process 152, 153

SGD stochastic gradient descent 145

SQRL safety Q-functions for reinforcement learning xvi, 153

STD standard deviation 97, 98, 126, 127, 131, 133, 134

TD temporal difference xxx, 30, 106, 107, 111, 112, 157

TD3 twin delayed deep deterministic policy gradient 28, 29, 32, 33, 104, 117, 130, 134, 135

TL transfer learning 4, 22, 23

UAV unmanned aerial vehicles 46, 48

UCB upper confidence bound 28

VAE variational autoencoder 23

ViT vision transformer 15, 16, 57, 91

VLM visual language model 140

XOR exclusive OR 144

ZST zero-shot transfer 65–68, 71, 74, 77, 79

Summary of Notation

Notation	Description
	General
$\mathbb{R}$	set of real numbers
$\mathbb{R}^+$	set of positive real numbers
$\mathbb{N}$	set of natural numbers; $\mathbb{N} = \{0, 1, 2, 3, \ldots\}$
$\mathbb{Z}$	set of integer numbers
$\mathbb{Z}^+$	set of positive integer numbers
$\doteq$	equality relationship that is true by definition
$\approx$	approximately equal
$\leftarrow$	assignment
$\emptyset$	empty set
$\in$	is an element of
$\subset$	subset of
$(a, b]$	the real interval between a and b including b but not including a
$\lvert S \rvert$	number of elements in set $\mathcal{S}$
$\cup$	union of sets
$\cap$	intersection of sets
$\mathbf{1}_{A=B}$	indicator function that evaluates whether the condition $A = B$ is true or false
$f : \mathcal{X} \to \mathcal{Y}$	function f from elements of set $\mathcal{X}$ to elements of set $\mathcal{Y}$
$\lfloor a \rfloor$	mathematical floor function on a, e.g., $\lfloor 1.6 \rfloor = 1$; $\lfloor \cdot \rfloor : \mathbb{R} \to \mathbb{Z}$
$P(X = x)$	probability that a random variable X takes on the value x

$X \sim p$	random variable X selected from distribution $p(x) \doteq P(X = x)$
$\mathbb{E}[X]$	expectation of a random variable X, i.e., $\mathbb{E}[X] \doteq \sum_x p(x)x$
$\arg\max_x f(x)$	value of x where $f(x)$ takes its maximal value
$F(\tau)$	tail distribution function at τ, $F(\tau) = P(X > \tau)$
$d(a, b)$	distance between a and b
$\mathcal{L}$	loss function
$\mathcal{U}(a, b)$	uniform distribution between a and b
$n_{\boldsymbol{r}}$	length of a vector $\boldsymbol{r}$; $n_{\boldsymbol{r}} \in \mathbb{N}$
$\overline{\boldsymbol{r}}$	arithmetic mean of vector $\boldsymbol{r}$
	Machine learning
$n_{\boldsymbol{x}}$	length of the input vector; $n_{\boldsymbol{x}} \in \mathbb{N}$
$n_{\boldsymbol{y}}$	length of the output vector; $n_{\boldsymbol{y}} \in \mathbb{N}$
$n_{\boldsymbol{w}}$	length of the weight vector; $n_{\boldsymbol{w}} \in \mathbb{N}$
$\boldsymbol{x}$	input vector; $\boldsymbol{x} \in \mathbb{R}^{n_{\boldsymbol{x}}}$
$\boldsymbol{y}$	output vector; $\boldsymbol{y} \in \mathbb{R}^{n_{\boldsymbol{y}}}$
$\boldsymbol{w}$	weight vector; $\boldsymbol{w} \in \mathbb{R}^{n_{\boldsymbol{w}}}$
	Computer vision
c_{ch}	number of channels of an image; $c_{ch} \in \mathbb{N}$
w, h	width and height of an image in pixels; $w, h \in \mathbb{Z}^+$
C	number of classes; $C \in \mathbb{Z}^+$
$\boldsymbol{b}_i$	bounding box ($\boldsymbol{b}_i = [x_i, y_i, w_i, h_i]$) of the i^{th} detection; $\boldsymbol{b}_i \in [0, 1]^4$
x_i, y_i	normalized x and y center coordinates of the bounding box of the of the i^{th} detection; $x_i, y_i \in [0, 1]$
w_i, h_i	normalized width and height of the bounding box of the i^{th} detection; $w_i, h_i \in [0, 1]$
c_i^{class}	class label of the i^{th} detection; $c_i^{\text{class}} \in \mathbb{N}$
p_i^{con}	confidence score of the i^{th} detection; $p_i^{\text{con}} \in [0, 1]$
τ_{con}	confidence threshold; $\tau_{\text{con}} \in [0, 1]$
τ_{iou}	IoU threshold; $\tau_{\text{iou}} \in [0, 1]$
$K * I$	convolution between K and I
$\mathcal{P}$	set of a prediction item, e.g., area of the bounding box of a prediction
$\mathcal{G}_T$	set of a ground truth item, e.g., area of the bounding box of a ground truth
$S^{\mathcal{P},\mathcal{G}_T}$	closest convex shape (e.g., rectangle) enclosing both $\mathcal{P}$ and $\mathcal{G}$

$J_{\text{IoU}}(\mathcal{P}, \mathcal{G}_T)$	intersection over union (IoU) or Jaccard index, $J_{\text{IoU}}(\mathcal{P}, \mathcal{G}_T) = \frac{\lvert\mathcal{P} \cap \mathcal{G}_T\rvert}{\lvert\mathcal{P} \cup \mathcal{G}_T\rvert}$; $J_{\text{IoU}}(\mathcal{P}, \mathcal{G}_T) \in [0, 1]$
$J_{\text{GIoU}}(\mathcal{P}, \mathcal{G}_T)$	generalized intersection over union ($GIoU$), $J_{\text{GIoU}} = J_{\text{IoU}} - \frac{\lvert S^{\mathcal{P}, \mathcal{G}_T} \setminus (\mathcal{P} \cup \mathcal{G}_T)\rvert}{\lvert S^{\mathcal{P}, \mathcal{G}_T}\rvert}$; $J_{\text{GIoU}}(\mathcal{P}, \mathcal{G}_T) \in [0, 1]$
$\mathcal{L}_{\text{IoU}}$	IoU loss function; $\mathcal{L}_{\text{IoU}} = 1 - J_{\text{IoU}}$, $\mathcal{L}_{\text{IoU}} : \mathcal{P} \times \mathcal{G} \rightarrow [0, 1]$
$\boldsymbol{D}$	confusion matrix; $\boldsymbol{D} \in \mathbb{N}^{C \times C}$
$\boldsymbol{D}^{\text{gen}}$	generalized confusion matrix; $\boldsymbol{D}^{\text{gen}} \in \mathbb{N}^{C+1 \times C+1}$
$n_{\boldsymbol{v}}$	length of the feature vector; $n_{\boldsymbol{v}} \in \mathbb{N}$
$\boldsymbol{v}$	feature vector; $\boldsymbol{v} \in \mathbb{R}^{n_{\boldsymbol{v}}}$
	Transfer learning
$\mathcal{X}$	feature space
$\mathcal{Y}$	label space
$P(X)$	marginal probability distribution where $X = [x_1, x_2, x_3, ..., x_n] \in \mathcal{X}$
$f(\cdot)$	predictive function; $f(\cdot) = P(Y \mid X)$
$\mathcal{D}$	domain; $\mathcal{D} = \{\mathcal{X}, P(X)\}$
$\mathcal{T}$	task; $\mathcal{T} = \{\mathcal{Y}, P(Y \mid X)\}$
$\{\mathcal{D}_S, \mathcal{T}_S\}$	source domain-task pair
$\{\mathcal{D}_T, \mathcal{T}_T\}$	target domain-task pair
n_{obj}	number of objects; $n_{\text{obj}} \in \mathbb{N}$
n_{grid}	grid size; $n_{\text{grid}} \in \mathbb{Z}^+$
d_{space}	grid spacing; $d_{\text{space}} \in \mathbb{R}^+$
d_{height}	position of the grid (initial object position) in z direction; $d_{\text{height}} \in \mathbb{R}^+$
$\boldsymbol{r}^{\text{grid}}$	grid position vector before the disturbance in x, y, and z direction; $\boldsymbol{r}^{\text{grid}} \in \mathbb{R}^3$
$\boldsymbol{\epsilon}^{\text{pos}}$	normalized translation disturbance vector, the limits of the magnitude of translations in the x, y, and z directions; $\boldsymbol{\epsilon}^{\text{pos}} \in [0, 1]^3$
$\boldsymbol{E}^{\text{rot}}$	rotation disturbance vector, the bounds of rotations in the x, y, and z directions; $\boldsymbol{E}^{\text{rot}} \in \mathbb{R}^{3 \times 2}$
$\boldsymbol{r}^{\text{obj}}$	pose of an object; $\boldsymbol{r}^{\text{obj}} = [r_x^{\text{obj}}, r_y^{\text{obj}}, r_z^{\text{obj}}, r_{rx}^{\text{obj}}, r_{ry}^{\text{obj}}, r_{rz}^{\text{obj}}]$
p_{texture}	probability of random texture; $p_{\text{texture}} \in \mathbb{R}$
$\boldsymbol{p}^{\text{objects}}$	vector of object selection probabilities; $\boldsymbol{p}^{\text{objects}} \in \mathbb{R}^{n_{obj}+2}$
$\boldsymbol{R}^{\text{cam}}$	matrix describing the camera pose; $\boldsymbol{R}^{\text{cam}} \in \mathbb{R}^{3 \times 2}$
$\boldsymbol{R}^{\text{target}}$	matrix describing the camera target position; $\boldsymbol{R}^{\text{target}} \in \mathbb{R}^{3 \times 2}$

$\boldsymbol{m}^{\text{width}}$	vector containing the lower and upper bound of image width in pixels; $\boldsymbol{m}^{\text{width}} \in \mathbb{N}^2$
$\boldsymbol{m}^{\text{height}}$	vector containing the lower and upper bound of image height in pixels; $\boldsymbol{m}^{\text{height}} \in \mathbb{N}^2$
θ_{FOV}	field of view of the camera; $\theta_{\text{FOV}} \in \mathbb{R}$
m_{width}	width of the image in pixels; $m_{\text{width}} \in \mathbb{N}$
m_{height}	height of the image in pixels; $m_{\text{height}} \in \mathbb{N}$
m_{type}	type of image (RGB, depth, RGB-D); $m_{\text{type}} \in \{0, 1, 2\}$
G_{ML}	performance gap of the model between the train and the validation sets; $G_{\text{ML}} \in [0, 1]$
G_{reality}	reality gap; $G_{\text{reality}} \in [0, 1]$
θ_i	orientation angle of the i^{th} detection computed by the pose estimation method, before fine-tuning; $\theta_i \in [-\pi, \pi]$
θ_i^*	orientation angle of the i^{th} detection after fine-tuning; $\theta_i^* \in [-\pi, \pi]$
S_{θ_i}	$\sin(\theta_i)$; $S_{\theta_i} \in [-1, 1]$
C_{θ_i}	$\cos(\theta_i)$; $C_{\theta_i} \in [-1, 1]$
β_{res}	resolution of rotation in the S2R-PosEst method; $\beta_{\text{res}} \in \mathbb{R}$
n_{rot}	number of rotations in the S2R-PosEst method; $n_{\text{rot}} \in \mathbb{N}$
	Reinforcement learning
$\mathcal{S}$	state space, set of all states
$\mathcal{A}$	action space, set of all actions
$A(s)$	set of all actions available in state s
$\mathcal{R}$	set of all possible rewards, a finite subset of $\mathbb{R}$
$\mathcal{G}$	set of all possible goals
s, s'	states; $s, s' \in \mathcal{S}$
a	action; $a \in \mathcal{A}$
r	reward; $r \in \mathcal{R}$
γ	discount-factor parameter; $\gamma \in [0, 1]$
μ_0	initial state distribution
t	discrete time step; $t \in \mathbb{N}$
T	final time step of an episode; $T \in \mathbb{N}$
A_t	action at time t; $A_t \in \mathcal{A}$
S_t	state at time t; $S_t \in \mathcal{S}$
R_t	reward at time t; $R_t \in \mathcal{R}$
g	goal; $g \in \mathcal{G}$
π	policy (decision-making rule)
π_B	behaviour policy

π_T	target policy
$\pi(s)$	action taken in state s under deterministic policy π
$\pi(a \mid s)$	probability of taking action a in state s under stochastic policy π
G_t	return following time t; $G_t \in \mathbb{R}$
G^j	return following time 0 of the j^{th} episode; $G^j \in \mathbb{R}$
h	time horizon; $h \in \mathbb{N}$
$p(s', r \mid s, a)$	probability of transition to state s' with reward r, from state s and action a
$p(s' \mid s, a)$	probability of transition to state s', from state s taking action a
$r(s, a)$	expected immediate reward from state s after action a; $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$
$r(s, a, s')$	expected immediate reward on transition from s to s' under action a; $r : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$
$v_\pi(s)$	value of state s under policy π (expected return); $v : \mathcal{S} \to \mathbb{R}$
$v_*(s)$	value of state s under the optimal policy; $v_* : \mathcal{S} \to \mathbb{R}$
V, V_t	estimates of state-value function v_π or v_*
$q_\pi(s, a)$	value of taking action a in state s under policy π; $q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$
$q_*(s, a)$	value of taking action a in state s under the optimal policy; $q_* : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$
Q, Q_t	estimates of state-value function q_π or q_*
r^g	reward function parameterised by the goal $g \in \mathcal{G}$
$\mathcal{S}^g$	set of goal states $\mathcal{S}^g \subset \mathcal{S}$
$\mathcal{S}^w,\ \mathcal{S}^d$	States where the result is a win or a draw; $\mathcal{S}^w \subset \mathcal{S}$ and $\mathcal{S}^d \subset \mathcal{S}$
$\mathcal{S}^f$	failure states; $\mathcal{S}^f \subset \mathcal{S}$
δ_t	temporal-difference (TD) error at t (a random variable); $\delta_t \in \mathbb{R}$
d	done flag; $d \in \{0, 1\}$
$\mathcal{B}_{\text{er}}$	experience replay
$\theta,\ \phi$	the weights of the actor and the critic models
$\mathcal{E}$	episode
	Curriculum learning
$\mathcal{B}_{\text{ser}}$	standard experience replay
$\mathcal{B}_{\text{hier}}$	highlight experience replay
$\mathcal{H}(\mu_0)$	entropy of the initial state-goal distribution

$\mathcal{H}(\mathcal{S}\|\mathcal{G})$	entropy of the goal-conditioned visited states
j	episode index; $j \in \mathbb{N}$
k	weight update index; $k \in \mathbb{N}$
λ	threshold for the highlight experience replay; $\lambda \in \mathbb{R}$
ξ	sampling ratio between $\mathcal{B}_{\text{ser}}$ and $\mathcal{B}_{\text{hier}}$ for weight update; $\xi \in [0, 1]$
n_{batch}	batch size for weight update; $n_{\text{batch}} \in \mathbb{N}$
n_{hier}	batch size of the highlight experience replay for weight update; $n_{\text{hier}} \in \mathbb{N}$
$\mathcal{D}_{\text{ser}}$	batch of the data from the standard experience replay for weight update
$\mathcal{D}_{\text{hier}}$	batch of the data from the highlight experience replay for weight update
$\mathcal{D}$	batch of data for weight update; $\mathcal{D} \leftarrow \mathcal{D}_{\text{ser}} + \mathcal{D}_{\text{hier}}$
$\mathcal{L}_{\text{hier}},\ \mathcal{L}_{\text{ser}}$	TD errors of the training batches $\mathcal{D}_{\text{hier}}$ and $\mathcal{D}_{\text{ser}}$; $\mathcal{L}_{\text{hier}},\ \mathcal{L}_{\text{ser}} \in \mathbb{R}$
α_p	parameter of prioritization of the highlight experience replay; $\alpha_p \in [0, 1]$
c	parameter of easy2hard initial state entropy method which controls the initial state entropy; $c \in [0, 1]$
$\mu_0(c)$	initial state distribution as a function of c
δ_{step}	step size in the `self-paced` and the `control` E2H-ISE methods; $\delta_{\text{step}} \in [0, 1]$
$\boldsymbol{\nu}^{\text{train}}$	sequence containing the training success rates; $\boldsymbol{\nu}^{\text{train}} = [\nu_1^{\text{train}}, \nu_2^{\text{train}}, \ldots, \nu_{n_{\text{train}}}^{\text{train}}] \in [0, 1]^{n_{\text{train}}}$
$\boldsymbol{\nu}^{\text{eval}}$	sequence containing the evaluation success rates; $\boldsymbol{\nu}^{\text{eval}} = [\nu_1^{\text{eval}}, \nu_2^{\text{eval}}, \ldots, \nu_{n_{\text{eval}}}^{\text{eval}}] \in [0, 1]^{n_{\text{eval}}}$
$\Psi_{\text{high}},\ \Psi_{\text{low}}$	threshold values in the `self-paced` E2H-ISE method; $\Psi_{\text{high}},\ \Psi_{\text{low}} \in [0, 1]$
ψ	target success rate in the `control` E2H-ISE method; $\psi \in [0, 1]$
ψ_{max}	maximum value allowed for ψ in the `control adaptive` E2H-ISE method; $\psi_{\text{max}} \in [0, 1]$
Δ	constant shift in the `control adaptive` E2H-ISE method; $\Delta \in [0, 1]$

Chapter 1

Introduction

Contents

1.1 Context: Adaptive robots 2
1.2 Problem statement: Transferability and universality 4
1.3 Contribution 5
1.4 Outline 8

1.1 Context: Adaptive robots

Humans learnt to create and utilise tools over two million years ago [9]. Until the first industrial revolution around 250 years ago, production was based on human and animal labour. Since then, machines have increasingly taken on a larger role in the production process [10]. Until the past decades, machines lifted heavy pieces following rule-based logic for the automotive industry, precisely implanted tiny parts into circuit boards for the electronics industry, or performed similar processes requiring little intelligence or adaptivity but high accuracy, speed, and robustness. These tasks are typically described as easy for machines and hard for humans.

Moravec's paradox: "It is comparatively easy to make computers exhibit adult level performance on intelligence tests or playing checkers, and difficult or impossible to give them the skills of a one-year-old when it comes to perception and mobility" [11]

As the works on traditional, rule-based manufacturing applications saturated, the interest of the scientific community shifted towards tasks that require some sort of adaptive behaviour or intelligence, primarily in the field of perception and mobility. Oftentimes, the environment is unstructured and dynamic. For this reason, robots cannot simply execute a pre-computed path and repeat it many times. They need to perceive and understand their environment and act according to their goals. Robots started picking and sorting different objects based on their vision algorithms, precisely assembling parts aided by their force sensors, collaborating with human workers by predicting their intentions, and autonomously transporting materials, goods, or humans inside factories and cities. Robots are intelligent machines that can *sense*, *think*[1], and *act*. [12].

In terms of human perception, vision is one of the most essential sensory inputs as it is inherently high bandwidth. An average 4-year-old has seen 50 times more data than the state-of-the-art *large language models* (LLMs) [13]. As a consequence, vision sensors such as cameras and lidars are widely used in robotics. Nevertheless, humans rely on other senses as well to perform specific tasks. For assembling fine parts, tactile or force feedback is essential, while for tuning a musical instrument, the primary feedback mechanism is auditory. Thus, robots are equipped with different types of sensors according to the tasks they need to execute.

Regardless of the type of sensor, the raw sensory data must be interpreted, meaningful information extracted, and the subsequent action determined—in essence, the robot

[1]It is essential to clarify that *thinking*, in this context, does not imply possessing consciousness or self-awareness; rather, it refers to the ability to make rational decisions based on the circumstances presented. For further information, refer to Appendix A.

needs to *think*. The robot's decision-making process is guided by a model which can be categorised as either rule-based or data-driven. Rule-based models are created by human experts who define specific rules, whereas data-driven models learn functions or behaviours through example-based learning. Data-driven methods fall under the domain of *machine learning* (ML). It is important to note that data-driven algorithms have specific human-designed structures that introduce *inductive bias* through the set of assumptions embodied in the given constraints. Models with more inductive bias fall closer to rule-based solutions than ones with less.

Deep learning (DL) is a subset of ML using deep artificial neural networks to find complex patterns in data, replacing hand-engineered feature selection with data-driven feature extraction. As a result, the DL approach reduces the inductive bias at the expense of requiring a substantially larger amount of data. Revolutionising numerous fields, deep learning is the flagship of the novel phenomenon commonly referred to as the AI revolution. In healthcare, the novel algorithms based on DL significantly improved the accuracy of image analysis in radiology [14]–[16], opened a whole new chapter in drug discovery [17], [18], and was applied to unravel the secrets of human aging [19]. In psychology, DL has demonstrated superior performance in mental health outcome research [20], whether applied to clinical data [21], genomics samples [22], vocal and visual expressions [23], or social media content [24]. In finance and banking, DL models are applied across various domains [25], [26], including stock market prediction [27], portfolio management [28], or fraud detection [29]. In social sciences DL can be utilised for various use cases, such as classifying social graphs [30] or measuring social inequalities [31]. Furthermore, now generative AI can create music, text, images, and even videos, pushing the boundaries of creative industries [32].

The deep learning revolution has dramatically transformed robotics and related fields as well. Methods based on deep learning outperformed then state-of-the-art rule-based or shallow machine learning approaches in image classification [33], object detection [34], [35], image segmentation [36], [37], natural language processing [38]–[41], point cloud and 3D modeling [42]–[48], robotic manipulation [49]–[52], localization [53], trajectory prediction of self-driving cars [54], [55], autonomous drifting [56], traffic forecasting [57], human gestion recognition [58] or intention prediction [59], human-robot collaboration [60], dog behaviour modelling for mobile robots [61], health monitoring [62], multi agent environment exploration [63], and several other domains. Nonetheless, as in other fields, there are specific challenges that need to be addressed if we want to unleash the full potential of DL algorithms and create truly adaptive intelligent robots [64]. At a high level, this thesis focuses on how to overcome some of these obstacles.

1.2 Problem statement: Transferability and universality

One of the most crucial challenges of robotics is that the robotic applications are *task-*, *robot-*, and *domain-specific*, meaning that if any element of the task-robot-domain triplet is changed, the carefully engineered or learnt solution may not work anymore. A typical manifestation of this phenomenon is when a robotic agent learns to perform a specific task but by lacking generalization skills, it fails to execute a slightly modified version of the task (*differences in tasks*). Another case is when a model works perfectly on one robot configuration but completely fails on another (*differences in robots*). Finally, a model might work in the lab or simulation settings but fail in the real world (*differences in domains*). *Transferability* in robotics means transferring knowledge between tasks [65], robots [66], or domains [35], [1], [4], [67] and falls under the field of *transfer learning* (TL) [68], [69]. As an example, DL models work well on the immense online datasets [70], [71], nevertheless, constructing such datasets for every real-world problem is not feasible. Transfer learning approaches might attempt to reuse once-learnt knowledge or extract relevant knowledge from more accessible data, e.g., synthetic images, or general real-world robotic datasets [72]. Transferability is essential for achieving sufficient scale and robustness of robotic applications in the industry or our everyday lives.

Emphasising the importance and challenges of scale and robustness, consider a real-world scenario where a self-driving car is trained on data from European roads. For safety concerns and economic feasibility, it is required that the car drives safely and reliably outside of Europe where the traffic signs and even the rules of traffic could be fundamentally different. While the adaptation to the altered environment does not pose a serious challenge to human drivers, it is considerably more difficult for machine-learning-based algorithms.

Industrial prototyping is another domain where transferability is key. Introducing a new product at a high-volume car manufacturer presents a significant challenge. Stopping the assembly lines for time-consuming training and testing can cause significant downtime which is economically infeasible. Consequently, engineers utilise virtual environments to reduce development time in the real environment. Thus, efficiently transferring knowledge from simulation to the real world – bridging the *reality gap* – is essential in this attempt.

The endeavour for adaptive robots is coupled not only with transferability but *universality* as well. Besides connecting specific tasks, robots, or domains, it is also important to create fundamentally universal algorithms and methods with a general training framework that can be applied to a wide range of robotics problems. In an ideal scenario, generic models can learn to control arbitrary robots to solve various tasks in any given environment

(domain) without changing the algorithm or the training process. *Reinforcement learning* (RL) [73]–[75] inspired by human/animal learning attempts precisely that by learning from trial-and-error through interactions with the environment. It is important to note that transferability and universality are not completely separate concepts as by definition, universal solutions are easily transferable to other tasks, robots, or domains. Furthermore, it is worth mentioning that the aforementioned line of research still falls under the field of narrow AI, and is not equivalent to the academic pursuit of *artificial general intelligence* (AGI)[2].

Even though reinforcement learning achieved superhuman performance in other domains such as playing chess [77], Go [78], or Atari games [79], the field of robotics poses significant challenges as the state and action spaces are continuous, the reward function is predominantly sparse, and on many occasions, the agent is devoid of expert demonstrations. These characteristics exacerbate the problem of exploration, leading the robot to struggle in finding a solution to its given task [2].

In parallel to constructing more efficient RL algorithms such as state-of-the-art actor-critic models [80]–[83], another line of research focuses on improving existing RL algorithms with methods based on *curriculum learning* (CL) [2], [84]–[95]. The core idea of curriculum learning [96]–[98] is that ML models – similarly to human education – might benefit from an organised training process, e.g., starting with easy tasks (learning addition and multiplication) and gradually increase the difficulty up to the target (solving differential equations).

Furthermore, as RL algorithms learn from trial-and-error, meaning that they not only require ample data (as fast as possible) but also a safe environment where the robot cannot harm itself or its environment, especially at the early stage of the learning process[3]. For these reasons, training RL algorithms in simulation is paramount[4]. However, the models must be applied in the physical environment eventually, thus transferring the knowledge from simulation to the real world is essential once again.

1.3 Contribution

This PhD has been conducted at the Faculty of Informatics at Eötvös Loránd University, Budapest, Hungary in collaboration with the Institute for Computer Science and Control, Hungarian Research Network, Budapest, Hungary, and also, in a doctoral internship, at

[2]Also referred to as AMI or *autonomous machine intelligence* [76].
[3]At the early stage of the training, the movement of the robots is close to random.
[4]There is a line of research on training robots only in real life [99]–[102].

the Centre for Robotics at the École Nationale Supérieure des Mines de Paris, University Paris Sciences & Lettres in the framework of "Campus France Bourse du Gouvernement Français – Bourse Excellence Hongrie", funded by the Government of France. Although the results presented below are claimed to be mine, I use plural wording in the text for stylistic purposes.

In this thesis, we present our findings on data-driven adaptive robotic systems from the aspect of transfer and curriculum learning. First, we investigate how to transfer knowledge from simulation to the real world focusing on visual data. Bridging the reality gap, we introduced a sim2real transfer learning method based on domain randomization for object detection (S2R-ObjDet), a generalised confusion matrix (GCM) for performance evaluation, and a public and annotated real-world dataset of industrial objects (InO-10-190). In addition, we propose two vision-based, multi-object grasp pose estimation models (MOGPE) – the real-time (MOGPE-RT) and the high-precision (MOGPE-HP) – with the augmentation of our S2R-ObjDet method with pose estimation (S2R-PosEst). Our framework provides an industrial tool for fast data generation and model training and requires minimal data from the target distribution. Besides testing the computer vision components individually, our methods were also validated in a robotic pick-and-place experiment.

Furthermore, we examined how to facilitate the training of reinforcement learning agents with curriculum learning. We propose a novel data exploitation curriculum learning method, named the highlight experience replay (HiER). Additionally, to exploit the full potential of HiER, we introduce HiER+ in which HiER is enhanced with an arbitrary data collection curriculum learning method. In addition, as an example of the data collection curriculum learning method, we present the easy2hard initial state entropy method (E2H-ISE). Our methods significantly outperformed state-of-the-art, validated on 8 tasks of three robotic benchmarks, focusing on the push, slide, and pick-and-place tasks.

The scope of this work can be defined through the following research questions:

- How to transfer knowledge from simulation to the real world in the case of object detection? [1]
- How to extend our S2R-ObjDet method to multi-object grasp pose estimation? [4]
- How to improve the training process of state-of-the-art reinforcement learning algorithms with curriculum learning? [2]

The main publications of this thesis in chronological order are as follows:

- **D. Horváth**, G. Erdős, Z. Istenes, T. Horváth, and S. Földi, 'Object Detection Using Sim2Real Domain Randomization for Robotic Applications', *IEEE Transactions on Robotics*, vol. 39, no. 2, pp. 1225–1243, Apr. 2023, ISSN: 1941-0468. DOI: 10.1109/TRO.2022.3207619.

- **D. Horváth**, K. Bocsi, G. Erdős, and Z. Istenes, 'Sim2Real Grasp Pose Estimation for Adaptive Robotic Applications', in *the 22nd IFAC World Congress*, ser. IFACPapersOnLine, vol. 56, 2023, pp. 5233–5239. DOI: 10.1016/j.ifacol.2023.10.121.

- **D. Horváth**, J. Bujalance Martín, F. Gábor Erdos, Z. Istenes, and F. Moutarde, 'HiER: Highlight Experience Replay for Boosting Off-Policy Reinforcement Learning Agents', *IEEE Access*, vol. 12, pp. 100 102–100 119, Jul. 2024, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2024.3427012.

The primary research tasks associated with the main articles [1], [4], [2] – including formulating research questions, proposing solutions, developing and implementing algorithms, designing and executing experiments, analysing and visualising data, and drafting the manuscripts – were carried out by me. Generally, the co-authors contributed through supervision, providing periodic feedback and guidance that helped refine the research focus and presentation. Additionally, Sándor Földi assisted in the data annotation and the experimentation for [1], while Kristóf Bocsi helped in the experimentation, the robot control, the camera calibration, and the implementation of the pattern-matching algorithm for [4].

In additional publications, I developed and implemented algorithms for vision-based adaptive robotic applications, in the context of smart factories, listed chronologically, as follows:

- Z. Kemény, R. Beregi, J. Nacsa, C. Kardos, and **D. Horváth**, 'Human–robot collaboration in the MTA SZTAKI learning factory facility at Győr', in *the 8th CIRP Sponsored Conference on Learning Factories (CLF)*, ser. Procedia Manufacturing, vol. 23, Jan. 2018, pp. 105–110. DOI: 10.1016/j.promfg.2018.04.001.

- Z. Kemény, R. Beregi, J. Nacsa, C. Kardos, and **D. Horváth**, 'Example of a problem-to-course life cycle in layout and process planning at the MTA SZTAKI learning factories', in *the 9th Conference on Learning Factories (CLF)*, ser. Procedia Manufacturing, vol. 31, Jan. 2019, pp. 206–212. DOI: 10.1016/j.promfg.2019.03.033.

- M. Hajós and **D. Horváth**, 'Robotos pakolási feladat megoldása környezetérzékelés segítségével', in *Nemzetközi Gépészeti Konferencia (OGÉT)*, Apr. 2020, pp. 305–308. [Online]. Available: https://ojs.emt.ro/oget/article/view/156.

- G. Erdős, **D. Horváth**, and G. Horváth, 'Visual servo guided cyber-physical robotic assembly cell', in *the 17th IFAC Symposium on Information Control Problems in Manufacturing (INCOM)*, ser. IFAC-PapersOnLine, vol. 54, Jan. 2021, pp. 595–600. DOI: 10.1016/j.ifacol.2021.08.068.
- G. Erdős, K. Abai, R. Beregi, **et al.**, "Enabling Technologies for Autonomous Robotic Systems in Manufacturing," *Transactions of Nanjing University of Aeronautics and Astronautics*, vol. 41, no. 4, pp. 403–431, Aug. 2024, ISSN: 1005-1120. DOI: 10.16356/j.1005-1120.2024.04.001.

1.4 Outline

The thesis contains six chapters:

- **Chapter 1: Introduction.** We briefly present the context of adaptive robotics, the open challenges of transferability, and universality. Furthermore, list our contributions and outline the structure of this thesis.
- **Chapter 2: Theoretical background.** The essential background knowledge of the following chapters is outlined, including machine learning, computer vision, transfer learning, reinforcement learning, and curriculum learning. The literature review is detailed at the beginning of the specific chapters.
- **Chapter 3: Sim2real knowledge transfer for object detection.** Our domain-randomization-based sim2real knowledge transfer method for object detection (S2R-ObjDet) is presented with the literature review, our generalised confusion matrix (GCM), our industrial dataset (InO-10-190), our test results, and a thorough ablation study.
- **Chapter 4: Sim2real grasp pose estimation.** Our vision-based, multi-object grasp pose estimation models (MOGPE) are presented with the augmentation of our S2R-ObjDet method with pose estimation (S2R-PosEst), alongside the literature review and our experimental results.
- **Chapter 5: Highlight experience replay.** In this chapter, our curriculum learning methods (HiER, E2H-ISE, and HiER+) are presented along with the literature review and our experimental results.
- **Chapter 6: Conclusion.** Finally, we conclude our findings and give an outline for possible future research.

Chapter 2

Theoretical background

Contents

2.1 Computer vision 11
2.1.1 Problem formulation 11
2.1.2 Convolutional neural networks 13
2.1.3 Vision transformers 15
2.1.4 Image classification 16
2.1.5 Object detection and pose estimation 18
2.2 Transfer learning 22
2.2.1 Definitions and notations 22
2.2.2 Sim2real object detection 23
2.3 Reinforcement learning 24
2.3.1 Markov decision process 25
2.3.2 Multi-goal tasks 25
2.3.3 Reward functions 26
2.3.4 Demonstrations 28
2.3.5 Categorization of RL algorithms 28
2.3.6 Tabular reinforcement learning 30
2.3.7 Reinforcement learning in robotics 30
2.3.8 Evaluation of RL algorithms 33

2.4 Curriculum learning 34
2.4.1 Easy2hard curriculum learning 35
2.4.2 Generalised curriculum learning 36
2.4.3 CL in supervised learning 37
2.4.4 CL in reinforcement learning 37

In this chapter, the essential theoretical background is presented. For brevity, the introduction of artificial intelligence, machine learning, and deep learning with neural networks are placed in Appendix A. Furthermore, the main domains of this thesis are presented as follows: computer vision in Section 2.1, transfer learning in Section 2.2, reinforcement learning in Section 2.3, and curriculum learning in Section 2.4.

2.1 Computer vision

Computer vision (CV) is a field of AI which aims to extract meaningful information from visual data such as images or videos. It advanced rapidly in the past 10 years with the deep learning revolution and the improvement of *graphical process units* (GPUs). Vision is paramount when it comes to perception. With CV algorithms, we can recognise objects, track movements, analyse scenes, and aid the decision-making processes.

As *large language models* (LLMs) overwhelm scientific discourse today, it is important to emphasise the relevance of vision-based models in the domain of robotics and generally in AI. First and foremost, vision has a profound connection to the real world around us while language is essentially human-generated data inheriting the biases of the generator and the limitations of the language. Consider the difference between explaining and showing how to sit down, or what an elephant looks like. Additionally, for human perception, vision is a high-bandwidth input with around 20 MB/s compared to language with around 12 bytes/s. An average 4-year-old has seen 50 times more data than the state-of-the-art LLMs [13]. Nevertheless, the importance of LLMs should not be underestimated either. They are integral to robotics research, having already driven numerous breakthroughs, with several more anticipated in the future.

In this section, the problem formulation of CV and learning-based methods are presented. For a more thorough review, we refer the readers to [103], [104].

2.1.1 Problem formulation

Computer vision problems typically fall within the domain of supervised learning, where the goal is to determine the function f^* that maps the input to the output. Thus, the training dataset contains labelled input–output pairs, where the input is visual data – an image or images – and the output depends on the CV task. The labelled output is also referred to as the *ground truth* (GT).

A typical formulation of the input is: $\boldsymbol{x} \in \mathbb{R}^{w \times h \times c_{\text{ch}}}$, where $w, h \in \mathbb{N}$ are the width and height of the image in pixels, and $c_{\text{ch}} \in \mathbb{N}$ is the number of channels. In the case of

grayscale images or depth data, $c_{\text{ch}} = 1$, for RGB images, $c_{\text{ch}} = 3$, and for RGB-D images, $c_{\text{ch}} = 4$.

The output of the model depends on the specific problem. In the case of image classification, the output vector contains the probabilities that the input belongs to specific classes, $\boldsymbol{y} \in [0,1]^C$, where C is the number of classes. For traditional object detection, $\boldsymbol{y} = \{(\boldsymbol{b}_i, c_i^{\text{class}}, p_i^{\text{con}}) \mid i = 1, 2, \ldots, N\}$, where $\boldsymbol{b}_i = [x_i, y_i, w_i, h_i]$ represents the 2D *axis-aligned bounding box* (AABB)[1] of the i^{th} detection, where $x_i, y_i \in [0,1]$ are the normalised coordinates of the centre point, while $w_i, h_i \in [0,1]$ are the normalised width and height of the i^{th} detection. Furthermore, $c_i^{\text{class}} \in \mathbb{N}$ is the class label[2] of the i^{th} detection, $p_i^{\text{con}} \in [0,1]$ is the confidence score, and $N \in \mathbb{N}$ is the number of detected objects. In the case of semantic segmentation, the output is the pixel-wise classification of the image, $\boldsymbol{y} \in \mathbb{R}^{w \times h \times C}$. In this case, each pixel is associated with an output vector – with length C – representing the probabilities of that pixel belonging to the respective classes. Instance segmentation lies between object detection and semantic segmentation. Here, each individual object instance is segmented separately, even if multiple objects belong to the same class. It identifies both the class and the individual instance of each object. The comparison of the different CV tasks is depicted in Fig. 2.1.

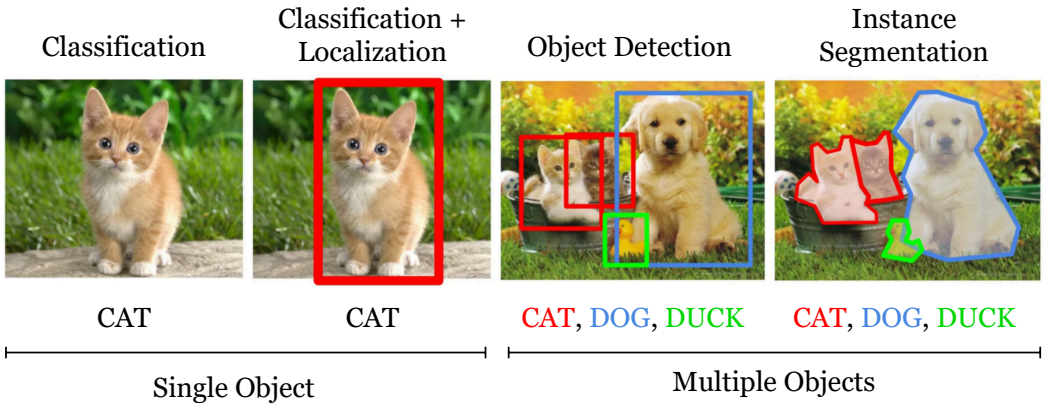


Figure 2.1: Comparison of classification, object detection, and instance segmentation from [105].

Since the input data is an image, it possesses a substantially high dimensionality. Consider a 3-channel 256×256 RGB image, the dimension of input $\boldsymbol{x}$ is $256 \times 256 \times 3 = 196608$.

[1] Alternative formulations are outlined in Section 2.1.5.
[2] Typically with one-hot encoding.

Thus, applying traditional feedforward *multilayer perception* (MLPs) with fully connected layers (see in Appendix A.3) is not feasible due to the memory requirements and training efficiency.

2.1.2 Convolutional neural networks

Convolutional neural networks (CNNs) are special *neural networks* (NN) that were first introduced by Lecun et al. [106] in 1989 but they gained widespread popularity only in 2012 with the success of AlexNet [33] at the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [107]. CNNs employ a mathematical operation known as *convolution* to extract features from the input data and are especially effective in tasks involving vision, audio, or time series data. A CNN can be either single or multi-dimensional, depending on the dimensionality of the convolution. The mathematical formulation for 2D convolution is provided in Eq. (2.1).

$$\boldsymbol{S}(i,j) = (\boldsymbol{K} * \boldsymbol{I})(i,j) = \sum_m \sum_n \boldsymbol{I}(i-m, j-n)\boldsymbol{K}(m,n), \tag{2.1}$$

where $\boldsymbol{S}$ represents the output image, $\boldsymbol{I}$ denotes the input image, and $\boldsymbol{K}$ is the convolution kernel (or filter). The indices i and j correspond to the global pixel positions in the input image, while m and n represent the local offset indices for the convolution operation.

Key characteristics of CNNs

Following [108], CNNs have three key ideas:

- **Sparse connectivity.** Compared to traditional fully connected MLPs – where all neurons of a layer are connected to all neurons of the previous layer – in convolutional layers, an output neuron is connected only to a small subset of the input. Sparse connectivity means fewer parameters and thus reduced memory usage, and statistical efficiency.
- **Parameter sharing.** The output is not only connected to a small subset of the input but the weights of these connections are shared across each output neuron, meaning that the same set of weights is reused across different input regions. In other words, the output is computed by sliding a (learnable) kernel over the input. While this does not impact the runtime of forward propagation, it does help reduce the memory required.

- **Equivariant representations.** CNNs are equivariant to translation. It is an essential property which enables the model to find similar patterns in different parts of the image. In simple terms, this means that when an object or feature shifts in the input, it shifts in the output in the same manner. Mathematically, $f(x)$ is equivariant to a function g if $f(g(x)) = g(f(x))$. In our case, g is any function that translates the input.

Building blocks of CNNs

CNNs consist of three distinct operations:

- **Convolutional layer.** The convolution layer is the core part of the structure – hence the name. A (learnable) kernel is slid over the input executing the mathematical convolution operation – described in Eq. 2.1 – in order to extract the relevant features of the image.
- **Activation function.** After the convolution operation, a non-linear *activation function* is applied to the output neurons of the convolutional layer. This step introduces non-linearity, allowing the network to learn more complex patterns than simple linear relationships. For more detail, we refer the reader to Appendix A.3.
- **Pooling layer.** To reduce the spatial dimensions of the output image – also referred to as the *feature map* – the technique of *pooling* is utilised. In this operation, the pixels (neurons) of the feature map are replaced with summary statistics [108] – e.g., *max pooling* [109] replaces the target value with the maximum value of its rectangular neighbourhood. Besides dimensionality reduction, pooling makes the model invariant for small translations which is beneficial if the presence of the feature does not need to be (pixel-wise) precise. Boureau et al. [110] examined different types of pooling in various situations. It is important to note that in some cases, not every convolutional layer is followed by a pooling layer – depicted in Fig. 2.2.

In practice, several convolutions are performed in parallel creating numerous feature maps capturing separate relevant aspects of the input image, such as edges, shapes, or texture. The convolutional layers are illustrated in Fig. 2.2.

A Typical CNN architecture for image classification

In the case of classification, CNNs are typically divided into two consecutive parts – depicted in Fig. 2.2 – as follows:

- **Feature Extractor (body).** The *feature extractor* consists of several consecutive convolutional – in each layer applying numerous kernels – and pooling layers. As data passes through the layers, the network learns more abstract and high-level features. By the end, the input image is transformed into a compact *feature vector* $\boldsymbol{v} \in \mathbb{R}^{n_{\boldsymbol{v}}}$ containing the relevant information of the image, where $n_{\boldsymbol{v}} \in \mathbb{N}$ is the length of the feature vector. Mathematically, $f_{\text{body}} : \mathbb{R}^{w \times h \times c_{\text{ch}}} \to \mathbb{R}^{n_{\boldsymbol{v}}}$.
- **Classifier (head).** After feature extraction, the feature vector is passed to a fully connected MLP – known as the head – which maps the feature vector to the desired output – e.g., class probabilities. Mathematically, $f_{\text{head}} : \mathbb{R}^{n_{\boldsymbol{v}}} \to \mathbb{R}^{C}$.

The feature extractor is a component of most CNN models, regardless of the specific task. Its primary role is to extract relevant information from an image, which is the main purpose of computer vision itself. However, in tasks such as object detection or segmentation, the network's *head* differs from the one described above.

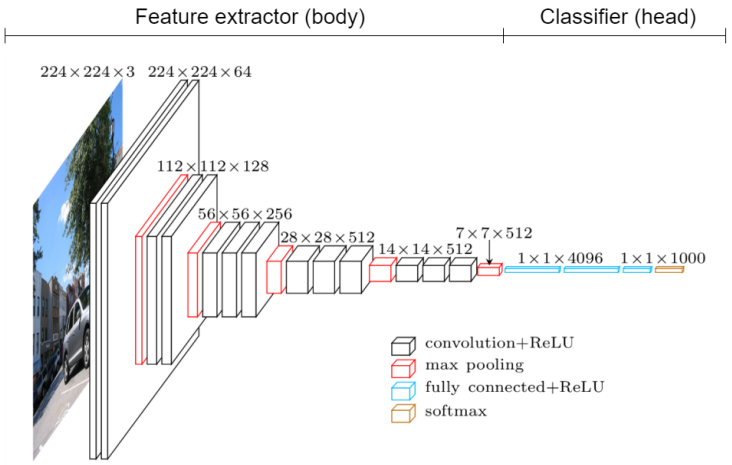


Figure 2.2: Architecture of VGG16 [111]: The feature extractor is composed of successive convolutional layers (with ReLU activations) and max pooling layers, while the classification head comprises fully connected layers with ReLU activations. The image has been adapted from [112].

2.1.3 Vision transformers

Vision transformers (ViTs) [113], [114] belong to another line of research, offering an alternative to CNNs. ViT is an adaptation of the novel *transformer* architecture [39].

Standard transformers were introduced for natural language processing where they process one-dimensional time series. By leveraging self-attention mechanisms, transformers effectively capture relationships between different parts of the input, regardless of their spatial locations, striking a balance between short-term and long-term dependencies. To handle multi-dimensional image data, ViT models divide images into tokens – small patches of $n \times n$ pixels – and process them as sequential data. While ViTs have shown promising results in computer vision, they typically require extensive pre-training on large datasets due to their lower inductive bias compared to CNNs. Since this thesis focuses on CNNs, for a more thorough review of ViTs, we refer the reader to [114].

2.1.4 Image classification

In this section, we provide an overview of image classification, followed by object detection and pose estimation in Section 2.1.5. Since image segmentation is not relevant to this work, it is not discussed here.

As introduced in Section 2.1.1, the input of the model is the image itself, $\boldsymbol{x} \in \mathbb{R}^{w \times h \times c_{\text{ch}}}$, while the output is the probabilities that the input belongs to specific classes, $\boldsymbol{y} \in [0,1]^C$. Thus, $f_{\text{class}} : \mathbb{R}^{w \times h \times c_{\text{ch}}} \to [0,1]^C$.

The output layer contains C neurons, one for each class. To convert *logits*[3] to probabilities, typically the softmax activation function is applied $\sigma : \mathbb{R}^C \to [0,1]^C$, presented in Eq. (2.2). Additionally, during training, the cross-entropy loss function is applied, defined in Eq. (2.3).

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{c=1}^{C} e^{z_c}}, \tag{2.2}$$

where $\mathbf{z} \in \mathbb{R}^C$ is the logit vector.

$$l_{\text{CE}} = -\sum_{c=1}^{C} y_{i,c} \cdot \log(p_{i,c}), \tag{2.3}$$

where $y_{i,c}$ indicates if the input i is in class c, while $p_{i,c}$ is the predicted probability that the input i is in the class c.

There are numerous metrics to evaluate classification models. The *confusion matrix* $\boldsymbol{D} \in \mathbb{N}^{C \times C}$ helps to navigate among these metrics, depicted in Fig. 2.3. $\boldsymbol{D}$ is a $C \times C$

[3]The raw, unnormalised output values of the final layer of the NN before any activation function is applied, typically before applying a softmax function.

matrix, where $D_{i,j}$ indicates the number of cases when an image of class i was classified as class j. Following that, the most common classification metrics are:

- **Accuracy.** The portion of correctly classified images, presented in Eq. (2.4). It is equivalent to the number of samples in the diagonal of the confusion matrix ($i = j$) divided by all the samples.

$$Accuracy = \frac{\sum_i D_{i,i}}{\sum_{i,j} D_{i,j}} \tag{2.4}$$

- **Top-k accuracy.** Top-k accuracy evaluates the model's performance by considering its top k predictions, rather than just checking if the top prediction is correct. A prediction is deemed correct if the true class appears within the top k predicted classes. This approach offers a more flexible measure of performance, particularly when distinguishing between classes is challenging[4]. Typically, k is set to one (traditional accuracy) or five.

- **Precision (positive predictive value).** The portion of correctly labelled examples among all the predictions of a class. Following the notation of the confusion matrix:

$$Precision(i) = \frac{D_{i,i}}{\sum_j D_{j,i}} \tag{2.5}$$

  It is beneficial to use the precision metric as an evaluation when the high quality of the prediction is more important than missing some samples, e.g., spam detection.

- **Recall (sensitivity).** The portion of correctly labelled examples of a given (ground truth) class. Following the notation of the confusion matrix:

$$Recall(i) = \frac{D_{i,i}}{\sum_j D_{i,j}} \tag{2.6}$$

  It is advised to use the recall metric as evaluation when detecting all the samples of a class is crucial, e.g., medical screening for diseases.

[4]E.g., in the ImageNet dataset, the 'Siberian Husky' and the 'Alaskan Malamute' are two different dog breeds that share many main characteristics.

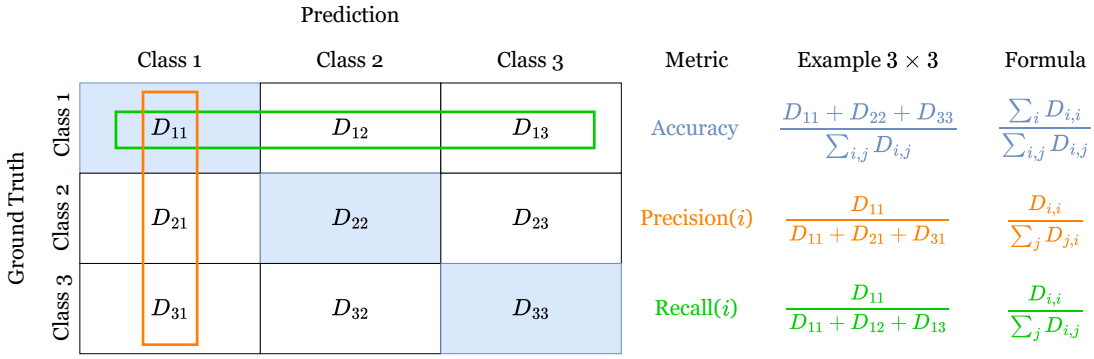


Figure 2.3: Confusion matrix.

2.1.5 Object detection and pose estimation

As state-of-the-art CNNs exceed human performance in most classification benchmarks, the research interest has shifted from classification towards more challenging tasks such as object detection or pose estimation.

The goal of object detection is to detect all objects of given classes in an image. Object detection is both a regression and a classification problem since both the locations of the objects and the classes of the objects need to be computed. Moreover, as the number of objects in an image can vary, the number of outputs is not predetermined – in contrast to classification tasks. It is important to note that object detection does not attempt to assign individual pixels to the given objects as it is done in semantic segmentation. Instead, it uses *bounding boxes* (BBs) to define the boundaries of the objects.

The objects are located on an image with BBs which are n-dimensional hyperrectangles (e.g., rectangles in 2D) encompassing the objects. If the axes of the hyperrectangles are aligned with the axes of the image's coordinate system, then the bounding boxes are axis-aligned, as depicted on the left side of Fig. 2.4. In classical object detection, the image is 2D and the BBs are axis-aligned (AABB). Mathematically, the output of a traditional object detection model is $\boldsymbol{y} = \{(\boldsymbol{b}_i, c_i^{\text{class}}, p_i^{\text{con}}) \mid i = 1, 2, \ldots, N\}$, where $\boldsymbol{b}_i = [x_i, y_i, w_i, h_i]$ represents the 2D AABB, $c_i^{\text{class}} \in \mathbb{N}$ is the class label, and p_i^{con} is the confidence score[5] of the i^{th} detection, while $N \in \mathbb{N}$ is the number of detected objects. Alternatively, a 2D AABB can also be formulated as $\boldsymbol{b}^{\text{2D}} = [x^{\min}, y^{\min}, x^{\max}, y^{\max}]$, where the items correspond to the minimum and maximum values of the rectangle. Consequently, in 3D, $\boldsymbol{b}^{\text{3D}} = [x^{\min}, y^{\min}, z^{\min}, x^{\max}, y^{\max}, z^{\max}]$.

[5]Note that for the ground truth, there is no $p_i^{\text{con}} \in [0, 1]$ confidence score.

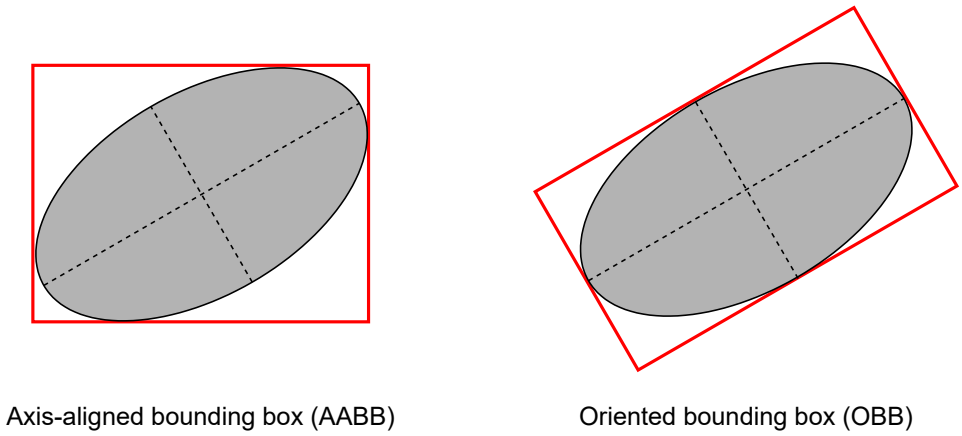


Figure 2.4: The comparison of axis-aligned bounding boxes and oriented bounding boxes.

For pose estimation, the classical axis-aligned bounding boxes (AABB) are not sufficient as they lack orientation information. Thus, for pose estimation, *oriented bounding boxes* (OBB) need to be used. In the 2D case, it means that another parameter $\phi \in \mathbb{R}$, describing the rotation, needs to be added to the object description. While OBBs are more complex and thus harder to train, they offer a more accurate representation of the object compared to AABBs. The comparison of AABB and OBB is depicted on Fig. 2.4.

Training and evaluation metrics

For training and evaluating, it is important to define what counts as correct detection, which is not as straightforward as in the case of classification. The object detection models output a set of BBs. First and foremost, the proposed bounding boxes are filtered based on their confidence score p_i^{con}. To keep specific predictions, their p_i^{con} must be greater or equal than the *confidence threshold* $\tau_{\text{con}} \in [0, 1]$. In the second step, the predicted BBs and the ground truth BBs need to be compared. Typically, the *intersection over union* (IoU) metric, also known as the Jaccard index, defined in Eq. (2.7), is applied. If the area of overlap divided by the area of union exceeds a certain threshold $\tau_{\text{iou}} \in [0, 1]$, then the prediction is considered to be correct. A typical threshold value is $\tau_{\text{iou}} = 0.5$. It is important to note that there are many objects in an image, and the IoU of each prediction–GT pair need to be evaluated.

$$J_{\text{IoU}}(\mathcal{P}, \mathcal{G}_T) = \frac{|\mathcal{P} \cap \mathcal{G}_T|}{|\mathcal{P} \cup \mathcal{G}_T|}, \tag{2.7}$$

where $\mathcal{P}$ is the predicted BB and $\mathcal{G}_T$ is the ground truth BB.

In terms of model training, the IoU metric can be applied as a loss function as well, $\mathcal{L}_{\text{IoU}} = 1 - J_{\text{IoU}}$. Nevertheless, IoU has a weakness. If $|\mathcal{P} \cap \mathcal{G}_T| = 0$, then $J_{\text{IoU}}(\mathcal{P}, \mathcal{G}_T) = 0$. In other words, if the prediction and the ground-truth BB do not overlap, then their IoU is zero. To overcome this, an alternative version of the standard IoU measure, the *generalised intersection over union* (GIoU) [115] is introduced, formulated as in Eq. (2.8). GIoU is a normalised measure minimising the empty area between $\mathcal{P}$ and $\mathcal{G}_T$.

$$J_{\text{GIoU}} = J_{\text{IoU}} - \frac{|S^{\mathcal{P},\mathcal{G}_T} \setminus (\mathcal{P} \cup \mathcal{G}_T)|}{|S^{\mathcal{P},\mathcal{G}_T}|}, \tag{2.8}$$

where $S^{\mathcal{P},\mathcal{G}_T}$ is the closest convex shape (in this case rectangle) enclosing both $\mathcal{P}$ and $\mathcal{G}_T$.

For evaluation, the *precision-recall* curve is essential. As presented above, the predicted BBs are filtered out based on the τ_{con} threshold. Thus, changing τ_{con} affects how many BBs are kept. If τ_{con} is set too high, only a few BBs will pass the threshold and be subject to the IoU-based GT comparison, resulting in high precision and low recall. On the other hand, setting τ_{con} to a low value will result in many BBs, resulting in a low precision but a high recall. This is the precision-recall trade-off.

In the precision-recall curve, data points of different τ_{con} threshold values (from 0 to 1) are presented forming a curve. For one class, the *average precision* (AP) score $\in [0, 1]$ represents the area under the curve. The *mean average precision* (mAP) score $\in [0, 1]$ represents the mean AP across the different classes. Another metric is the F_1 score $\in [0, 1]$ which is the harmonic mean of the precision and the recall. An example of the precision-recall curve with the maximum F_1 score is depicted in Fig. 2.5.

State-of-the-art object detection models

Recent deep CNN architectures can be categorised into two groups: two-stage detectors and one-stage detectors. Two-stage detectors have a proposal detection stage where a set of bounding box candidates is generated, and a verification stage where these bounding boxes are separately evaluated whether they contain an object of a specific class. Examples of these networks are R-CNN [117], SPPNet [118], Fast R-CNN [119], and Faster R-CNN [120].

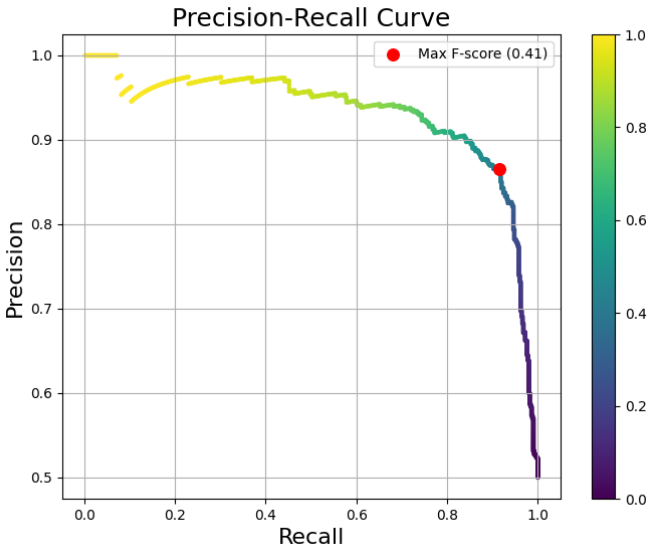


Figure 2.5: An example of the precision-recall curve from [116]. The colour bar indicates the τ_{con} threshold values.

In the case of one-stage detectors, on the other hand, a single neural network is applied to the full image that predicts the bounding boxes straight away. The slow detection time, which is the biggest disadvantage of the two-stage detectors, can be overcome with the one-stage approach. Detection time is crucial for many applications, especially but not exclusively in the field of robotics or self-driving cars. Redmon et al. proposed the first one-stage detector YOLO in 2015 [121], being the first real-time object detector. Subsequent updates introduced its second [122] and third versions [123]. Single Shot MultiBox Detector (SSD) [124] and RetinaNet [125] are two other popular one-stage detectors.

Bochkovskiy et al. [34] created YOLOv4 aiming to improve the accuracy of the model while still keeping an optimal accuracy–speed trade-off. With the CSPDarknet-53 [34] backbone, 65.7% mAP_{50} could be achieved for the MS COCO dataset [71] and around 65 FPS speed on a Tesla V100. Since then, further YOLO versions were introduced to gradually improve the performance: YOLOv5 [126], YOLOv6 [127], YOLOv7 [128], YOLOv8 [129], and YOLOv9 [130]. YOLOv5-X r7.0 [126] achieves 68.9%, while YOLOv9-E [130] yields 72.8% mAP_{50} score on the MS COCO dataset. In comparison, on the same dataset, SSD with VGG-16 [111] backbone performed 48.5% mAP_{50} and RetinaNet with ResNet-101 [131] backbone achieved 57.5% mAP_{50}. For further comparison, we refer the reader to [132].

2.2 Transfer learning

One of the biggest challenges of robotics is how to transfer knowledge between tasks [65], robots [66], or domains [35], [1], [4], [67] – contrary to learn everything from scratch. The aforementioned problems belong to the field of *transfer learning* (TL). In this work, we focus on transferring knowledge between different tasks and domains. Illustrating the importance of TL, consider an example of two people attempting to learn to play the piano. One is a guitarist with extensive music knowledge, while the other has no background in music. Expectedly, the former student will master the piano faster than the one without any knowledge to transfer [68].

However, transfer learning is not only about optimising how we use learnt knowledge. Frequently, it is the only way to handle real-life problems. The traditional assumption in ML is that the training data and the test data come from the same distribution. Nevertheless, in real-world situations, this is often infeasible. For instance, the training data might be unreasonably expensive, difficult to obtain, or simply not available. Thus, we need models that can learn from data sampled from an accessible domain and then applied in the target domain [69].

2.2.1 Definitions and notations

Following [68], [69], [133], a domain $\mathcal{D}$ is defined with a feature space $\mathcal{X}$ of all possible feature vectors and a marginal probability distribution $P(X)$, where $X = [x_1, x_2, x_3, ..., x_n] \in \mathcal{X}$, while a task $\mathcal{T}$ is defined with a label space $\mathcal{Y}$ and a predictive (or decision) function $f(\cdot)$ which is in the centre of our attention as it is the function that is expected to be learned from the sample data – e.g., in visual classification the input is the image, the output is the class label, while the predictive function maps the input to the output. From a probabilistic point of view, $f(\cdot)$ can be seen as a conditional probability $P(Y \mid X)$. Thus, a domain $\mathcal{D} = \{\mathcal{X}, P(X)\}$ and a task $\mathcal{T} = \{\mathcal{Y}, P(Y \mid X)\}$. Given a specific source domain and task pair $\{\mathcal{D}_S, \mathcal{T}_S\}$ and a specific target domain and task pair $\{\mathcal{D}_T, \mathcal{T}_T\}$, transfer learning can be defined as the process of increasing the performance of the target predictive function $f_T(\cdot)$ with the help of knowledge gained from $\{\mathcal{D}_S, \mathcal{T}_S\}$, where $\mathcal{D}_S \neq \mathcal{D}_T$ or $\mathcal{T}_S \neq \mathcal{T}_T$.

An example of $\mathcal{T}_S \neq \mathcal{T}_T$ is when an image classifier, trained on a large public dataset, is reused and altered to perform object detection on the same or a similar domain ($\mathcal{D}_S = \mathcal{D}_T$). In this case, the label spaces are different, and consequently, the conditional probability distributions of the inputs and the outputs $P(Y \mid X)$ are disparate as well. Nevertheless, the marginal distributions of the inputs $P(X)$ are equivalent or similar.

The other case of transfer learning is when $\mathcal{T}_S = \mathcal{T}_T$, but $\mathcal{D}_S \neq \mathcal{D}_T$, i.e., the tasks are the same, yet, the domains are different. In order to improve $f_T(\cdot)$, our aim is to extract the relevant (not domain-specific) knowledge from $\{\mathcal{D}_S, \mathcal{T}_S\}$. Thus, the model will perform well on the $\mathcal{D}_T$ target domain.

In the case of $\mathcal{D}_S \neq \mathcal{D}_T$, it means that $\mathcal{X}_S \neq \mathcal{X}_T$ and/or $P(X_S) \neq P(X_T)$. When $\mathcal{X}_S \neq \mathcal{X}_T$, it is referred to as *heterogeneous transfer learning*, while $\mathcal{X}_S = \mathcal{X}_T$ is called *homogeneous transfer learning*. Heterogeneous TL requires feature space adaptation [134] which makes it more complicated. On the other hand, if the marginal probability distributions are different $P(X_S) \neq P(X_T)$, then possibly the conditional probability distributions are disparate as well $P(Y_S \mid X_S) \neq P(Y_T \mid X_T)$. These phenomena are referred to as the *frequency feature bias* or *covariate shift* [135], and the *context feature bias* [134], respectively.

If $\mathcal{D}_S$ and $\mathcal{D}_T$ or $\mathcal{T}_S$ and $\mathcal{T}_T$ are not related enough, the knowledge transfer might not improve the performance of $f_T(\cdot)$. In the worst-case scenario, negative transfer can occur meaning that the model's performance might even decrease [136].

2.2.2 Sim2real object detection

Sim2real object detection is a special case of transfer learning. Instead of real images obtained from the target domain, the model is trained on synthetic data, thus $\mathcal{D}_S \neq \mathcal{D}_T$. On the other hand, it performs the same task, namely object detection (on the same classes of objects), therefore $\mathcal{T}_S = \mathcal{T}_T$. Nevertheless, the model trained on synthetic data, ceteris paribus, does not work on real images as the domains are disparate. This phenomenon is referred to as the *reality gap*, and the main goal of sim2real transfer is to bridge this gap.

Domain adaptation (DA) is an approach to diminish the reality gap. DA attempts to transform one domain into the other domain or transform both domains into a common domain. In the case of sim2real object detection, it usually consists of generating photo-realistic images for the training dataset. The more the generated images resemble the real ones, the more the difference between domains is reduced, and thus, the model's performance on the real images is improved. Synthetic data can be generated with advanced render softwares [137] or based on data-driven approaches, such as *variational autoencoders* (VAE) [138] or *generative adversarial networks* (GAN) [139]. In addition, novel methods based on *neural radiance fields* (NeRF) [44] or Gaussian splatting [45] enable us to generate high-quality synthetic images.

Domain randomization (DR), on the other hand, introduces variability by adding artificial noise to the synthetic training images. The idea is that the added noise makes the

model robust to different domains, as it does not overfit on the domain-specific characteristics, but learns the domain-independent underlying data representation. Another possible interpretation is to consider the different domains as perturbed versions of one common domain. The general idea of introducing variance to simulation was first presented by Jacobi [140].

Other important concepts of transfer learning are the zero-shot and the one-shot transfers. Zero-shot transfer means that there are not any samples accessible from $\{\mathcal{D}_T, \mathcal{T}_T\}$, while one-shot transfer means that only one or a few data are available from $\{\mathcal{D}_T, \mathcal{T}_T\}$.

2.3 Reinforcement learning

Alongside supervised and unsupervised learning, reinforcement learning is one of the main branches of machine learning. It is simultaneously a class of problems, a class of solutions, and the name of the field itself [73]. In RL, an agent attempts to learn the optimal policy for a task through trial-and-error interactions with an environment.

It is beneficial to use RL algorithms, compared to supervised learning (or rule-based systems), when it is easy to define the goal but hard to map specific situations (inputs) to the best actions. E.g., it is easy to define winning or drawing position in a chess game but it is incredibly hard to tell in a specific situation, what the best move is. For these problems, it is often advantageous to let an RL agent explore its environment and come up with a solution to the given problem with trial-and-error.

Compared to other fields of machine learning, the main challenges are the *delayed reward*, the dilemma of *exploration and exploitation*, and that the training data is not *independent and identically distributed* (i.i.d.).

Control theory is closely related to RL, as both aim to develop agents (controllers) that make decisions to optimize performance over time. Traditionally, control theory follows a model-driven approach while RL follows a data-driven approach. Furthermore, while traditional control theory focuses more on minimizing immediate deviations from a desired state, RL is oriented toward maximizing long-term rewards, making it versatile for applications where outcomes unfold over time. Nevertheless, the boundary of the fields has become less distinct in recent years. In Appendix B, a comparison of control theory and RL is presented with the most relevant differences in terms, notations, and formulations. Furthermore, in Appendix C, safe reinforcement is presented where – in several occasions – the methods combine control theory and RL. For readers less familiar with RL, we recommend reading Appendix B and Appendix C at the end of Section 2.3.

2.3.1 Markov decision process

Following [73], RL can be formalised with a Markov decision process represented by a tuple $(\mathcal{S}, \mathcal{A}, r, \gamma, p, \mu_0)$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, $\gamma \in [0, 1]$ is the discount factor, $p(s_{t+1}, r_t \mid s_t, a_t)$ is the transition probability with $s \in \mathcal{S}$ and $a \in \mathcal{A}$, and μ_0 is the initial state distribution.

Every episode[6] starts by sampling from the initial state distribution μ_0. In every timestep $t \in \mathbb{N}$, the agent performs an action according to its policy $\pi(a|s)$ and receives a reward, a new state[7], and a done flag[8] $d \in \{0, 1\}$ from the environment. The agent-environment interaction in MDP is depicted in Fig. 2.6.

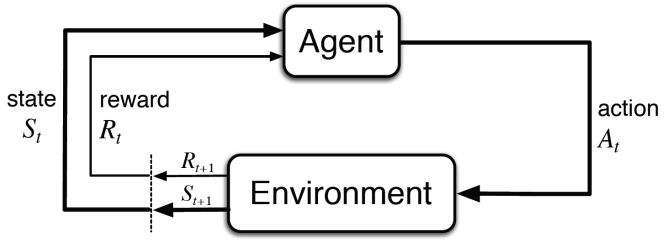


Figure 2.6: The agent–environment interaction in a Markov decision process [73]. The done flag is not depicted.

Learning the optimal policy is formulated as maximising the expected discounted sum of future rewards or expected return $\mathbb{E}_{s_0}[G_0^{\text{disc}} \mid s_0]$ and $G_t^{\text{disc}} = \sum_{i=t}^{T} \gamma^{i-t} r_i$, where $T \in \mathbb{N}$ is the time horizon. Value-based off-policy algorithms learn the optimal policy by learning the optimal Q (action-state value) function: $Q^\pi(s_t, a_t) = \mathbb{E}[G_t^{\text{disc}} \mid s_t, a_t]$.

2.3.2 Multi-goal tasks

In multi-goal tasks, there are multiple reward functions r^g parametrised by the goal $g \in \mathcal{G}$. A goal is described with a set of states $\mathcal{S}^g \subset \mathcal{S}$, and it is achieved when the agent is in one of its goal states $s_t \in \mathcal{S}^g$ [93]. Thus, according to [141] and [93], the policy is conditioned also on the goal $\pi(a|s, g)$. In our implementation, we simply insert goal g into state s

[6]The training process is oftentimes divided into separate attempts to solve the task. One episode is an attempt. After the agent succeeds, fails, or times out, the environment is reset, and the agent can try again.

[7]For simplicity, the environment is considered to be fully observable.

[8]Indicating the end of the episode.

and consequently, when the initial state is sampled from μ_0, the goal is sampled as well. Henceforth, we refer to μ_0 as the initial state-goal distribution.

2.3.3 Reward functions

The agent's goal is to maximise the total amount of reward it receives. Even though this formulation is generally flexible and widely applicable (transferability), the universality changes considerably among different reward functions.

There are two main categories for reward functions, *sparse* and *dense reward*. Sparse reward means that the reward occurs infrequently and typically after the task is completed, e.g., giving some reward for successful termination and zero otherwise. On the other hand, in the dense reward scenario, the informative feedback is frequent, e.g., distance from the target at every timestep. Sparse reward functions are beneficial as they are universal, while dense reward functions can significantly facilitate exploration by sacrificing universality.

Introducing prior knowledge in the form of reward shaping could facilitate the exploration by guiding the agent toward the desired solution. However, 1) constructing a sophisticated reward function requires expert knowledge, 2) the reward function is task-specific, and 3) the agent might learn undesired behaviours. Another source of prior knowledge could be in the form of expert demonstrations. Nonetheless, collecting demonstrations is oftentimes expensive (regarding time and resources) or even infeasible. Furthermore, it constrains transferability as demonstrations are task-specific.

Even though the classification of reward functions in the literature is not well-defined, the following list shows common reward functions.

- **Positive sparse reward.** The agent only gets any reward in case of completion of the tasks, described with Eq. (2.9). The main advantage of this formulation is its universality. Nevertheless, the reward function, without discounting, does not incentivise the agent to solve the task as fast as possible.

$$r(s, \cdot) = \begin{cases} 1, & \text{if } s \in \mathcal{S}^g \\ 0, & \text{otherwise} \end{cases} \tag{2.9}$$

- **Negative sparse reward.** Formulated in Eq. (2.10), the negative sparse reward function facilitates finding the shortest (optimal) solution even without discounting, while being universal.

$$r(s, \cdot) = \begin{cases} 0, & \text{if } s \in \mathcal{S}^g \\ -1, & \text{otherwise} \end{cases} \tag{2.10}$$

- **Sparse adversarial reward.** In adversarial games (e.g., chess), the agent receives the reward only when the game ends, based on the result. It might be formulated as in Eq. (2.11).

$$r(s,\cdot) = \begin{cases} +1, & \text{if } s \in \mathcal{S}^w \\ \frac{1}{2}, & \text{if } s \in \mathcal{S}^d \\ 0, & \text{otherwise,} \end{cases} \tag{2.11}$$

  where $\mathcal{S}^w \subset \mathcal{S}$ and $\mathcal{S}^d \subset \mathcal{S}$ are states where the result is a win or a draw. By definition, $\mathcal{S}^w \cap \mathcal{S}^d = \emptyset$.

- **Distance-from-start dense reward.** In these problems, the agent needs to move away from its starting position. Typical scenarios are when the robot needs to run, jump, or swim. On many occasions, the control effort is subtracted from the reward to incentives smaller actions. A distance-from-start dense reward is formulated in Eq. (2.12).

$$r(s,\cdot) = b_1 \cdot d(s, s_0) - b_2 \cdot \sum_{a \in \mathcal{A}} a^2, \tag{2.12}$$

  where $d : \mathcal{S} \times \mathcal{S} \to \mathbb{R}$ is a distance function and $b_1, b_2 \in \mathbb{R}$ are constants.

- **Distance-to-goal dense reward.** Similar to the distance-from-start reward function with the exception of having a negative reward proportional to the distance to the goal. It must be constructed carefully, otherwise, it is easy to get stuck in local minima. An example of the distance-to-goal dense reward functions is presented in Eq. (2.13).

$$r(s,\cdot) = b \cdot d(s, s_g), \tag{2.13}$$

  where $b \in \mathbb{R}$ is a constant.

In the list above we did not include reward functions with failure (terminal non-goal) states[9]. Nevertheless, with a simple modification these reward functions can be transformed to include failure states as well – e.g., Eq. (2.10) can be formulated as Eq. (2.14).

$$r(s,\cdot) = \begin{cases} 0, & \text{if } s \in \mathcal{S}^g \\ -P, & \text{if } s \in \mathcal{S}^f \\ -1, & \text{otherwise,} \end{cases} \tag{2.14}$$

where $\mathcal{S}^f$ is the set of failure states and $P \in \mathbb{R}^+$ is the penalty term for termination. P could be a function of the failure states as well. Furthermore, by definition, $\mathcal{S}^g \cap \mathcal{S}^f = \emptyset$.

[9] With the exception of the sparse adversarial reward.

2.3.4 Demonstrations

Another important aspect of RL is whether the agent has access to any form of demonstration. A demonstration is an example of the desired (optimal or suboptimal) behaviour provided by an external source which can significantly facilitate the exploration [142]. It can be especially beneficial in sparse reward scenarios. Oftentimes, an expert human provides these examples in which case it can be referred to as human demonstrations. Nevertheless, collecting expert demonstrations is expensive and time-consuming, or even not feasible. On the other hand, automatically generating demonstrations before the training presumes that the task can be solved already at least for a set of goals, raising the question of whether RL training is necessary.

2.3.5 Categorization of RL algorithms

Model-based vs model-free

Model-based RL algorithms attempt to build a model of the environment $p(s_{t+1}, r_t|s_t, a_t)$. The agent uses the model of the environment to plan its actions by simulating future states and rewards. This approach is beneficial if $p(s_{t+1}, r_t|s_t, a_t)$ is known or can be efficiently approximated. Examples for Model-based RL algorithms are Dynamic programming (DP) [143], Dyna-Q [144], and AlphaGo [78].

Model-free RL agents do not build a model of the environment but simply learn the policy or value function directly through trial-and-error. This approach is beneficial when the dynamics of the environment are not known and would be difficult to learn, such as in robotics. Examples of model-free RL algorithms are Q-learning [145], SARSA [146], Deep Q-learning (DQN) [79], or policy gradient methods such as REINFORCE [147], A2C/A3C [148], PPO [149], DDPG [80], [81], TD3 [82], and SAC [83].

Value-based vs policy-based

Value-based RL agents focus on estimating the $v(s)$ state value function, Eq. (2.15) or the $q(s, a)$ state-action value function, Eq. (2.16), which is then utilised to derive the optimal policy. In most cases, the policy simply chooses greedily the best action with $1 - \epsilon$ probability or a random action with $\epsilon \in [0, 1]$ probability (ϵ-*greedy* [73]). The random actions are beneficial for exploration preventing the agent from getting stuck in local minima. Note that $\epsilon = 0$ results in a pure *greedy policy*. An alternative to the greedy or ϵ-greedy policies is the *upper confidence bound* (UCB) algorithm [150]. *Optimistic*

initialization of the value function [151] and the technique of *exploring starts* [73] are two further methods to aid exploration. Even though value-based methods can be more stable and more sample-efficient, their application is limited to discrete (or discretized) action spaces. Examples of value-based RL methods are Q-learning, SARSA, and DQN.

$$v_\pi(s) = \mathbb{E}\left[G_t \mid S_t = s\right] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s\right] \tag{2.15}$$

$$q_\pi(s,a) = \mathbb{E}\left[G_t \mid S_t = s, A_t = a\right] = \mathbb{E}_\pi\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \mid S_t = s, A_t = a\right] \tag{2.16}$$

Policy-based RL algorithms directly approximate the policy, which is typically a probability distribution over the available actions, by directly mapping the states to actions without explicitly evaluating the value function. One of the main advantages of policy-based RL methods is their capability to handle continuous action spaces. Nevertheless, they can suffer from high variance and slow convergence, and in general, they are less sample-efficient than value-based methods. An example of a pure policy-based method is REINFORCE.

There exists a third type of RL algorithms, named actor-critic models, combining the advantages of value-based and policy-based RL agents. Actor-critic methods learn both the policy and the value function. The actor uses feedback from the critic (bootstrapping) to adjust its policy parameters. Nevertheless, actor-critic methods fall closer to policy-based methods as the policy maps the states to actions. Actor-critic models, such as A2C/A3C, PPO, DDPG, TD3, SAC are especially relevant in the field of robotics.

On-policy vs off-policy

In the case of on-policy RL agents, the behaviour policy π_B – which is responsible for collecting the training data – and the target policy π_T – which is optimised by the agent – are the same. Mathematically, $\pi_B = \pi_T$. Consequently, when the policy of the agent is updated, the previously collected data cannot be utilised anymore causing low sample efficiency. Examples of on-policy RL algorithms are SARSA, REINFORCE, and PPO.

In the case of off-policy RL algorithms, $\pi_B \neq \pi_T$. Thus, off-policy RL is better for sample efficiency as it can use data collected by previous policies or even external data (demonstrations). Moreover, in this scenario, the training data is closer to i.i.d. Examples of off-policy RL agents are Q-learning, DQN, DDPG, TD3, and SAC.

2.3.6 Tabular reinforcement learning

Tabular RL refers to the problems when the state and action spaces are discrete and typically small enough to be efficiently represented in tables. These problems are easier to handle due to the limited number of states and actions. There are different approaches to tackle this type of RL problem, such as *dynamic programming* (DP), *Monte Carlo* (MC), and *temporal difference* (TD) methods. Nevertheless, all these approaches build on some type of value function estimation utilising tables, representing the states or state-action pairs. Since efficient discretization is often infeasible for continuous robotic problems, tabular methods are less practical in such cases. Nonetheless, as the theory of tabular methods forms the basis of continuous methods, a short summary is provided in Appendix D. For a thorough description of tabular RL, we refer the reader to Part 1 of [73].

2.3.7 Reinforcement learning in robotics

Even though RL algorithms achieved superhuman performance in numerous domains, the field of robotics poses significant challenges as the state and action spaces are continuous, and the reward function is predominantly sparse. Furthermore, on many occasions, the agent is devoid of access to any form of demonstration. In this section, the main challenges, characteristics, and state-of-the-art solutions are presented.

Deep Q-learning (DQN) [79] offers a solution to the problem of continuous state spaces. In DQN, instead of tables, the value function is approximated with a neural network, and thus its input (the state) can be continuous. As updating the NN indirectly alters the values nearby, it is beneficial to apply a *target network* which is a copy of the NN that is updated less frequently.

Even though DQN offers a solution for continuous state spaces, handling continuous action spaces remains an even more challenging problem. In this case, choosing the best action in the Bellmann optimality equation, Eq. (D.2), is not feasible due to the continuous action space.

The cross-entropy algorithm [152] is a policy-based and on-policy method that can handle continuous state and action spaces. It is easy to implement and works at a satisfactory level in the case of simple problems with frequent rewards. The policy is realised with a neural network that attempts to compute the best action in a given state. The network is trained on a subset of data collected by the policy. In the subset, the best episodes (with the highest return) are selected.

The REINFORCE algorithm [147] is one of the fundamental methods of RL. It is a policy-based and on-policy algorithm, sharing characteristics with the cross-entropy

method. In the cross-entropy method, there is a binary distinction between the data used for training and not used. In contrast, the REINFORCE algorithm utilises all collected data for training. However, the weights of the data are set based on their discounted total reward[10], presented in Eq. (2.17). Even though REINFORCE significantly outperforms the cross-entropy method, it has limitations, namely, high gradient variance, challenging exploration, and correlation between samples (data is not i.i.d.). For reducing the gradient variance, different baselines can be subtracted from the scaler. Typical baselines are the means or the estimated values of the states.

$$\mathcal{L} = -\sum_{k,t} Q_{k,t} \cdot \log\left[\pi\left(a_{k,t} \mid s_{k,t}\right)\right], \tag{2.17}$$

where $k \in \mathbb{N}$ is the index of the episodes, $t \in \mathbb{N}$ is the timestep in that given episode, and the $Q_{k,t} \in \mathbb{R}$ scaler is the discounted total reward.

Actor-critic methods introduce bootstrapping to the REINFORCE algorithm. There are two NNs, an actor and a critic. The former is responsible for the policy, while the latter approximates the value of the states for bootstrapping. The architecture of a traditional actor-critic model is presented in Fig. 2.7a. In the *advantage actor-critic*, or A2C algorithm, the gradient is scaled with the *advantage* term, presented in Eq. (2.18). Thus, the $Q_{k,t}$ scaler term in Eq. (2.17) is replaced with the $A(s_t, a_t)$ scaler from Eq. (2.18). The advantage represents how much more beneficial to take a specific action in a given state, compared to the average/general action in that state. The *asynchronous advantage actor-critic*, or A3C [148] method introduces parallelism in the training process where many A2C *workers* are combined either at the data or at the gradient level.

$$A(s_t, a_t) = Q_w(s_t, a_t) - V_v(s_t), \tag{2.18}$$

where $A(s_t, a_t) \in \mathbb{R}$ is the advantage, $Q_w(s_t, a_t) \in \mathbb{R}$ is the value of a given state-action pair, and $V_v(s_t) \in \mathbb{R}$ is the value fo a given state.

One of the most challenging aspects of RL is that the training data is generated mostly or fully by the trained policy itself (not i.i.d.) causing instability in the training. On-policy RL algorithms, e.g., REINFORCE or A2C/A3C, can be trained, by definition, only on the last batch of data[11] which exacerbates the problem of training stability. When there is a considerable update on the weights of the policy, the new policy might be significantly worse than the previous one. As the training data for the next weight update comes from

[10] From another point of view, in the cross-entropy method, the weight of a data point can only be zero or one, depending if it is included in the training or not.

[11] Typically a couple of episodes.

the new policy, it could cause a downward spiral. The aim is to move to the optimal solution as fast as possible while maintaining stability and avoiding downward spirals. *Trust regions* [149], [153], [154] offer a solution to the aforementioned problem. One of the most widely used trust region methods is the *proximal policy optimization* (PPO) [149] algorithm which utilises a clipped objective function, presented in Eq. (2.19) to ensure the stability of the policy updates.

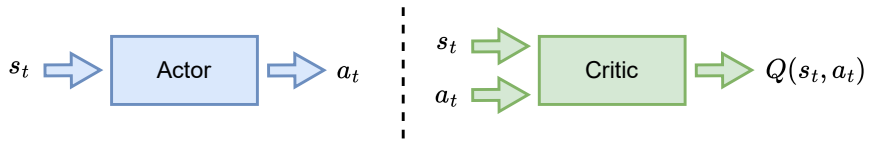
$$\mathcal{L}^{\text{clip}}(\theta) = \mathbb{E}_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t \right) \right], \tag{2.19}$$

where $r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$ is the probability ratio, $\hat{A}_t$ is the estimated advantage at time step t, and ϵ is a hyperparameter that controls the clipping range.

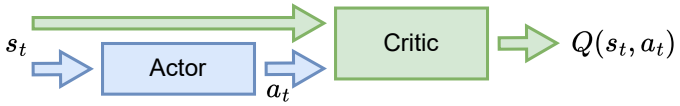
Even though trust region methods could improve the stability of RL training, they are still on-policy algorithms with the aforementioned drawbacks. *Deep deterministic policy gradient* or DDPG [80], [81] is an off-policy actor-critic model combining ideas of Q-learning and policy-based methods. Up to this point, the algorithm of this section must have stochastic or probabilistic policies. The main reason behind this is that the target function needs to be smooth and continuously differentiable to allow for gradient updates. In DDPG, the actor and critic are trained together as depicted in Fig. 2.7b. As the value function $Q(s_t, a_t)$ is the target for the actor, the policy can be deterministic. For training, the policy must be deterministic because sampling would disrupt the continuity of the gradient. For collecting training data, DDPG adds noise to the action to encourage exploration. Furthermore, to stabilise training, DDPG applies target networks for both the actor and critic. As DDPG is off-policy, it utilises an experiment replay buffer $\mathcal{B}_{\text{er}}$ where the transitions are stored. For training, the transitions are sampled from $\mathcal{B}_{\text{er}}$. Consequently, the data used for training is collected by different policies at different times of the training, making the data closer to i.i.d. Additionally, DDPG being off-policy enables the use of external data, e.g., human demonstrations. The main drawbacks of DDPG are: 1) it is considerably sensitive to its hyperparameters, 2) balancing exploration and exploitation is challenging due to the deterministic policy, and 3) the learnt Q function dramatically overestimates the real Q-values, breaking the policy by exploiting the errors in the Q function.

The *twin delayed deep deterministic policy gradient* (TD3) [82] and the *soft actor-critic* (SAC) [83] are two separate lines of research[12] to improve DDPG, sharing some key similarities. TD3 introduces three tricks to address the overestimation of Q-values which is a common failure of DDPG. Firstly, instead of one, two Q functions are learnt for the Bellman error loss function. For the target, the smaller Q-value is selected (*clipped*

[12]Published around the same time.



(a) Traditional actor-critic



(b) DDPG

Figure 2.7: **Top.** The architecture of the traditional actor-critic model. The actor and the critic are trained separately. **Bottom.** The architecture of DDPG. The actor and the critic are trained together. As the value function $Q(s_t, a_t)$ is the target for the actor, the policy can be deterministic. For training, the policy must be deterministic because sampling would disrupt the continuity of the gradient.

double-Q learning). Secondly, the policy is updated less frequently than the Q function (*delayed policy updates*). Thirdly, to avoid exploiting the Q function errors, noise is added to the target action (*target policy smoothing*). On the other hand, while SAC also utilises the clipped double-Q learning, it is equipped with an entropy-regularised stochastic policy. An entropy term $H(\pi(\cdot \mid s))$ is introduced to the loss function, making it maximise both the entropy (exploration) and the expected return (exploitation). Due to its inherent stochasticity, it also gains a similar advantage as the target policy smoothing in TD3.

2.3.8 Evaluation of RL algorithms

State-of-the-art deep reinforcement learning models are compared based on just a few experiments, primarily due to constraints on training time. Therefore, simple point estimates of aggregate performance such as mean and median scores across tasks are insufficient as they do not capture the statistical uncertainty implied by the finite number of training runs. In this section, we present the most relevant statistical evaluation methods utilised in RL.

In general, *confidence intervals* (CIs) are beneficial to measure uncertainty. The bootstrap CI method creates multiple datasets by resampling with replacement from a set of data points (results of independent training runs). As the distribution of the means of

the resampled datasets approaches a normal distribution[13], the CI can be calculated. Traditionally, bootstrap CI is performed on a single task [155]–[157]. Agarwal et al. [158] proposes the method of stratified bootstrap CI which performs a bootstrap CI across multiple tasks using stratified sampling.

Another useful evaluation method is presenting the performance profiles. A tail distribution function is defined as $F(\tau) = P(X > \tau)$, where $\tau \in \mathbb{R}$, and X is a real-valued random variable[14]. The performance profiles are beneficial for comparing different algorithms at a glance. In mathematical terms, X has stochastic dominance over Y if $P(X > \tau) \geq P(Y > \tau)$, for all τ, and for some τ $P(X > \tau) > P(Y > \tau)$, where X and Y are random variables. Two main versions are the run-score distribution [158] and the average-score distributions [159]. Examples of performance profiles are presented among our experimental results in Fig. 5.4 and in Fig. 5.8.

Displaying the probability of improvement is another beneficial evaluation method. It shows the probability of Algorithm X exceeding Algorithm Y in a set of tasks. It is important to note that it only indicates the probability of improvement and not the magnitude of the improvement.

Finally, standard aggregate performance metrics have shortcomings. The median has high variability and it is unchanged even when half of the results are zero, while the mean can be significantly influenced by some outliers. Thus, [158] proposes to use the *interquartile mean* (IQM) and the *optimality gap* (OG) as alternatives to the median and the mean. IQM removes the bottom and top 25% of the runs and calculates the mean of the remaining 50% of the runs. OG is the amount that the algorithm fails to reach a desirable target. It is important to note that in the OG metric if an algorithm surpasses the desirable target, it does not affect its OG score.

2.4 Curriculum learning

Humans require a highly organised training process – introducing different concepts at different times – to become fully functional adults. From kindergarten to university, we progress through various stages of the educational system, ultimately reaching a university mathematics lecture where the professor teaches us how to solve differential equations. This phenomenon is not only valid for the academic education system. Without a doubt, Usain Bolt had to learn how to walk before he could set the world record for the 100-meter sprint race.

[13] Central limit theorem.

[14] Performance estimates are random variables, based on a finite number of runs.

The core idea of *curriculum learning* (CL) is that ML models – similarly to humans – might as well benefit from an organised training process (curriculum). Depending on the ML field, it might mean that a model does not receive all the training data at once, but subsets of data are introduced at different times, or that it learns tasks of varying difficulty in a structured manner. In this section, firstly, the general concepts of curriculum learning are presented, and then the RL-specific CL is detailed. For a more thorough overview, we refer the reader to [97], [98].

2.4.1 Easy2hard curriculum learning

Originally, the curriculum followed an *easy2hard* or *starting small* structure. After Selfridge et al. [160] designed a curriculum for the cart pole control problem, Bengio et al. [96] introduced CL by showing that an image-based shape recognition model could benefit from an easy2hard structure. Two datasets of rectangle, ellipse, and triangle shapes were given. One with general shapes, and the other with special cases of these shapes: squares, circles, and equilateral triangles. The training with a 2-stage curriculum – first on the special cases with less variability and then on the general shapes (target distribution) – yielded significantly better performance than the traditional no-curriculum training. An illustration of easy2hard CL is depicted in Fig. 2.8.

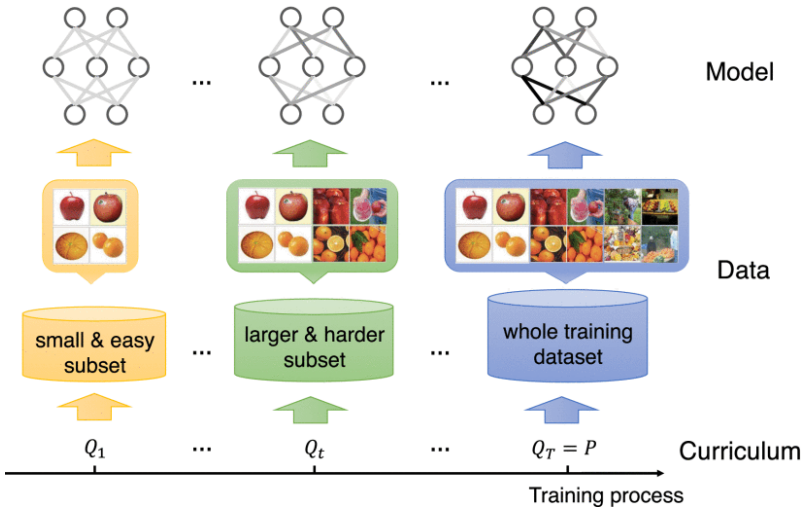


Figure 2.8: Illustration of the easy2hard CL from [98].

As an optimization problem, CL can be seen as a particular continuation method. Continuation methods [161] are optimization strategies for non-convex criteria which solve

first an easier (smoother) objective and then gradually increase the difficulty (less smooth function) until reaching the target objective, depicted in Fig. 2.9. From a transfer learning point of view, the preceding objectives can be seen as stages of pre-training. From a data distribution point of view, CL can be seen as focusing more on cleaner, less noisy data, especially in the first stages of the training, and wasting less time on the noisier, harder data [98], [162].

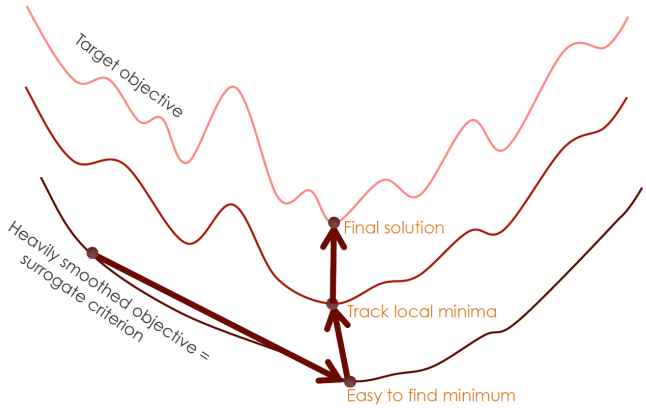


Figure 2.9: Illustration of the continuation method from [163].

2.4.2 Generalised curriculum learning

Even though easy2hard CL methods are shown to be superior compared to no-curriculum baselines, interestingly, in some cases [164], [165], the opposite of easy2hard CL, the strategy of *hard2easy* or *starting big* was beneficial. Methods of hard example mining [166], [167] aim to select the most difficult examples. Consequently, the definition of CL was broadened.

Definition of generalised curriculum learning: "... a curriculum is a sequence of training criteria over T training steps. Each criterion Q_t includes the design for all the elements in training a machine learning model, e.g., data/tasks, model capacity, learning objective, etc. Curriculum learning is the strategy that trains a model with such a curriculum." [98]

2.4.3 CL in supervised learning

In supervised learning, a CL framework typically consists of two main components: the *difficulty measurer* and the *training scheduler*. The difficulty measurer assigns a difficulty score to the samples. According to the difficulty score, the training scheduler can introduce the subsets of data at different times[15]. If both the difficulty measurer and the training scheduler are constructed by human prior knowledge, then it is referred to as predefined CL [96], [170], [171]. If data-driven algorithms are applied, then it is considered to be an automatic CL method. Automatic CL can be further categorised into self-paced learning [172], [173], transfer teacher [168], [174], RL teacher [164], [175], and other automatic CL [176], [177].

2.4.4 CL in reinforcement learning

Compared to supervised learning, in reinforcement learning, CL is slightly different. One of the most relevant differences is that for supervised learning the training dataset is given, while in RL, it is generated by the agent. Thus, before the training, the training dataset cannot be divided into groups based on difficulty. On the other hand, as the data collection is part of the machine learning problem, there is more possibility of utilising CL. Following [97], CL can typically control either the data collection or the data exploitation process in RL. The data collection process can be controlled by changing the initial state distribution [89]–[91], the reward function [178], [179], the goals [92]–[95], the environment [175], [180], or the opponent [78], [181]. The data exploitation methods might control the transition selection [86], [88] or intervene by modifying the transitions [84], [85].

[15]The training scheduler is referred to as *pacing function* in [168] and competence function in [169].

Chapter 3

Sim2real knowledge transfer for object detection

Contents

3.1 Introduction 40
3.2 Related work 43
3.3 The S2R-ObjDet method 47
3.3.1 Sim2real knowledge transfer 47
3.3.2 Data generation 49
3.3.3 Training 57
3.4 Evaluation protocol 57
3.5 Generalised confusion matrix for object detection 58
3.6 The InO-10-190 dataset 61
3.7 Results 64
3.7.1 Zero-shot transfer (ZST) 66
3.7.2 One-shot transfer (OST) 69
3.8 Ablation study 74
3.8.1 Seed 74
3.8.2 Texture and post-processing 75
3.8.3 Data size 76

3.8.4 Gravity, positional disturbance, and bounding-box calculation . . 77
3.8.5 Cutouts 79
3.8.6 Faster R-CNN 79
3.9 Robotic application 80
3.10 Conclusion 81

Robots working in unstructured environments must be capable of sensing and interpreting their surroundings. One of the main obstacles of DL models in the field of robotics is the lack of domain-specific labelled data for different industrial applications. In this chapter, we present a sim2real transfer learning method based on domain randomization for object detection (S2R-ObjDet) with which labelled synthetic datasets of arbitrary size and object types can be automatically generated. Subsequently, an object detection model is trained to detect the different types of industrial objects. With the proposed domain randomization method, we could shrink the reality gap to a satisfactory level, achieving 86.32% and 97.38% mAP_{50} scores respectively in the case of zero-shot and one-shot transfers, on our publicly available manually annotated InO-10-190 dataset, containing 190 real images of 920 object instances of 10 classes. The class selection was simultaneously based on different and similar objects in order to test the robustness of the model in terms of detecting different classes and differentiating between similar objects. Our solution fits for industrial use as the data generation process takes less than 0.5 s per image and the training lasts only around 12 h, on a GeForce RTX 2080 Ti GPU. Furthermore, the model can reliably differentiate similar classes of objects by having access to only one real image for training. To our best knowledge, this was the first work satisfying these constraints. Moreover, we proposed the generalised confusion matrix (GCM) which is an adaptation of the traditional confusion matrix to object detection. It offers a solution to the shortcomings of the classical precision-recall-based mAP and F_1 score. With GCM, the misclassification error can be quantified and evaluated. For a short presentation and additional materials, we refer the reader to the project page: www.danielhorvath.eu/sim2real.

3.1 Introduction

New-generation intelligent manufacturing (NGIM) is a recent trend embodying the in-depth integration of new-generation artificial intelligence with advanced manufacturing technology such as robotics. It became the driving force of the fourth industrial revolution and it belongs to the *human-cyber-physical system* 2.0 (HCPS 2.0) framework. Contrary to traditional manufacturing where robots work in structured environments and perform high-accuracy, repetitive tasks with minimal sensory input, an NGIM system is designed to be flexible and to take over as much intellectual and manual labor as possible. Thus, human workforce can concentrate on more valuable creative work. [10]

Consequently, the main interest has been shifting towards adaptive robotic applications that can cost-efficiently handle low-volume customised products and integrate human operators with different skills and abilities. Computer vision plays an essential role here – the highly researched field has already proven useful in pick-and-place, bin picking, grasping,

navigation, or quality assurance tasks. As two examples, Zeng et al. [182] created a vision-based model that can predict parameters of motion primitives through trial and error for robotic grasping and throwing, and Alonso et al. [183] designed a model for real-time semantic segmentation of RGB images for a mobile robot application.

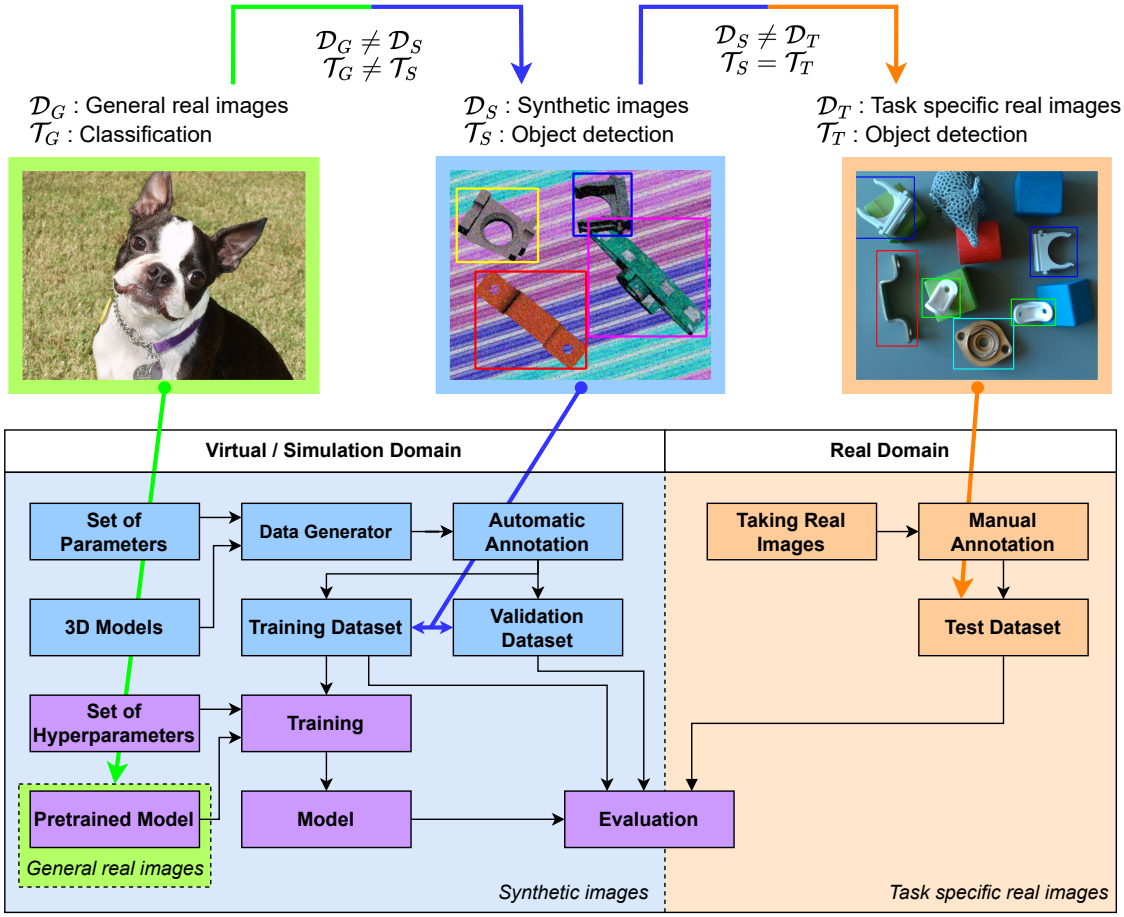


Figure 3.1: **Top.** Pipeline of knowledge transfer. **Bottom.** Flowchart diagram of our data generation, training, and evaluation process. The picture of the Boston bull is from ImageNet [70].

In a computer vision context, *deep convolutional neural networks* (DCNNs) have been performing incredibly well on large public datasets such as ImageNet [70] or MS COCO [71]. Having reached human-level performance in classification, the main focus of computer

vision research has shifted to object detection and led to networks such as the Faster R-CNN [120], Single Shot MultiBox Detector (SSD) [124], and multiple versions of YOLO [34], [121]–[123], [126]–[130]. Even though these networks outperform the traditional machine-vision-based methods by a significant margin, their application in robotic systems has its difficulties. One of the main obstacles in applying DL models is that they need to be trained on sufficiently large, domain-specific, and expertly labelled datasets.

Levine et al. [49] conducted two experiments, in order to create a dataset of real images for their DCNN model predicting the success of robotic grasp attempts, as well as to control these attempts. Yielding records of more than 1.7 million grasp attempts with the simultaneous use of 6–14 robots, the process took months to complete. This example shows that, in general, collecting a dataset from the real world not only requires an immense amount of resources but is a time-consuming process as well.

The main motivation behind transfer learning is to overcome the aforementioned obstacle by transferring knowledge between tasks or domains [68], [69]. Sim2real transfer is a special case of transfer learning, where the source domain is the virtual simulation of the real world, while the target domain is the physical reality itself [184]. With sim2real knowledge transfer, the model can be trained in a virtual simulation, having the necessary amount of labelled synthetic data. In the case of computer vision, the images are rendered, and the labels, too, can be generated for them in a self-supervised way. Thus, the time-consuming process of data collection and labeling can be omitted. As the domains of training and test datasets are inherently different, ceteris paribus, the learnt model will perform poorly in the target domain. The phenomenon of performance loss from the simulation to the real domain is called *reality gap*. Domain adaptation [185] and domain randomization [186], [187] are two ways of shrinking this gap.

Our contributions can be summarised as follows:

- Our novel sim2real domain randomization method for object detection (S2R-ObjDet) which describes the data generation process, and our sim2real training pipeline.
- Our InO-10-190 dataset, which is a real-world dataset of 190 manually annotated images (RGB and depth) containing 920 objects of 10 classes that address the problem of high class similarity to validate our method. The dataset is publicly available alongside our code and can serve as a benchmark for object detection algorithms.
- For evaluation, we introduced the generalised confusion matrix (GCM) which is an altered version of the traditional confusion matrix fit to object detection. It proved to be extremely useful for detecting and quantifying misclassifications which is the primary cause of performance loss in the case of similar classes.

- A real-world robotic implementation of the method as a proof of concept containing a ROS-based robot control system.
- Our implementations of our S2R-ObjDet method[1], and our robot control framework[2] that are available at the project's git repositories. Both can be used as out-of-the-box software modules for industrial robotic applications.

The results can be highlighted as follows:

- We achieved 86.32% mAP_{50} and 97.38% mAP_{50} scores in zero-shot and one-shot transfers that show the usability of our methods even in the case of an industrial application where high reliability is crucial.
- Our experiments show that having even only one sample image from the target domain significantly improves the model's performance for similar classes.
- A thorough ablation study focusing on finding the key factors of the data generation process.

The industrial benefit of our work is a freely available tool streamlining the training of CNN-based models for object detection. Our built-in sim2real domain randomization method spares the user the effort of collecting and annotating a large dataset, as it automatically generates training data from 3D models. Optionally, one annotated image with all relevant objects can be added for improved performance. With training being automated as well, the entire workflow from 3D models to trained CNN takes only around 13 h.

In Sections 2.2, the problem statement of the field of transfer learning was outlined. In further parts, in Sections 3.2 the related work is presented. In Sections 3.3 and 3.4, our S2R-ObjDet method is outlined with the evaluation protocol. In Section 3.5, our GCM is introduced. In Sections 3.6 and 3.7, our InO-10-190 dataset and our results are shown. Additionally, in Section 3.8, a thorough ablation study of our method is presented. Finally, Section 3.9 shows a real-world robotic implementation of our method.

3.2 Related work

Tobin et al. [186] trained a modified version of VGG-16 [111] deep neural network architecture for object localization. They generated non-realistic synthetic RGB images

[1] https://git.sztaki.hu/emi/sim2real-object-detection
[2] https://git.sztaki.hu/emi/robot_control_framework

randomising the number and shape of the distractor objects, the position and texture of all objects, the texture of the background, the position, orientation and the field of view of the camera, the number of lights in the scene, the position, orientation, and specular characteristics of the lights, and the type and amount of random noise added to images. The random textures were either a random single colour, a gradient between two colours, or a checker pattern of two random colours. The following non-industrial objects were used: cones, cubes, cylinders, hexagonal prisms, pyramids, rectangular prisms, tetrahedrons, and triangular prisms. The images were rendered with the built-in renderer of the MuJoCo Physics Engine [188], and no real images were used for training the model. They achieved around 1.5 cm accuracy in the real-world environment. Tobin et al. conducted further research [50] where they trained a deep neural network for grasp planning using only synthetic images and domain randomization, and achieved an 80% success rate in a real-world environment.

Borrego et al. [189] presented a plug-in for the Gazebo simulator [190]. Introducing variation reduced the reality gap between simulated and real-world data. In the case study, three types of objects were detected: box (cube), cylinder, and sphere. The simulated scenes contained a ground plate and a single light source. The objects were placed on a grid to prevent collusion, but they were rotated randomly[3]. Then, the camera and the light source were moved to random positions. 4 different types of textures were used, namely: flat, gradient, checkerboard, and Perlin noise [191]. For training, the SSD [124], and separately, the Faster R-CNN [120] networks were used. With the two networks, 70% and 88% mAP_{50} were achieved, respectively, using 121 real images. Training the same networks with 9000 simulated images yielded 64% and 82% mAP_{50}, respectively. Interestingly, the hybrid approach (using real and synthetic images) accomplished 62% and 83% mAP_{50}, respectively. For all experiments, IoU 0.5 threshold (mAP_{50}) was used, and the test results were validated on 121 real images (different from the 121 images used for the training). A follow-up ablation study [187] revealed that Perlin noise has a crucial influence on the performance of the model. Furthermore, data generation process was further accelerated to 9000 full-HD images in roughly 1.5 hours (around 0.6 s per image).

Pashevich et al. [184] trained manipulation policies in a simulation environment with an object localization proxy task. Depth images for training were simulated in PyBullet [192] and gathered with a Kinect-1 camera from the real world. For finding the best data augmentation transformation and their order, Monte Carlo Tree Search (MCTS) [193] was used. The transformations were selected from the Python Image Library (PIL). The transformations were evaluated individually and as a sequence as well. From all transformations

[3]In this regard, we found that introducing some disturbance to object placement significantly increases the performance, see in Section 3.8.4.

examined, the cutout transformation [194] performed best on real images (however, in our experiments, this was not the case for RGB images, see in Section 3.8.5), and the best sequence of transformations was: cutout, erasing an object, white noise, edge noise, scale, salt noise, posterise, and sharpness, in this order. With the aforementioned sequence, 1.09 $\pm$ 0.73 cm position error was achieved in the real environment, for cubes of 4.7 cm edge length.

James et al. [195] trained an end-to-end robotic controller on synthetic data with domain randomization. The inputs of the deep-neural-network-based model were an image and the joint angles of the robot, while its output were motor velocities. The task was an abstract tidying manipulation, namely, putting a cube into a basket. Similarly to [189], Perlin noise was used as a perturbation. The model was examined in dynamically changing illumination settings, in the presence of distractors, including human presence, new cube size in test time, and with a moving basket. Experiments yielded at least a 75% success rate in all conditions, except for a spotlight and a smaller cube in test time. In these cases, the model had a 56% and a 41% success rate, respectively.

Devo et al. [196] used domain randomization to train a target-driven visual navigation model. The goal was to find a specific object in a maze. Maze wall heights, maze wall textures, maze floor textures, light colour and intensity, and the light source angle were subject to randomization. For simulation, the Unreal Engine 4 [197] was used. An average of 72% success rate was achieved in simulation, and 46% in the real world. The experiments showed that direct sim2real transfer is possible for this kind of problem as well.

Chen et al. [198] created the Domain Adaptive Faster R-CNN model for cross-domain object detection. Domain shift stemming from image-level and instance-level shifts were tackled with an approach based on $\mathcal{H}$-divergence theory and adversarial training. The study focuses on street images for self-driving cars where the domains are disparate due to different camera types and setups, different cities and diverse appearance of objects, or the particular weather conditions. Some experiments were also carried out with sim2real knowledge transfer, as the model was trained on SIM 10k [199] and evaluated on the Cityscapes dataset [200]. Their method improved the performance of the Faster R-CNN model from 30.12 AP_{50} to 38.97 AP_{50} in the case of the car class.

Focusing on street scenarios, Sankaranarayanan et al. [201] proposed an unsupervised domain adaptation approach based on generative adversarial networks for semantic segmentation problems. For the synthetic source domain, the SYNTHIA [202] and the GTA V [203] datasets, and for the target domain, Cityscapes dataset [200] were used. The approach achieved 36.1 mIoU and 37.1 mIoU scores transferring knowledge from SYNTHIA and GTA V, respectively. Without domain adaptation, the method scored 26.8 and 29.6 mIoU.

Tremblay et al. [204] generated synthetic images with domain randomization techniques to perform object detection of cars in street scenarios. 100K images were generated with maximum 14 cars each, selected randomly from 36 car models. The models were evaluated on the KITTI dataset [205]. Three DCNN architectures were trained (Faster R-CNN [120], R-FCN [206], and SSD [124]), scoring 78.1, 71.5, and 46.3 AP_{50}, respectively, on the single-class object detection problem. Interestingly, better results were obtained than by training the same architectures on the virtual KITTI dataset [207] which has a high correlation to the KITTI dataset. The performance could be improved by fine-tuning the models on real images. With 6000 real images, the performance of the Faster R-CNN model reached 98.5 AP_{50}.

Barisic et al. [35] proposed an approach to generate a synthetic aerial dataset for *unmanned aerial vehicle* (UAV) monitoring. The images are rendered in Blender [137]. In order to facilitate the learning of a shape-based representation, they applied random textures on the UAV models but added real background from the Polyhaven [208] dataset. They achieved a 17% and a 3.7% AP_{50} improvement with the tiny version of YOLOv4 and a rise of 20% and 1.1% AP_{50} in the case of the Faster R-CNN, compared to the baselines without texture randomization on two real-world datasets.

Hinterstoisser et al. [209] inserted 3D models of objects in real images, using OpenGL with Phong shading [210] for rendering. Small perturbation were permitted in the ambient, diffuse, and specular parameters, and the light colour. Gaussian noise and a blur with Gaussian kernel were added to better integrate the objects with the background. A Faster R-CNN model was primarily used for training, with freezing the weights of the feature extractor. The latter significantly improved the performance of the model (although, Tremblay et al. [204] later reported the opposite effect in their case).

Zhang et al. [211] propose an adversarial discriminative sim2real approach to transfer visio-motor policies. The method was demonstrated in a table-top object-reaching task. A blue cuboid object had to be reached with a velocity-controlled 7 DoF robot arm. The method could reduce the real data requirement by 50%, while 97.8% success rate and 1.8 cm control accuracy were achieved.

Clever et al. [212], [213] proposed a method to predict human position (resting on a bed) and contact pressure from depth data and gender information. The method achieved 3.837 MSE(kPa^2) trained on 97K synthetic images. In comparison, the same method reached 3.151 MSE(kPa^2) trained on 11K real images and 2.849 MSE(kPa^2) trained on both real and synthetic images (108K). For evaluation, the SLP dataset [214] was used.

Gomes et al. [215] proposed a simulated model for the GelSight tactile sensor. Having computed the height map of the elastomer, the internal illumination of the elastomer is

calculated. The usefulness of the model was also demonstrated with a sim2real classification task. For the study, 12 texture maps resembling real objects were created and randomly perturbed on the captured synthetic data, improving the classification accuracy from 43.76% to 76.19%.

The above works are diverse in terms of the problem itself, the input type, the domain of the application, and the amount of synthetic and real images used to train the model, making a complete comparison a challenge. Nevertheless, a general overview organised by selected characteristics is presented for reference in Tab. 3.1. In general, certain limitations of the above works relate closely to our work (solving object detection):

- The classification part of the problem is less challenging as the works use simple shapes such as cubes, spheres, and cones, or even have one class only.
- The works rely on considerably more synthetic and real images for training.

Even though the cited works use transfer learning (domain adaptation or domain randomization) to reduce the reality gap, they do not solve the same machine learning problem, and may use different models as well. All of this needs to be taken into consideration if an in-depth comparison is desired.

3.3 The S2R-ObjDet method

This section presents our sim2real domain randomization method for object detection (S2R-ObjDet). In Section 3.3.1, the sim2real training pipeline is outlined. Then, in Section 3.3.2, the data generation module is detailed. Finally, in Section 3.3.3, the details of the training are presented. The implementation is freely available at our git repository[4].

3.3.1 Sim2real knowledge transfer

The flowchart diagram of our data generation, training, and evaluation process is depicted in Fig. 3.1. It can be broken down into functionally separable tasks. The data generation process creates randomised and post-processed synthetic images of given objects. It also automatically generates the annotations for the images. Thus, the output of the data generation process is a set of images paired with their labels grouped into a training and a validation dataset.

[4] https://git.sztaki.hu/emi/sim2real-object-detection

Table 3.1: Summary of related works. Abbreviations are the following Sim: Simulator, Synt: Synthetic, Img: Images, P&P: Pick-and-place, Segm: Segmentation, Class: Classification, ObjDet: Object detection, Nav: Navigation, FrR-CNN: Faster R-CNN, MJC: MuJoCo [188], Gaz: Gazebo [190], PyB: PyBullet [192], V-R: V-REP [216], UE4: Unreal Engine [197], OGL: OpenGL [217], DFP: DART [218], FleX [219], Pyrender [220], Blen: Blender [137], acc: accuracy, [a]: domain adaptation (otherwise domain randomization), [u]: unlabelled. The AP and mAP scores are with IoU=0.5. YCB [221].

Work	Problem	Input	Domain	Base Model	Sim	Synt. Img.	Real Img.	Results
Tobin [186]	ObjDet	RGB	Shapes	VGG-16	MJC	5K–50K	0	1.5 cm acc
Tobin [50]	Grasping	Depth	YCB	CNNs	MJC	2K/obj	0	80% success
Borrego [189]	ObjDet	RGB	Shapes	FrR-CNN	Gaz	9K	0	82% mAP
						9K	121	83% mAP
				SSD		9K	0	64% mAP
						9K	121	62% mAP
Pashevich [184]	ObjDet	Depth	Cubes	ResNet-18	PyB	2K	0	1.09±0.73 cm
	Placing		Cups			—	0	15/20
James [195]	P&P	RGB, joints	Cube	CNNs	V-R	100K–1M	0	≥41% success
Devo [196]	Nav	2xRGB	—	CNNs	UE4	540K	0	46% success
Chen[a] [198]	ObjDet	RGB	Street	FrR-CNN	—	10K	3K[u]	38.97 AP
Sankarana-rayanan[a] [201]	Segm.	RGB	Street	GAN	—	25K	5K[u]	37.1 mIoU
Tremblay [204]	ObjDet	RGB	Street	FrR-CNN	UE4	100K	0	78.1% AP
						100K	6K	98.5% AP
				R-FCN		100K	0	71.5% AP
				SSD		100K	0	46.3% AP
Barisic [35]	ObjDet		UAV	YOLOv4-t	Blen	32k	0	+17% AP
				FrR-CNN				+20% AP
Hinter-stoisser [209]	ObjDet	RGB	House - hold	CNNs	OGL	20K	0	ca. 70% mAP
Zhang [211]	Grasping	RGB	Cube	VGG-16	V-R	3K	93+186[u]	97.8% success, 1.8 cm acc
Clever [212], [213]	Contact pressure	Depth	Humans	CNNs	DFP	97K	0	3.837 kPa^2
						97K	11K	2.849 kPa^2
Gomes [215]	Class.	RGB	Shapes	ResNet-50	Gaz	1470	0	76.19% acc
Our results	**ObjDet**	**RGB**	**Indus-trial**	**YOLOv4**	**PyB**	**4K**	**0**	**86.32% mAP**
						2K	**1**	**97.38% mAP**

For training, only the images from the training dataset are used. As the initial layers of the neural network perform low-level image processing tasks such as detecting contours, lines, or edges, we utilised a pretrained image classifier model as a feature extractor of our object detector. This is the first phase of our knowledge transfer, depicted in the top-left section of Fig. 3.1. The knowledge transfer goes from $\{\mathcal{D}_G, \mathcal{T}_G\}$, where $\mathcal{D}_G$ is the domain of the dataset of general public images and $\mathcal{T}_G$ is classification, to $\{\mathcal{D}_S, \mathcal{T}_S\}$, where $\mathcal{D}_S$ is the domain of synthetic images (source domain of the sim2real knowledge transfer), and $\mathcal{T}_S$ is object detection. Here, $\mathcal{D}_G \neq \mathcal{D}_S$, and $\mathcal{T}_G \neq \mathcal{T}_S$. The second phase of knowledge transfer is the sim2real transfer which goes from $\{\mathcal{D}_S, \mathcal{T}_S\}$ to $\{\mathcal{D}_T, \mathcal{T}_T\}$, where $\mathcal{D}_T$ is the domain of our industrial environment (target domain), and $\mathcal{T}_T$ is object detection. Here, $\mathcal{D}_S \neq \mathcal{D}_T$ but $\mathcal{T}_S = \mathcal{T}_T$. Although the pretrained network does possess some learnt knowledge from the domain of a given general public dataset ($\mathcal{D}_G$), it does not have direct knowledge of the target objects. Consequently, $\mathcal{D}_G \neq \mathcal{D}_S \neq \mathcal{D}_T$. Even though $\mathcal{D}_G \neq \mathcal{D}_T$, the characteristics of the domains are similar.

3.3.2 Data generation

The data generation process is responsible for the creation of synthetic images paired with accurate automatic ground-truth annotations. In several stages of this process, artificial random perturbations are applied as domain randomization techniques. It is important to mention that although cluttered scenes may occur, we restrict our focus to images without significant occlusion.

Framework

For data generation, the PyBullet [192] physics simulator was utilised since it has an easy-to-use, intuitive API, including an image renderer tool, and an integrated physics simulator where the gravitational force can be simulated easily.

The duration of dataset generation is a relevant aspect of the method, as in the industry, on many occasions, it is not feasible to wait long hours or even days to start the training, which can be a time-consuming process itself. One of the advantages of domain randomization over domain adaptation is that it is generally faster as images do not need to be photo-realistic. In our case, for data generation, we could achieve less than 0.5 s per image on a GeForce RTX 2080 Ti GPU. With 4000 images, this amounts to around 33 minutes. If 1 image is rendered in 1 minute (which was plausible in the case of photo-realistic images at the time of the research), then instead of 33 minutes, the aforementioned 4000 images

would take more than 66 hours[5]. Having generated the dataset, the training lasts around 12 h, thus a complete generation and training process can be executed automatically in around 13 h.

Object generation

The framework is capable of placing any type of object into the simulation if its 3D description file is given. In industrial applications, which are the focus of this research, these 3D models are often readily available.

The most relevant input parameters of the object generation process are summarised in Tab. 3.2 and the process works as follows:

- A horizontal plane is placed at the vertical $z = 0$ position.
- According to the grid size $n_{\text{grid}} \in \mathbb{Z}^+$ and the grid spacing $d_{\text{space}} \in \mathbb{R}^+$ parameters, the $x, y \in \mathbb{R}$ coordinates of the grid points are set.
- The vertical coordinates of the grid points are set by $d_{\text{height}} \in \mathbb{R}^+$.
- The objects are not placed exactly at the grid points. The x, y, and z coordinates of the objects are obtained from uniform distributions described in Eq. (3.1).

$$r_i^{\text{obj}} \sim \mathcal{U}(r_i^{\text{grid}} - \epsilon_i^{pos} \cdot d_{\text{space}}, r_i^{\text{grid}} + \epsilon_i^{\text{pos}} \cdot d_{\text{space}}) \mid i = x, y, z, \tag{3.1}$$

  where $\mathcal{U}(\cdot, \cdot)$ is the uniform distribution, $r_i^{\text{obj}} \in \mathbb{R}$ is the i direction element of the pose (position and rotation angles) of the object, $\boldsymbol{r}^{\text{obj}} = [r_x^{\text{obj}}, r_y^{\text{obj}}, r_z^{\text{obj}}, r_{rx}^{\text{obj}}, r_{ry}^{\text{obj}}, r_{rz}^{\text{obj}}]$. Furthermore, $r_i^{\text{grid}} \in \mathbb{R}$ is the grid position and $\epsilon_i^{\text{pos}} \in [0, 1]$ is normalised maximum shift in i direction. Note that $\boldsymbol{r}^{\text{grid}} = [r_x^{\text{grid}}, r_y^{\text{grid}}, r_z^{\text{grid}}]$ and $\boldsymbol{\epsilon}^{\text{pos}} = [\epsilon_x^{\text{pos}}, \epsilon_y^{\text{pos}}, \epsilon_z^{\text{pos}}]$. Typically $\boldsymbol{\epsilon}^{pos} = [0.1, 0.1, 0.0]$, which corresponds to a $\pm 10\%$ disturbance in the x and y direction. It is important to note that at this stage only the x,y, and z components of the $\boldsymbol{r}^{\text{obj}}$ vector are set, the rotation parameters rx,ry, and rz are set later.
- Once object selection has been performed, the appropriate 3D model of the object is loaded into the specific coordinates. Predefined weights describe the probability of selecting a specific object. Furthermore, a distracting cuboid object (which is

[5]Since our experiments in 2021, methods based on DA became significantly faster, thus the time advantage might not be that relevant.

not in any of the classes) or a void object (leaving that grid point empty) can be selected. The sizes of the distracting objects are also individually randomised. The aforementioned probabilities are set by $\boldsymbol{p}^{\text{objects}}$.

- The objects are also randomly rotated around the x, y, and z axes, described in Eq. (3.2).

$$r_i^{\text{obj}} \sim \mathcal{U}(E_{i,\text{lower}}^{\text{rot}}, E_{i,\text{upper}}^{\text{rot}}) \mid i = rx, ry, rz, \tag{3.2}$$

  where $\mathcal{U}(\cdot,\cdot)$ is the uniform distribution, $r_i^{\text{obj}} \in \mathbb{R}$ is the corresponding element of the $\boldsymbol{r}^{\text{obj}} = [r_x^{\text{obj}}, r_y^{\text{obj}}, r_z^{\text{obj}}, r_{rx}^{\text{obj}}, r_{ry}^{\text{obj}}, r_{rz}^{\text{obj}}]$ vector. Furthermore, $E_{i,\text{lower}}^{\text{rot}} \in \mathbb{R}$ and $E_{i,\text{upper}}^{\text{rot}} \in \mathbb{R}$ are the limits of rotation around the corresponding axis.

- The objects and the ground plane are given some random textures drawn from three public datasets [222]–[224], with the probability of $p_{texture}$. Some examples of the textures are shown in Fig. 3.2. Random RGB colours are assigned to the objects (or to the ground plane) which do not receive any texture.

- Before rendering the image, the objects are dropped down from their original position to the ground plane. Thus, the objects are captured in a natural stable position. The simulation of the free fall takes around 0.05–0.1 s per scenario (with every step included, the generation of an image with its label is around 0.45–0.5 s).

Figure 3.2: Some examples of the textures used from [222]–[224].

Table 3.2: The most relevant input parameters of the data generator module in terms of object generation. Param: Parameter.

Param.	Set	Type	Description
n_{grid}	$\mathbb{Z}^+$	scalar	The grid size ($n_{\text{grid}} \times n_{\text{grid}}$).
d_{space}	$\mathbb{R}^+$	scalar	The grid spacing. The distance between the adjacent grid points.
d_{height}	$\mathbb{R}^+$	scalar	The position of the grid (initial object position) in z direction.
$\boldsymbol{\epsilon}^{\text{pos}}$	$\mathbb{R}^3$	vector	Contains the proportionate (to the grid spacing) perturbation limits in the directions of x, y, and z around the centre point.
$\boldsymbol{E}^{\text{rot}}$	$\mathbb{R}^{3\times 2}$	matrix	$\boldsymbol{E}^{\text{rot}}$ describes the lower and upper bounds for the possible rotation angles of the directions in x, y, and z.
$\boldsymbol{p}^{\text{objects}}$	$\mathbb{R}^{n_{obj}+2}$	vector	Contains the selection probabilities of the given objects (including the distractor object and the void object).
p_{texture}	$\mathbb{R}$	scalar	The probability that a specific object or the ground plane has a random texture. Otherwise, it gets a random colour.

Image rendering

For rendering an image, the camera pose, its inner parameters, and additional parameters must be set. The most relevant parameters of the image rendering are presented in Tab. 3.3. The algorithm works as follows:

- The camera is placed in a randomised position pointing to a random point around the centre of the grid defined by $\boldsymbol{R}^{\text{target}} \in \mathbb{R}^{3\times 2}$ and $\boldsymbol{R}^{\text{cam}} \in \mathbb{R}^{3\times 2}$. The randomization is constrained to ensure that the centre points of all objects appear on the rendered image.

$$r_i^{\text{target}} \sim \mathcal{U}(R_{i,\text{lower}}^{\text{target}}, R_{i,\text{upper}}^{\text{target}}) \mid i = x, y, z, \tag{3.3}$$

  where $r_i^{\text{target}} \in \mathbb{R}$ is the i coordinate of the camera target position, $R_{i,\text{lower}}^{\text{target}}$ and $R_{i,\text{upper}}^{\text{target}}$ are the limits of the uniform distribution for the i^{th} coordinate.

$$r_i^{\text{cam}} \sim \mathcal{U}(R_{i,\text{lower}}^{\text{cam}}, R_{i,\text{upper}}^{\text{cam}}) \mid i = \Psi, \Theta, d, \tag{3.4}$$

  where $r_i^{\text{cam}} \in \mathbb{R}$ is the corresponding element of $\boldsymbol{r}^{\text{cam}} = [r_\Psi^{\text{cam}}, r_\Theta^{\text{cam}}, r_d^{\text{cam}}]$ vector, describing the rotation angle around the global z axis, the pitch angle from the global x–y plane, and the distance between the camera and the target point. $R_{i,\text{lower}}^{\text{cam}}$ and $R_{i,upper}^{\text{cam}}$ are the corresponding lower and upper limits.

- The camera field-of-view (FOV) θ_{FOV} and its additional intrinsic parameters are set. Image width and height are obtained from a uniform distribution defined by Eq. (3.5) and Eq. (3.6).

$$m_{\text{width}} \sim \mathcal{U}(m_{\text{lower}}^{\text{width}}, m_{\text{upper}}^{\text{width}}), \tag{3.5}$$

  where $m_{\text{width}} \in \mathbb{N}$ is the width of the image in pixels and $\boldsymbol{m}^{\text{width}} \in \mathbb{N}^2$ vector contains the lower and upper bound of the image width.

$$m_{\text{height}} \sim \mathcal{U}(m_{\text{lower}}^{\text{height}}, m_{\text{upper}}^{\text{height}}), \tag{3.6}$$

  where $m_{\text{height}} \in \mathbb{N}$ is the width of the image in pixels and $\boldsymbol{m}^{\text{height}} \in \mathbb{N}^2$ vector contains the lower and upper bound of the image height.

- The RGB, D (depth), or RGB-D images are taken, defined by $m_{\text{type}} \in \{0, 1, 2\}$. RGB-D images are created by concatenating the RGB and the D images. For one layout (object generation), only one image is taken.

Table 3.3: The most relevant input parameters of the data generator module in terms of image rendering. Param: Parameter.

Param.	Set	Type	Description
$\boldsymbol{R}^{\text{target}}$	$\mathbb{R}^{3\times 2}$	matrix	The camera points to a certain point in the 3D space. $\boldsymbol{R}^{\text{target}}$ describes the lower and upper bound of the camera target point in the directions of x, y, and z.
$\boldsymbol{R}^{\text{cam}}$	$\mathbb{R}^{3\times 2}$	matrix	The pose of the camera is defined by 3 parameters: the rotation angle around the global z axis, the pitch angle from the global x–y plane, and the distance between the camera and the target point. $\boldsymbol{R}^{\text{cam}}$ describes the lower and upper bound of these parameters.
θ_{FOV}	$\mathbb{R}$	scalar	The field-of-view of the camera.
$\boldsymbol{m}^{\text{width}}$	$\mathbb{N}^2$	vector	The $\boldsymbol{m}^{\text{width}}$ parameter describes the lower and upper bound of image width in pixels.
$\boldsymbol{m}^{\text{height}}$	$\mathbb{N}^2$	vector	The $\boldsymbol{m}^{\text{height}}$ parameter describes the lower and upper bound of image height in pixels.
m_{type}	$\{0, 1, 2\}$	scalar	Three types of images can be rendered: 3-channel RGB images, depth images, and 4-channel RGB-D images.

Label generation

Having generated the objects and rendered the image, the ground-truth annotation must be computed. As it is object detection, the label $\boldsymbol{y} = \{(\boldsymbol{b}_i, c_i^{\text{class}}) \mid i = 1, 2, \ldots, N\}$, where $\boldsymbol{b}_i = [x_i, y_i, w_i, h_i]$ represents the bounding box (BB) coordinates and c_i^{class} is the class label (see in Section 2.1.5).

The aforementioned ground-truth generation is an automatic process involving a coordinate transformation from the simulation 3D world coordinate system to the image 2D coordinate system.

The 4×4 transformation matrix is the matrix product of the view matrix and projection matrix of the camera, respectively. In order to transform a point from the world coordinate system to the image space, it must be multiplied with this transformation matrix and then scaled back by its fourth coordinate to get the true projection. For a detailed explanation of the projection, we refer the reader to [225].

Furthermore, we implemented two ways of computing the bounding boxes. The two approaches differ in the number of points that are transformed into the image space. One approach transfers only the 8 points (per object) of the world 3D axis-aligned bounding boxes (AABB), which is available in the PyBullet simulator, whereas the other transforms all the points of the objects to the image space. Henceforward, we refer to the former approach as the 8-point method and the latter as the all-point approach. Having obtained the transformed points, the second part of the algorithm is the same in both cases: the minimum and maximum values in x and y directions are selected to define the limits of the BBs. The centre points can be computed as the arithmetic means of the minimum and maximum values. As a result, the latter method gives tighter, more accurate bounding boxes at the cost of extra computation.

Post-processing

The technique of domain randomization was performed in multiple steps of the previously defined synthetic image generation process. In the post-processing phase, as a domain randomization technique, additional artificial noise is introduced to alter the images. The images are perturbed with a randomised multi-colour pepper-and-salt noise and a Gaussian blur. Furthermore, as Pashevich et al. [184] found the rectangle cutouts useful for depth images, experiments were made with rectangle cutouts, and additional circle, as well as line cutouts. The noise types are shown in Fig. 3.3, described in Tab. 3.4, and were applied in the following order: 1.) rectangle cutout, 2.) circle cutout, 3.) line cutout, 4.) multi-colour pepper-and-salt, and 5.) Gaussian blur.

Table 3.4: The types of noises in post processing.

Name	Description
Multi-colour pepper-and-salt	With a given p_{mc} probability, every channel of every pixel is set either to zero or to the maximum value.
Gaussian blur	With a given p_{gauss} probability, the whole image is blurred with a randomised filter size.
Cutout	A given number of rectangle-, circle-, or line-shaped regions with randomised dimensions and at randomised positions are coloured to a random RGB value.

The goal of post-processing is to force the model not to learn the synthetic domain-specific characteristics, but to try to learn the domain-independent underlying data representation. The ablation study on our experiments, described in Section 3.8, shows that having the post-processing module undoubtedly improves the performance of the models with the test dataset. Nevertheless, it also reveals that the added cutout noises did not improve the performance compared to the default Gaussian blur and multi-colour pepper-and-salt noise in the case of our RGB images.

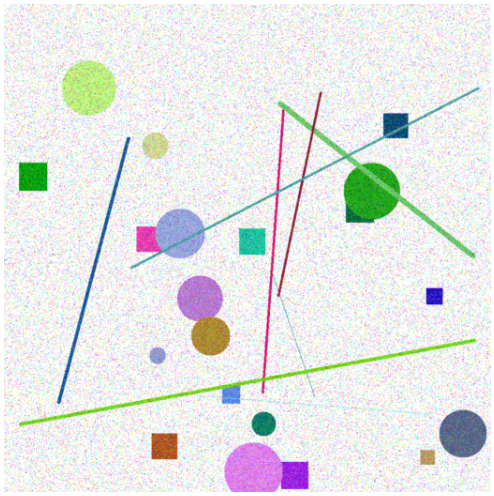


Figure 3.3: Post-process transformation on a blank image.

Table 3.5: The most relevant advanced data augmentation tools in the training process.

Name	Description
Angle	Randomly rotates images.
Saturation	Randomly changes the saturation.
Exposure	Randomly changes the brightness.
Hue	Randomly changes the hue colour channel.
Mosaic	Combine 4 random images into a single, assembled mosaic [34].
Jitter	Randomly changes the size of the images and their aspect ratio.
Random	Randomly resizes network size after every 10 batches.

3.3.3 Training

The method is compatible with arbitrary object detection models, although their efficiency might vary. At the time of the research, ViTs were at their early stage and CNN-based architectures dominated the object detection benchmarks. Therefore, we have chosen the CNN-based YOLOv4 [34] architecture for the following reasons:

- At the time of the research, it had the best speed and accuracy trade-off which made it a good fit for robotic applications. It also has a tiny version, allowing it to run in real-time even on a microcomputer such as a Raspberry Pi or NVIDIA Jetson Nano.
- Its training framework contains additional advanced data augmentation tools. For more information, we refer the reader to [34]. These tools help to introduce further perturbation to the system.

For the training, a model pretrained on ImageNet [70] is used. The most relevant hyperparameters for the advanced data augmentation tools are shown in Tab. 3.5, keeping the original names of the parameters.

In Section 3.8.6, we present the results of our method only changing the object detection model from YOLOv4 to Faster R-CNN.

3.4 Evaluation protocol

In this section, we outline the metrics used to evaluate our models. One of the bases of our evaluation is comparing the mAP score of the different models. The mAP score is

described in Section 2.1.5. Here, we define how we measured the reality gap and outline further details of our evaluation process. Our novel evaluation metric, the generalised confusion matrix is presented separately in Section 3.5.

To evaluate the solution of the classical machine learning problem, (training and evaluation on the same domain), real-world images would not be needed. In this case, the performance is assessed on the generated validation dataset that is not used for training but comes from the same distribution $P_{\text{train}}(X) = P_{\text{valid}}(X)$. The solution can be assessed by the value of the mAP score of the model on the valid dataset, and the capability of generalization (within the specific domain) can be measured by comparing the performance of the model on the training and the validation datasets, as in Eq. (3.7).

$$G_{\text{ML}} = \text{mAP}_{\text{train}} - \text{mAP}_{\text{valid}}, \tag{3.7}$$

where $G_{\text{ML}} \in \mathbb{R}$ is the performance gap between the model's performance on the train (mAP_{train}) and the validation (mAP_{valid}) sets measured with the mean average precision metric (mAP).

To evaluate the performance of the knowledge transfer, a manually annotated test set of real images is needed. In this case, $P_{\text{valid}}(X) \neq P_{\text{test}}(X)$. We expect that the given model performs notably worse on the test set than on the validation and training sets. To measure the magnitude of the reality gap, we can define it as the difference of the performance of the model on the validation and test sets, as shown in Eq. (3.8).

$$G_{\text{reality}} = \text{mAP}_{\text{valid}} - \text{mAP}_{\text{test}}, \tag{3.8}$$

where $G_{\text{reality}} \in \mathbb{R}$ is the reality gap.

As presented in detail in Section 3.7, several training runs were carried out to test our method. For every dataset generated, three independent training sessions were conducted, resulting in three different models (sets of weights) in order to measure the deviance of the training process. The average performance of the models refers to the arithmetic mean of the results of these three models. We also use the F_1 score measure, which is defined as the harmonic mean of the precision and the recall values.

3.5 Generalised confusion matrix for object detection

Alongside the S2R-ObjDet method, one of the added values of this research is the proposed generalised confusion (GCM) which is an adaptation of the traditional confusion matrix

to object detection. Confusion matrix is standard for classification, however, to our best knowledge, it has never been utilised for object detection before publishing our results in [1]. Since then, a similar concept was published in [226].

Most typically, the methods are evaluated on the mAP or F_1 score derived from the precision-recall curves of the models, see in Section 2.1.5. Nevertheless, these metrics have disadvantages, most importantly, they do not address the problems of object misclassification, false positives, and false negatives. Even though a method's mAP or F_1 score would decrease if an object is detected but misclassified, it is equivalent to a scenario when the model completely failed to detect any object in that position. Thus, detecting misclassification errors is highly beneficial for analyzing and improving the models' performance.

The confusion matrix tracks misclassifications in the context of classification problems (see in Section 2.1.4). In the case of classification problems, for every ground truth, there is exactly one prediction, and for every prediction there is precisely one ground truth. The prediction can be either correct or incorrect, but the 1-1 ratio is fixed. For object detection, for every image, the ground truth and equally the output of the given model are lists, thus for one ground truth there can be zero, one, or any number of predictions and vice versa. When there is no prediction to a ground truth, it means that the model did not find any object (of any class) at that position (false negative). On the other hand, if there is a prediction belonging to no ground truth, it means that there is a prediction where there should not be (false positive). These scenarios cannot be handled with a classical confusion matrix.

Thus, we introduce the generalised confusion matrix $\boldsymbol{D}^{\text{gen}} \in \mathbb{N}^{C+1 \times C+1}$ and use it as an additional performance measure. The adaptation is depicted in Fig. 3.4 and detailed below:

- Adding an extra row and an extra column to the classical confusion matrix. Thus, there are $C + 1$ rows and columns, where $C \in \mathbb{Z}^+$ is the number of classes. The additional column represents the objects that are not predicted to any of the classes but actually belong to one class (false negatives). On the other hand, the additional row of the matrix represents the cases when the model predicted an object of a class in a position where there should not be any object (false positives).

- The values of the diagonal, $D^{\text{gen}}_{i,i}$, are the correct predictions. For simplicity, the last element of the diagonal is zero, $D^{\text{gen}}_{C+1,C+1} \doteq 0$. This element should contain the number of objects that are not in the images and the model rightfully did not find them, which does not have any meaning.

- As more than one prediction can belong to one ground-truth object, a given ground-truth object appears in the matrix as many times as many predictions are paired with it. Therefore, contrary to the traditional confusion matrix, the sum of all elements in the matrix will not necessarily be equal to the sum of all ground-truth objects or predictions. A prediction-GT pairs are selected based on their class-independent IoU scores, see in Section 2.1.5.

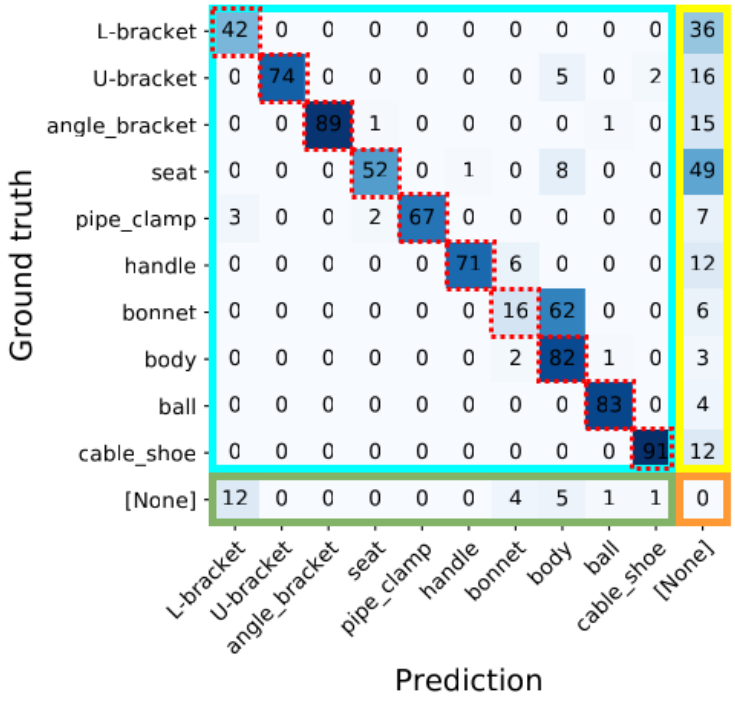


Figure 3.4: Generalised confusion matrix (GCM). In the cyan frame, the traditional 10×10 confusional matrix for the 10 classes. The correct detections are in the diagonal, marked with red dotted lines. An extra row is added, marked in green, to the predictions that do not belong to any ground truth object (false positives). Furthermore, an extra column is added, marked in yellow, for the objects that were not found (false negatives). Finally, for simplicity, marked in orange, the extra square at the bottom right of the matrix which is zero by definition. In this example, it can be seen at a glance, that several (62) bonnet objects were misclassified as body objects, and only 16 bonnet objects were classified correctly.

Utilising the GCM, not only the class-specific performance of the model can be easily

assessed, but the misclassification can be seen at a glance. Misclassification is especially important in the case of similar classes as it could be the primary cause of performance loss, as it is shown in our case study in Section 3.7, and it is not shown in the traditional precision-recall curve.

As the predictions of the given object detection model depend on the τ_{con} confidence threshold[6], the GCM depends on τ_{con} as well. Thus, it is beneficial to find the optimal τ_{con} based on the precision-recall curve or the F_1 score, and then plot the GCM with the optimal τ_{con}.

3.6 The InO-10-190 dataset

This section presents our InO-10-190 dataset which serves as the test dataset, detailed in Section 3.7.

Ten industrial parts were selected for the dataset. Object diversity as well as object similarity were the two major points of consideration. The former helps us to evaluate the detection performance of the model for various types of objects, whereas the latter is important in assessing the classification performance of the model. In general, it is easier to misclassify objects with similar features. Thus, this problem can be considered to be more challenging than the detection of less complex and fairly different shapes such as cubes and spheres. The selected objects are depicted in Fig. 3.5, and their virtual counterparts in Fig. 3.6. These images are samples of $X \in \mathcal{X}$ obtained from two different $P(X)$ probability distributions. The virtual images are from the probability distribution P_S of $\mathcal{D}_S = \{\mathcal{X}, P_S(X)\}$, while the real images are from the probability distribution P_T of $\mathcal{D}_T = \{\mathcal{X}, P_T(X)\}$. It is important to note that the InO-10-190 test dataset contains only the real images, the virtual models are presented here only for comparison.

As it can be seen, on one hand, objects of different sizes, shapes, colours, and materials were selected to increase diversity. On the other hand, some objects share similar characteristics, such as circular holes. Furthermore, two parts, the bonnet (#7) and the body (#8) were chosen because of their high level of similarity, as shown in Fig. 3.7.

Constructing the dataset, 190 real images of 920 object instances were taken with different layouts and illumination settings. The images were captured with an Intel RealSense D435 camera. For easy and fast image capturing, a frame was designed that holds the camera 310 mm from the ground. A 300×210×10 mm light blue wooden base supports the frame – this is also where the parts were placed. The images show not only this base

[6]Only the predictions with confidence $p_i^{\text{con}} \geq \tau_{\text{con}}$ are accepted, see in Section 2.1.5.

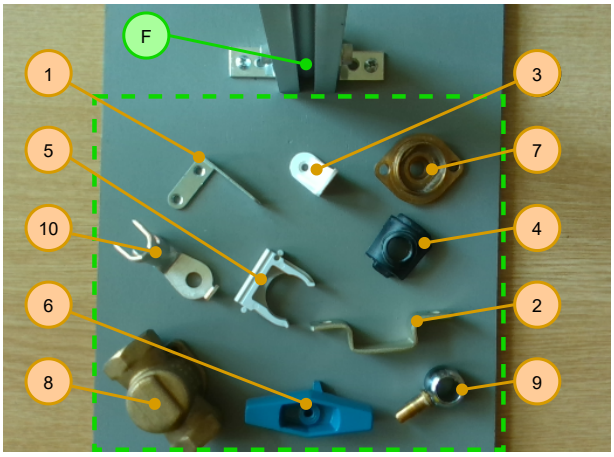


Figure 3.5: The selected industrial parts in the InO-10-190 dataset. Their names in order of their identifier numbers are the following: 1. L-bracket, 2. U-bracket, 3. angle bracket, 4. seat, 5. pipe clamp, 6. handle, 7. bonnet, 8. body, 9. ball, 10. cable shoe. The letter 'F' designates the camera holder frame. The green dashed lines show the borders of the cropped images.

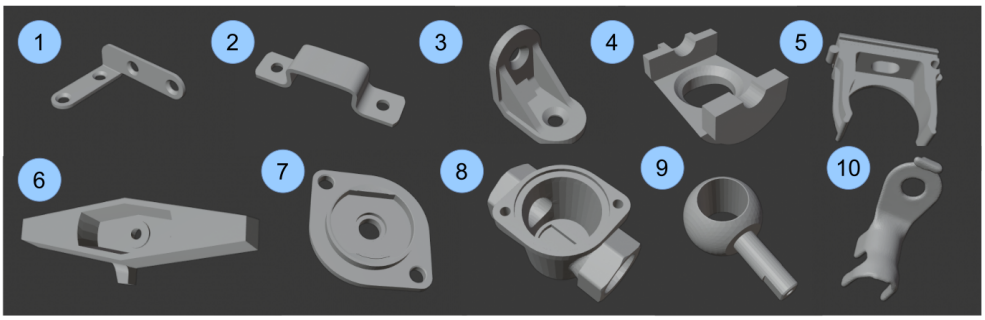


Figure 3.6: The 3D models of the selected industrial parts in the InO-10-190 dataset. Their names in order of their identifier numbers are the following: 1. L-bracket, 2. U-bracket, 3. angle bracket, 4. seat, 5. pipe clamp, 6. handle, 7. bonnet, 8. body, 9. ball, 10. cable shoe. Their scaling factors are different for better visualization.

area but the background (tabletop) as well – this is done on purpose. In order to have a slightly different dataset as a reference, we also transformed the aforementioned dataset by cropping the images to fit in the wooden base. The cropped area is signed with dashed green lines in Fig. 3.5. Some examples of cropped images are shown in Figs. 3.14 and 3.18.

The annotations for the test dataset were labelled manually and saved in the YOLO

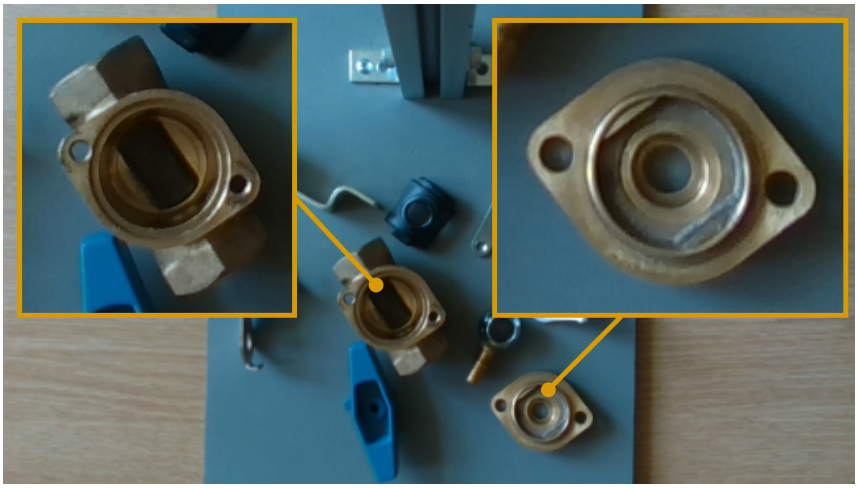


Figure 3.7: The similarity of the body and the bonnet objects.

annotation format. As it contains all the necessary bounding boxes and class information, other annotation formats can be generated from them. We emphasise that these images of real objects were not used at any point for training the models, except in the case of one-shot knowledge transfer. In this case, only one real image was used. The experiment of one-shot transfer is presented in Section 3.7.2.

For all images, the matching depth images are recorded as well. The depth images are transformed in a way that each pixel point of the RGB image can be associated with the same pixel point of the depth image (the transformation is necessary as the fields-of-view of the cameras for RGB and for depth images are different). Thus, all the annotations for the RGB images are the same for the depth images. Even though the depth images were not used in the current research, this additional data can be valuable for later use or for other researchers.

The InO-10-190 dataset is summarised in Tab. 3.6 and samples of the dataset are depicted in Fig. 3.8. In Group A, every image contains only one object (except one image without any objects). In Group B, every image contains multiple objects, but no class is represented more than once. In Group C, spotlight illumination is applied from one side to test the robustness of the models to illumination settings. In Group D, cluttered scenes are recorded. In Group E, distractor objects (cubes, cylinders, triangular prisms, and a 3D-printed elephant) are placed in the scene. Finally, in Group F, in every picture, only one class is presented, however, unlike in Group A, there are multiple instances of this class in every image.

The group-wise class distributions are depicted in Fig. 3.9. As it can be seen, the

classes are relatively equally distributed in the groups. Even though in the mAP metric, the mean of the classes is calculated, thus it is less influenced by class imbalance, it is advantageous to create an equally distributed test dataset. Obviously, for training, which can be sensitive to class imbalance, the synthetic data is generated with a random selection of objects, thus eliminating any notable class imbalance. The dataset can be downloaded from the project repository[7].

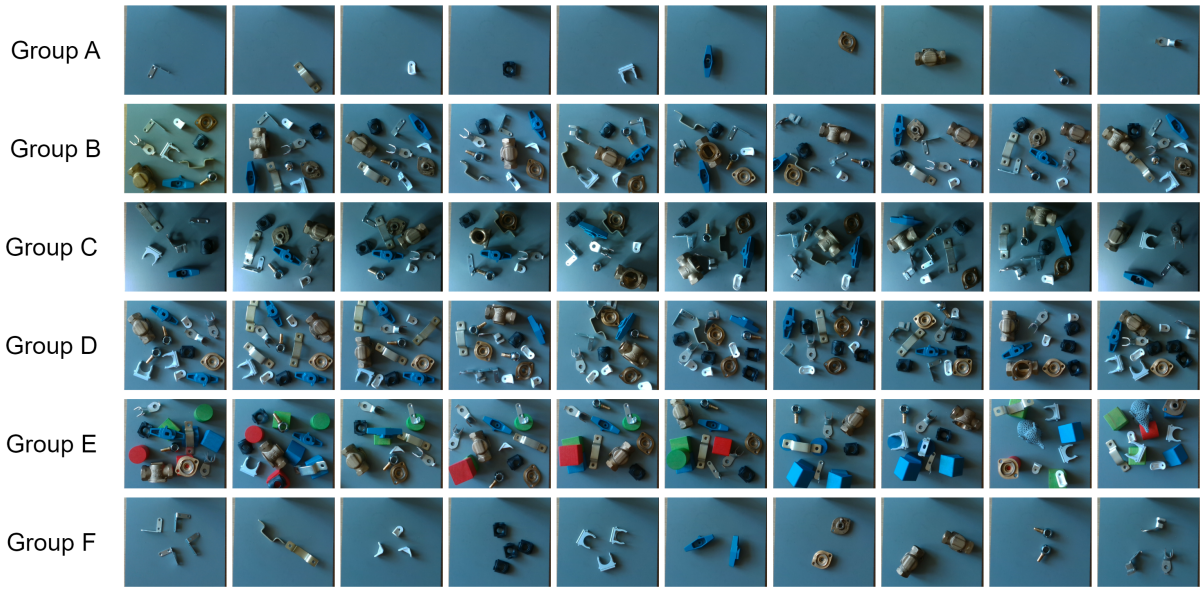


Figure 3.8: Samples of our public and annotated InO-10-190 dataset (cropped version).

3.7 Results

In this section, we show the strengths of our S2R-ObjDet method, described in Section 3.3, by applying it to the problem presented in Section 3.6. Moreover, we display the advantage of our GCM in the performance evaluation, presented in Section 3.5. The training and validation datasets are generated synthetic images by the S2R-ObjDet method[8], while the test dataset is the InO-10-190 dataset.

[7] https://git.sztaki.hu/emi/sim2real-object-detection

[8] In the case of one-shot transfer, in the training dataset there is one real image alongside the synthetic images.

Table 3.6: Summary of the InO-10-190 dataset.

ID	#Img	#Obj	#Obj / #Img	Illumination	Distractors
A	53	52	0.98	Normal	No
B	19	182	9.58	Normal	No
C	20	144	7.20	Spotlight	No
D	21	264	12.57	Normal	No
E	22	135	6.14	Normal	Yes
F	55	143	2.60	Normal	No
SUM	190	920	4.84		

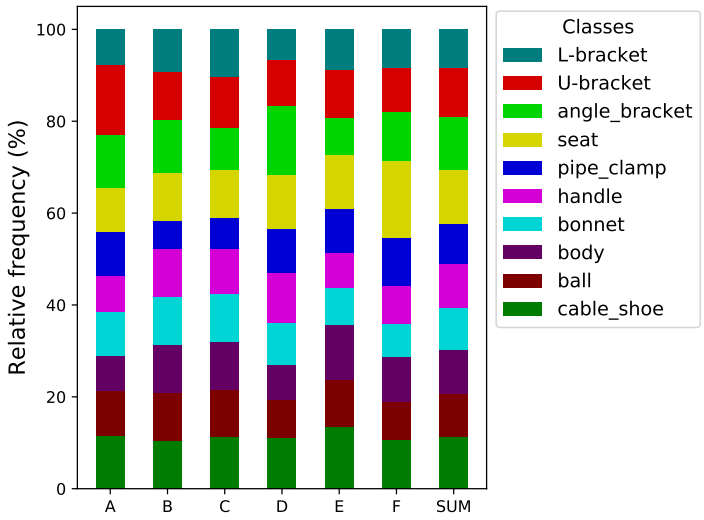


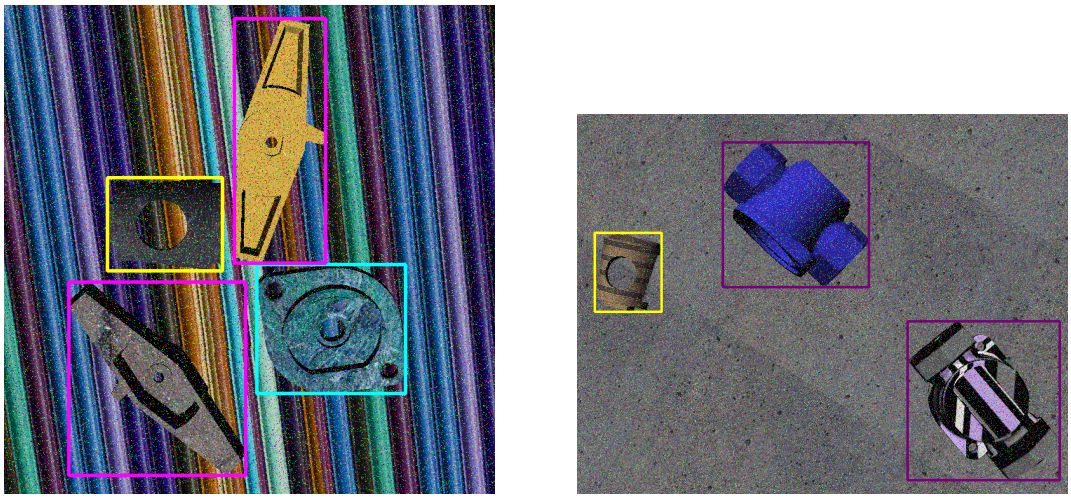
Figure 3.9: Class distributions of the InO-10-190 dataset.

The naming convention for the models is as follows: `TERM1_TERM2`$_{\text{index}}$, where `TERM1` refers to zero-shot (ZST) or one-shot transfer (OST) options. In the case of ZST, the model was trained only on synthetic images, while in the case of OST, one real image was utilised with the synthetic images. The second part, `TERM2` indicates the specific configuration of the S2R-ObjDet method, `BEST` stands for the configuration that performed the highest results. Finally, the lower index shows specific training seeds. For example `ZST_BEST`$_1$ stands for the 1^{th} run of a zero-shot transfer training with the best S2R-ObjDet configuration.

3.7.1 Zero-shot transfer (ZST)

The best-performing zero-shot transfer model (`ZST_BEST`$_1$) achieved 86.32% mAP$_{50}$ on the cropped test dataset. For data generation, a 2×2 grid ($n_{\text{grid}} = 2$) with fixed z positions and a placement disturbance of ±10% of the grid spacing was set in the horizontal directions. The simulation of gravity was enabled and the objects (including distractors and empty places) were selected with equal probability. The objects had random textures with a probability of 0.8 and a random colour with a probability of 0.2. The camera target position was set to the centre of the grid with a fixed 45° FOV. The pitch of the camera was randomised between −0.17 and 0.17 radians. The width and height of the images are chosen randomly, independently of each other. Their values lie between 640 and 1300 pixels. For post-processing, multi-colour pepper-and-salt noise and Gaussian blur were used with the probability of 1.0 and 0.5, respectively.[9]

With these parameters, 4000 images were generated for the training dataset, and 200 for the validation dataset. The evaluation of the model's performance on the training set was measured only on the first 200 images of the training set. Two examples of the synthetic images are shown in Fig. 3.10



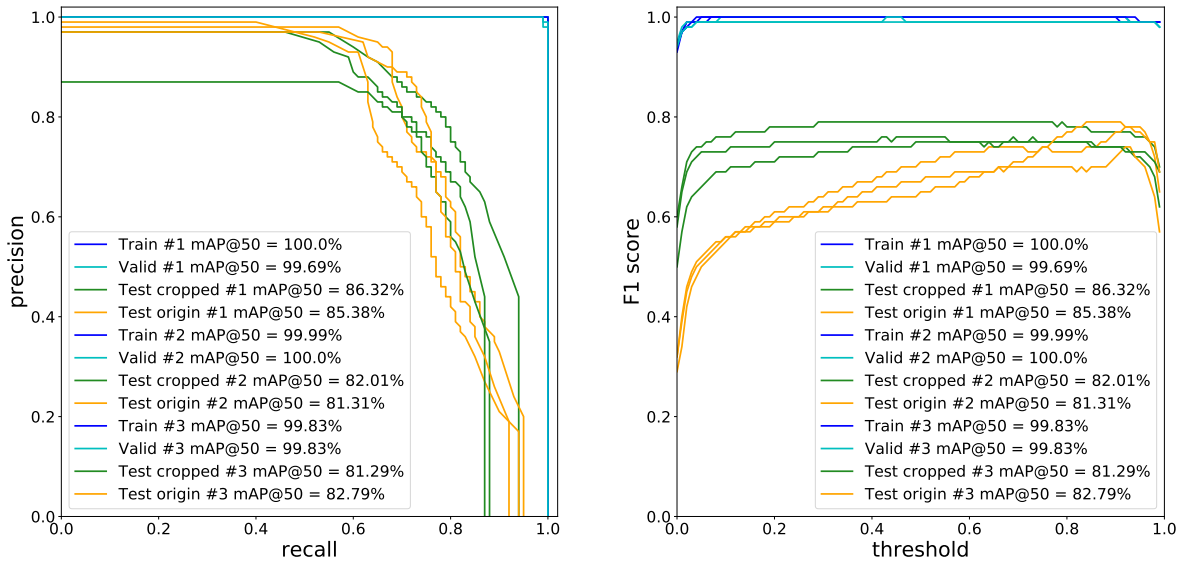
(a) Example 1 (b) Example 2

Figure 3.10: Two examples of synthetic images with the automatically generated annotations. The bounding boxes are shown here for illustration purpose only.

The precision-recall curves of the three `ZST_BEST` models (from the three training ses-

[9]The cutouts did not improve the performance, as shown in Section 3.8.5, thus they were not used here.

sions) are shown in Fig. 3.11a. As both the training and validation mAPs are close to 100%, it can be stated that the solution of the classical machine learning problem is satisfactory. Furthermore, observing the sim2real transfer, it can be seen that the models not only have a relatively good performance on the test data, but also have little variance. Moreover, the models perform relatively similarly on the original and on the cropped images which shows the robustness of the method. The F_1 score is depicted in Fig. 3.11b. It shows that while the performance of the model on the cropped images is not affected by the τ_{con} threshold, the models work better with higher τ_{con} values on the original images.



(a) The precision-recall curves of the `ZST_BEST` models.

(b) The F_1 scores of the `ZST_BEST` models.

Figure 3.11: Results of the `ZST_BEST` models. **Left.** The precision-recall curves. **Right.** The F_1 scores. The train and valid scores overlap and are relatively close to the perfect 100% score.

The performance of the models on the different groups of the test dataset (described in Tab. 3.6) is presented in Tab. 3.7. The performance is relatively stable across the different scene types. Group D, containing the most crowded images, performs just slightly worse than the others. In the case of the original images, group A has relatively low performance. This is due to the fact that the model occasionally falsely identifies the brackets of the camera holder frame (at the two sides of the camera holder frame which is displayed in Fig. 3.5, marked with letter 'F') as L-brackets – this is not surprising as these parts look

Table 3.7: The mAP_{50} scores of the ZST_BEST models in the different test groups.

	Train #1		Train #2		Train #3		AVG	
ID	Orig.	Crop.	Orig.	Crop.	Orig.	Crop.	Orig.	Crop.
A	82.43	95.71	75.85	84.86	83.97	87.00	80.75	89.19
B	89.27	86.48	85.70	80.26	86.00	81.18	86.99	82.64
C	85.02	82.41	86.69	82.27	83.97	79.11	85.23	81.26
D	84.50	83.94	82.54	80.07	79.89	76.58	82.31	80.20
E	87.22	85.05	82.99	84.91	82.02	83.52	84.08	84.49
F	86.03	95.11	79.63	87.50	88.41	91.68	84.69	91.43

similar. As group A has the lowest number of objects, this phenomenon has the most impact on results in this case. The cropped images, as shown in Fig. 3.5, do not contain this part of the image.

Furthermore, the performance of the models for the different classes is worth investigating. The data are presented in Tab. 3.8 and the average results are shown in Fig. 3.12. Looking at the dataset of cropped images (green), it can be seen that 6 out of 10 classes perform above 92%, one class is relatively close to them with 87.94% AP_{50}, two classes have worse results with 69.54% and 67.18% AP_{50}, and one class – the bonnet – has significantly worse performance with 30.72% AP_{50}. Otherwise, the performance on the validation dataset is close to 100% for all classes. The findings indicate that the performance loss is not caused by the unsuccessful solution of the classical machine learning problem, but by the existence of the reality gap. As most of the classes have relatively good APs, the bad classes are outweighed by them in the mAP calculation.

In order to investigate the aforementioned problem, the class-wise precision-recall graph of $\texttt{ZST\_BEST}_1$ is shown in Fig. 3.13a. As it can be seen, the bonnet, the L-bracket, and the seat are the worst classes consistently with Fig. 3.12.

The proposed generalised confusion matrix depicted in Fig. 3.13b (described in detail in Section 3.5) is essential in finding the root causes of the weaknesses of the models. In most cases, the objects are detected and classified to the correct class (diagonal). However, several instances of L-bracket and seat are not detected (36 and 49 examples) and many bonnets are classified as body objects (62 instances). The misclassification problem of the bonnet object is not surprising considering the high level of similarity of the two objects, as shown in Fig. 3.7. In general, this representation of the results not only confirms the aforementioned assumptions of class performances but also shows the underlying reason behind the lack of performance in their cases. It is worth considering why the bonnet-body

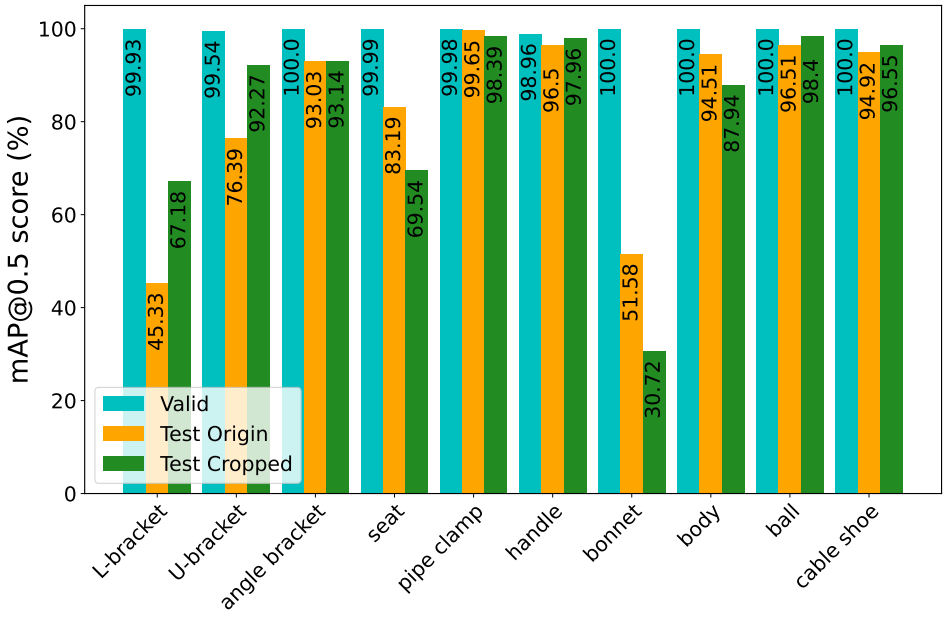


Figure 3.12: The average mAP_{50} scores of the `ZST_BEST` models in the different classes.

misclassification predominantly occurs in one direction (62 instances) and not the other (only 2 instances). On one hand, it seems that the learned representation of the body object is more inclusive than that of the bonnet. On the other hand, since both classes were correctly classified in the synthetic validation set, the misclassification could be a consequence of the sim2real transfer process – for example, the discrepancies between the 3D models and the real objects might significantly influence the knowledge transfer.

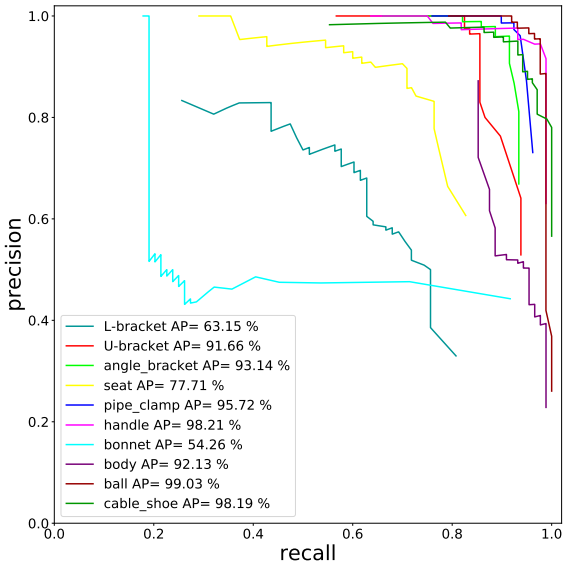
Finally, having presented the quantitative evaluation, two examples are given for qualitative evaluation as well. Fig. 3.14 shows an accurate and an inaccurate example, both with $\tau_{\text{con}} = 0.8$. While the model could accurately find and classify all the parts even in the presence of distractor objects in Fig. 3.14a, it fails to detect two instances of the seat class and assigns the bonnet object to the body class in Fig. 3.14b.

3.7.2 One-shot transfer (OST)

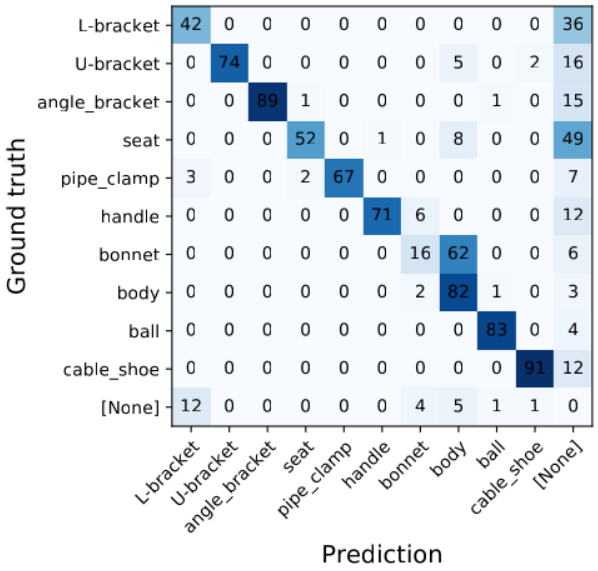
Even though the best zero-shot transfer method achieved 86.32% mAP_{50}, it had some difficulties with 4 classes. With one-shot transfer, we could overcome these difficulties. The parameters of data generation remained the same as it was described in the previous

Table 3.8: The mAP_{50} scores of the ZST_BEST models for the different classes.

	Train #1		Train #2		Train #3		AVG	
ID	Orig.	Crop.	Orig.	Crop.	Orig.	Crop.	Orig.	Crop.
1	41.45	63.15	45.26	71.85	49.28	66.54	45.33	67.18
2	76.34	91.66	77.51	91.58	75.32	93.56	76.39	92.27
3	94.14	93.14	91.56	93.17	93.39	93.11	93.03	93.14
4	84.96	77.71	71.88	60.16	92.72	70.74	83.19	69.54
5	99.82	95.72	99.37	99.49	99.76	99.97	99.65	98.39
6	95.77	98.21	94.82	97.53	98.90	98.14	96.50	97.96
7	71.77	54.26	52.39	22.01	30.57	15.90	51.58	30.72
8	96.51	92.13	93.77	92.23	93.26	79.46	94.51	87.94
9	97.13	99.03	94.27	96.89	98.14	99.27	96.51	98.40
10	95.87	98.19	92.32	95.22	96.57	96.24	94.92	96.55

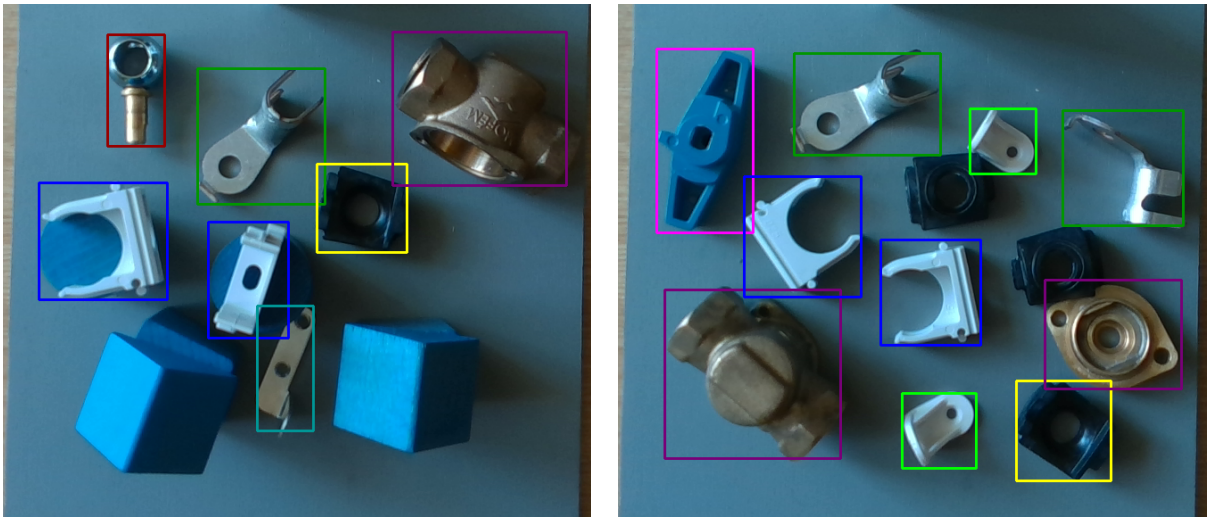


(a) Class precision-recall curves.



(b) Generalised confusion matrix.

Figure 3.13: Class specific results of the $\texttt{ZST\_BEST}_1$ model. **Left.** The precision-recall curves on the cropped images. **Right.** The GCM is evaluated on the cropped images with $\tau_{\text{con}} = 0.8$.



(a) Accurate example (b) Inaccurate example

Figure 3.14: Qualitative evaluation. **Left.** An accurate example. **Right.** An inaccurate prediction. Both are the results of the `ZST_BEST`$_1$ model with $\tau_{\text{con}} = 0.8$. The colour-coding follows Fig. 3.9.

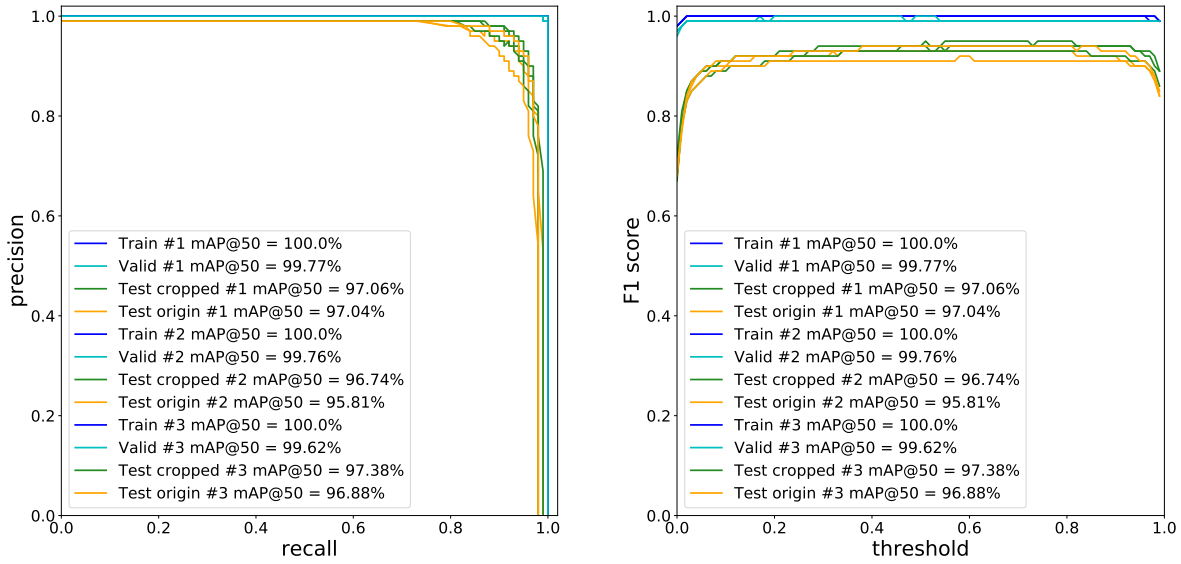
Table 3.9: Training datasets.

Model	Synthetic images	Real images
Zero-shot transfer (ZST)	4000	0
One-shot transfer (OST)	2000	1 (copied x2000)

zero-shot transfer example. The difference between the two approaches lies in the data used to train the model, which is shown in Tab. 3.9.

The `OST_BEST`$_3$ model achieved 97.38% mAP$_{50}$ on the cropped images, while the `OST_BEST`$_1$ model had 97.04% mAP$_{50}$ on the original images. These results are significantly better than the results with zero-shot transfer. The precision-recall curves are shown in Fig. 3.15a, and the F_1 scores are shown in Fig. 3.15b. The mAP scores are close to optimal and there is only an insignificant deviation between the different training sessions. The F_1 score is also relatively high and flat in all cases, indicating that the models are not sensitive to different τ_{con} thresholds.

The performance on the different types of test images is presented in Tab. 3.10. In general, the models work well, above 94% mAP$_{50}$ in each case. The crowded scenes (Group



(a) The precision-recall curves of the `OST_BEST` models.

(b) The F_1 scores of the `OST_BEST` models.

Figure 3.15: Results of the `OST_BEST` models. **Left.** The precision-recall curves. **Right.** The F_1 scores. The train and valid scores overlap and are relatively close to the perfect 100% score.

D) have slightly worse performance on average, but the difference is not significant.

Furthermore, the mAP scores of the different classes are presented in Tab. 3.11 and shown in Fig. 3.16. All classes perform well, the worst-performing class with the original images being the L-bracket with 92.62% mAP_{50}. The precision-recall curves of the $\texttt{OST\_BEST}_3$ for the different classes are depicted in Fig. 3.17a. Compared to the zero-shot approach, the curves are shifted towards the top-right corner which demonstrates better performance.

The proposed GCM of the $\texttt{OST\_BEST}_3$ with $\tau_{\text{con}} = 0.8$ is shown in Fig. 3.17b. Almost all the values are in the diagonal, meaning that they are good predictions. However, some instances of L-bracket, U-bracket, and cable shoe were not found by the model. The number of false negative examples can be decreased by lowering the threshold at the expense of possible false positive examples.

For qualitative evaluation, Fig. 3.18 shows an accurate prediction and an inaccurate solution. In the latter case, the distractor objects resembling a body object in main characteristics could mislead the model, implying that the model learnt an overly general rep-

Table 3.10: The mAP_{50} scores of the `OST_BEST` models in the different test groups.

	Train #1		Train #2		Train #3		AVG	
ID	Orig.	Crop.	Orig.	Crop.	Orig.	Crop.	Orig.	Crop.
A	97.29	97.52	97.76	97.29	98.00	97.76	97.68	97.52
B	98.12	96.45	97.29	97.02	97.92	97.49	97.78	96.99
C	96.39	96.19	94.18	96.40	96.21	97.18	95.59	96.59
D	96.23	95.94	94.97	94.42	95.60	95.64	95.60	95.33
E	96.70	98.94	93.77	97.47	93.34	97.91	94.60	98.11
F	98.99	99.37	98.83	99.20	99.76	99.52	99.19	99.36

Table 3.11: The mAP_{50} scores of `OST_BEST` models for the different classes.

	Train #1		Train #2		Train #3		AVG	
ID	Orig.	Crop.	Orig.	Crop.	Orig.	Crop.	Orig.	Crop.
1	93.70	93.94	90.24	92.70	93.92	94.92	92.62	93.85
2	94.75	92.45	92.94	93.21	95.28	94.15	94.32	93.27
3	93.19	94.85	92.95	93.25	92.61	94.22	92.92	94.11
4	96.76	96.93	96.74	97.11	96.65	96.70	96.72	96.91
5	99.34	99.54	99.19	99.84	98.38	99.88	98.97	99.75
6	99.51	99.57	99.17	99.71	99.40	99.53	99.36	99.60
7	98.30	98.81	98.51	98.81	97.69	98.77	98.17	98.80
8	98.50	98.40	96.94	98.00	98.23	98.45	97.89	98.28
9	99.52	98.43	97.74	98.47	99.96	99.53	99.07	98.81
10	96.83	97.73	93.70	96.26	96.73	97.60	95.75	97.20

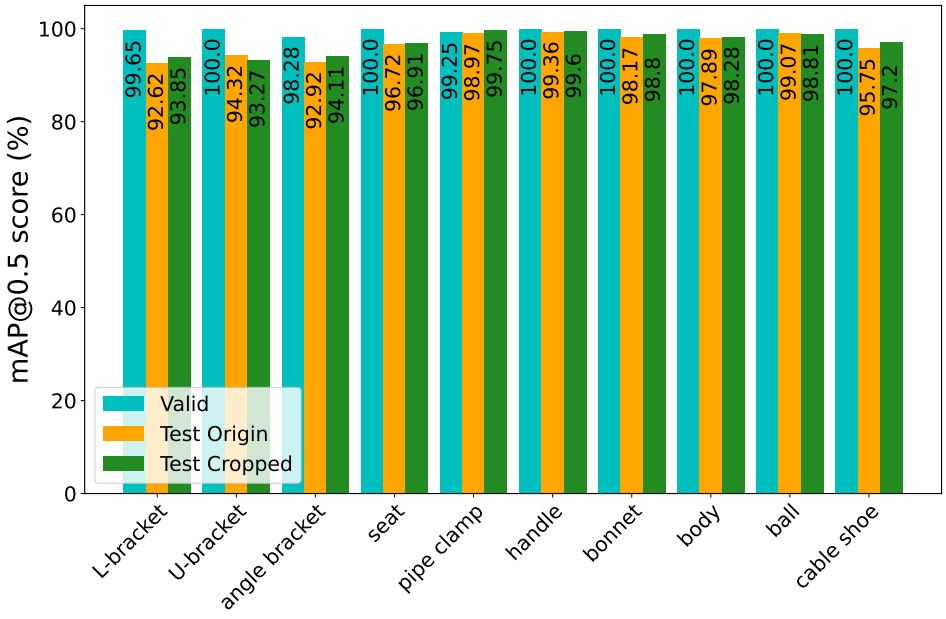


Figure 3.16: The average mAP_{50} scores of the `OST_BEST` models in the different classes.

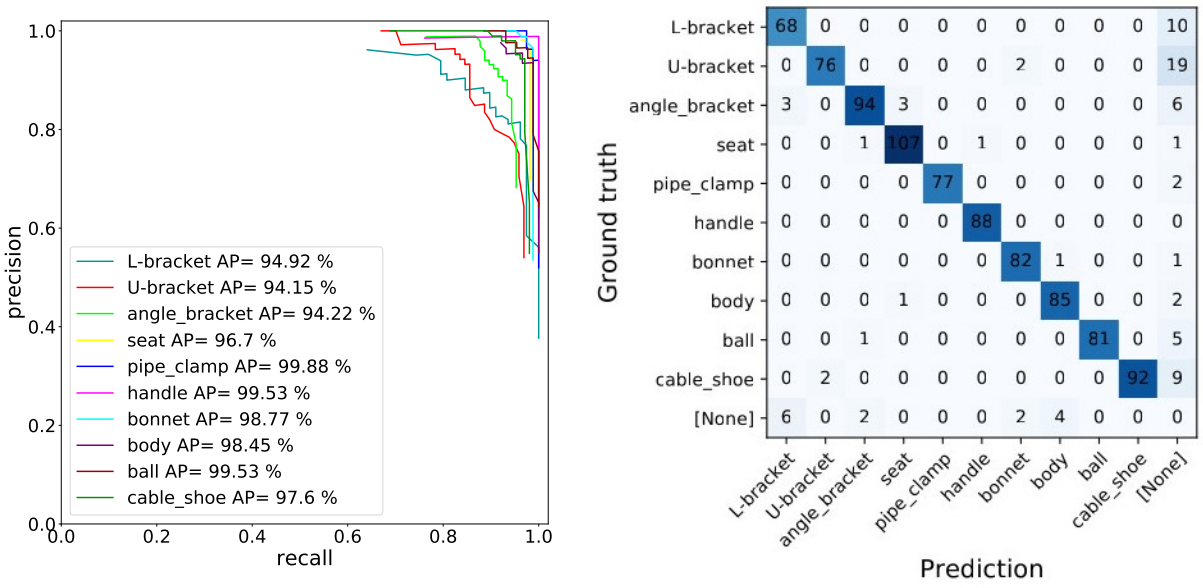
resentation of the object. As the quantitative results show, the vast majority of examples is accurate.

3.8 Ablation study

In this section, an ablation study is presented, focusing on the different elements of the domain randomization methods, the training data size, and the object detection model.

3.8.1 Seed

In general, the initial random seed of stochastic algorithms can significantly influence their performance. This phenomenon is unpleasant as it makes the algorithms unpredictable. We aim to measure the influence of the seed of our domain randomization method in the case of the `ZST_BEST` models. It is important to note that we do not use the same random seed for the training and for the domain randomization method. Tab. 3.12 shows different seeds (with two equal seeds for reference), with 3 independent training sessions



(a) Precision-recall curves.

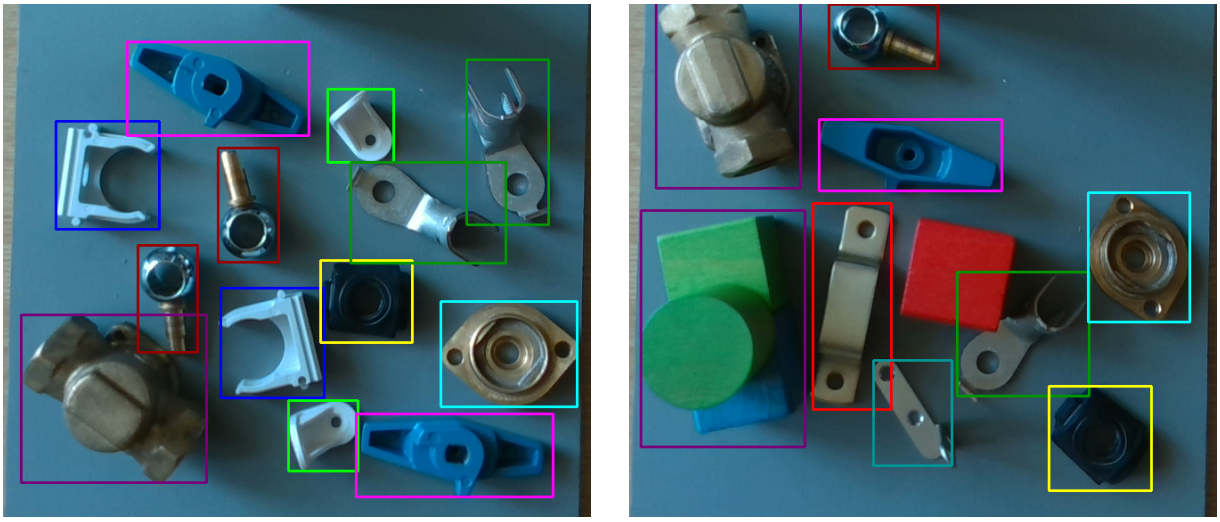
(b) Generalised confusion matrix.

Figure 3.17: Class specific results of the OST_BEST$_3$ model. **Left.** The precision-recall curves on the cropped images. **Right.** The GCM on the cropped images with $\tau_{\text{con}} = 0.8$.

each. In these experiments, we showed that the magnitude of the deviation of the results due to the stochastic training process of the neural network and due to the different seeds of the randomised data generation methods are comparable. Thus, our randomised data generation method is not less robust to a given seed than the stochastic training method itself.

3.8.2 Texture and post-processing

Two important factors in our domain randomization method are the random textures of the objects and the post-processing method. We have generated datasets without these factors. The results are summarised in Tab. 3.13 and the results on the original images are shown in Fig. 3.19. Both the added texture and the post-processing methods contribute significantly to the performance. Without the added texture, the performance drops to 63.50% and 74.71% mAP$_{50}$ in the case of the original and the cropped images. Without post-processing, the performance is only 55.87% and 60.42% mAP$_{50}$, respectively. Finally, the performance decreases drastically achieving 10.83% and 13.81% mAP$_{50}$ without the two methods. These experiments show how essential these types of domain randomization



(a) Accurate example (b) Inaccurate example

Figure 3.18: Qualitative evaluation. **Left.** An accurate example. **Right.** An inaccurate prediction. Both are the results of the `OST_BEST`$_3$ model with $\tau_{\text{con}} = 0.8$. The colour-coding follows Fig. 3.9.

methods are. As the average performance of the model on the validation dataset is 99.84% mAP_{50}, according to Eq. (3.8), the reality gap shrinks, in case of the original images, on average from 89.01% (`-TPP`) to 16.68% mAP_{50} (`BEST`).

3.8.3 Data size

The size of the training dataset is a key attribute of any machine learning problem. In general, the more data are used in the training, the better its distribution will match the

Table 3.12: The mAP_{50} scores of `ZST_BEST` models with different seeds.

	Train #1		Train #2		Train #3		AVG	
Seed	Orig.	Crop.	Orig.	Crop.	Orig.	Crop.	Orig.	Crop.
80725	79.71	78.67	81.93	84.55	78.30	80.69	79.98	81.30
80725	82.29	78.31	80.99	84.31	79.60	79.23	80.96	80.62
53418	83.57	80.69	78.13	80.95	74.74	76.92	78.81	79.52
16505	85.38	86.32	81.31	82.01	82.79	81.29	83.16	83.21

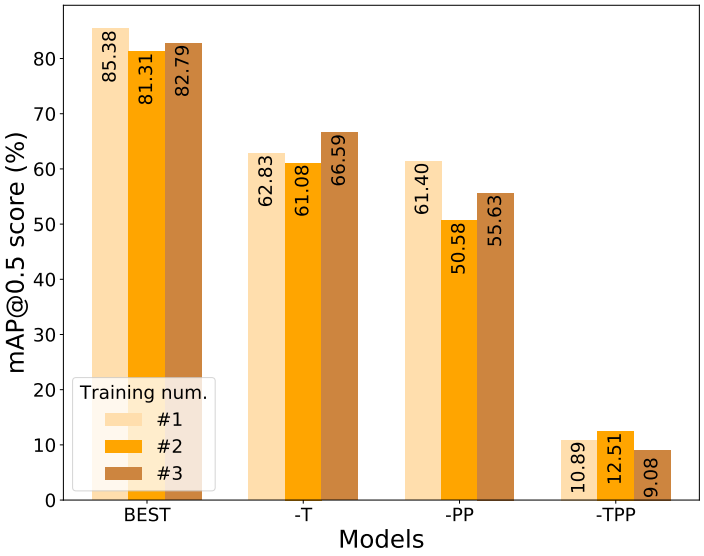


Figure 3.19: Results of the ablation study on the original images (`ZST` models). Model without added textures (`-T`), without post-processing methods (`-PP`), and without both (`-TPP`).

real probability distribution. Nevertheless, this phenomenon does not necessarily apply to the case of knowledge transfer. The results of the performance of the `ZST_BEST` models for different training data sizes are presented in Tab. 3.14. It is important to note that for the case of 8000 images, the training time was doubled from 5000 to 10000 iterations. Even though increasing the training data size from 1000 to 4000 allows the model to gain notable performance, doubling the data size from 4000 to 8000 only causes marginal improvement.

3.8.4 Gravity, positional disturbance, and bounding-box calculation

In this part of the ablation study, the effect of simulated gravity, the effect of random disturbance around the grid positions, and the effect of replacing the all-point bounding box calculation with the 8-point bounding box calculation (described in Section 3.3.2) are measured. The findings are summarised in Tab. 3.15. All of the aforementioned factors have a relevant effect on the performance. In the case of cropped images, on average, gravity brings 11.01% mAP_{50}, the randomness of object positions contributes 14.22% mAP_{50}, and the tight all-point bounding box calculation method is responsible for a 41.76% mAP_{50}

Table 3.13: The mAP_{50} scores of different ZST models. -T: without texture, -PP: without post processing, -TPP: without post processing and texture.

	Train #1		Train #2		Train #3		AVG	
Desc.	Orig.	Crop.	Orig.	Crop.	Orig.	Crop.	Orig.	Crop.
BEST	85.38	86.32	81.31	82.01	82.79	81.29	83.16	83.21
-T	62.83	73.26	61.08	74.07	66.59	76.80	63.50	74.71
-PP	61.40	63.14	50.58	60.62	55.63	57.51	55.87	60.42
-TPP	10.89	9.48	12.51	13.75	9.08	18.21	10.83	13.81

Table 3.14: The mAP_{50} scores of ZST_BEST models with different data sizes.

	Train #1		Train #2		Train #3		AVG	
Size	Orig.	Crop.	Orig.	Crop.	Orig.	Crop.	Orig.	Crop.
1000	70.07	72.12	77.94	76.60	75.37	78.12	74.46	75.61
4000	85.38	86.32	81.31	82.01	82.79	81.29	83.16	83.21
8000	84.56	85.64	86.95	83.68	81.96	81.46	84.49	83.59

performance gain. In the case of bounding box calculation, the performance drops with the less tight BBs, implying two possible reasons. Firstly, the ground-truth BBs are tight, thus computing the IoU_{50} with less tight BBs may result in many discarded matches. Secondly, in the crowded images, the BBs are too extensive, thus, they could significantly overlap each other which may confuse the model.

Table 3.15: The mAP_{50} scores of ZST models without different factors. -G: no gravity, -R: no randomness in grid positions and no gravity, 8P: 8-point bounding box calculation.

	Train #1		Train #2		Train #3		AVG	
ID	Orig.	Crop.	Orig.	Crop.	Orig.	Crop.	Orig.	Crop.
BEST	85.38	86.32	81.31	82.01	82.79	81.29	83.16	83.21
-G	61.12	71.47	65.86	69.89	68.76	75.25	65.25	72.20
-R	49.12	63.07	62.00	74.38	48.18	69.53	53.10	68.99
8P	31.09	40.66	28.98	37.77	39.19	45.93	33.09	41.45

Table 3.16: The mAP_{50} scores of ZST models with different cutouts at the post-processing method.

	Train #1		Train #2		Train #3		AVG	
ID	Orig.	Crop.	Orig.	Crop.	Orig.	Crop.	Orig.	Crop.
BEST	85.38	86.32	81.31	82.01	82.79	81.29	83.16	83.21
CUT1	68.24	77.18	66.93	76.77	71.85	83.53	69.01	79.16
CUT2	82.12	78.97	78.56	83.33	72.99	69.79	77.89	77.36
CUT3	58.25	75.13	56.39	74.45	58.97	76.46	57.87	75.35
CUT4	76.00	69.76	70.17	78.62	68.26	73.60	71.48	73.99
CUT5	70.32	74.74	75.50	78.74	74.59	74.91	73.47	76.13
CUT6	78.53	76.86	74.83	79.77	79.70	82.27	77.69	79.63
CUT7	82.25	80.26	70.22	69.95	74.51	77.48	75.66	75.90
CUT8	75.64	75.63	75.58	80.46	80.96	84.06	77.39	80.05

3.8.5 Cutouts

Some additional experiments were conducted with different types of cutouts at the post-processing phase of data generation presented in Tab. 3.16. In these experiments, 4 types of cutouts were considered: rectangles, partly transparent rectangles, circles, and lines. The number of cutouts and the bounds of the randomised sizes of the cutouts varied over the experiments. The results show that none of the cases could achieve a better performance than the ZST_BEST model which does not have this type of domain randomization. Nevertheless, a more thorough evaluation of the effect of different cutouts can be subject to further research.

3.8.6 Faster R-CNN

To test the performance of the data generation process in the case of a two-stage object detection model, we trained the R101-FPN version of the Faster R-CNN [120] model using the Detectron2 [227] framework and Pytorch [228]. This model uses the ResNet-101 [131] model with the Feature Pyramid Network [229] backbone. The results are shown in Table. 3.17. Even though the performance of the model falls behind YOLOv4, it could be increased with a more exhaustive hyperparameter search. Moreover, the Darknet framework uses extra data augmentation for training which we did not reproduce for the Detectron2 framework. It is important to note that here, too, considerable performance improvement is achieved by having one real image for training.

Table 3.17: The mAP_{50} scores of R101-FPN Faster R-CNN model.

	Zero-shot transfer		One-shot transfer	
	valid	test	valid	test
Train #1	85.90	49.78	97.18	71.97
Train #2	85.78	66.73	96.495	67.70
Train #3	86.38	55.41	96.78	70.73
AVG	86.02	57.31	96.82	70.13

The training of the Faster R-CNN model was approximately 4 times faster than in the case of YOLOv4. On the other hand, inference time was around 10 times slower with 2 FPS. In conclusion, YOLOv4 outperformed the Faster R-CNN approach in performance and in inference time which are the two most relevant factors.

3.9 Robotic application

Object detection can be utilised in many ways. Examples are robotic grasping or pick-and-place applications where the robot needs to detect different workpieces and grasp them or move them to specific locations.

In this section, a real-world robotic implementation of our method is presented. The application can serve as proof of concept built upon our previous work [5], where we proposed a 5C model-based [230] system architecture for visual-servo-guided cyber-physical robotic assembly cells. Relying on the object detection model, the parameters of grasping (micro plan) can be computed.

The robotic system consists of a 6 DoF collaborative robot arm equipped with a digital depth camera, a force sensor, and a two-finger gripper. The task of the robot is to detect scattered workpieces (centre points and bounding box information), as well as predict grasping poses. The sensors and actuators of the robot and the sim2real computer vision module are connected in a robot control framework based on ROS [231]. The setup is depicted in Fig. 3.20, while the software components of the robot control framework are shown in Fig. 3.21

For robotic applications, every component of the system must work reliably in real time. In order to evaluate the sim2real computer vision module in a new case, three new industrial parts were used, as depicted in Fig. 3.20. The data generation and training process went without any problems. Thus, within 13 hours, the new model was ready

to use. As a qualitative evaluation, the robot was programmed to follow a path over the workpieces while streaming the camera data. On a GeForce RTX 3060 GPU, our computer vision model ran with 20 FPS and constantly localised and classified the objects perfectly with more than 98% confidence most of the time, even in significantly different illumination settings and in the presence of distractor objects.

For grasping the workpieces, the grasping pose needs to be estimated and transformed into the robot coordinate system. These problems are addressed, in detail, in the following chapter. Here, the method of principal component analysis was applied for a simplified version of grasp pose estimation and a standard camera calibration method for the transformation.

This use case was presented in an exhibition[10] and the implementation of the ROS-based robot control system is available at the control module's git repository[11].

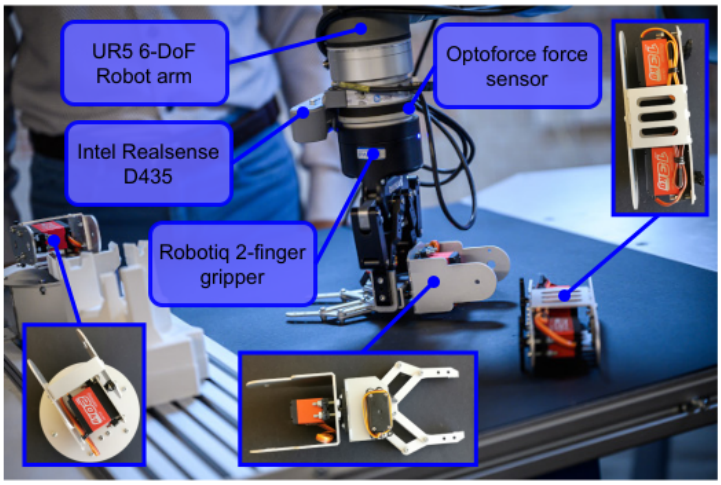


Figure 3.20: The setup of the robotic application.

3.10 Conclusion

In this chapter, our sim2real domain randomization (S2R-ObjDet) method for object detection was presented alongside our generalised confusion matrix (GCM) for performance evaluation and our InO-10-190 dataset of real images.

[10] https://youtu.be/6PhaXW1m9Xw
[11] https://git.sztaki.hu/emi/robot_control_framework

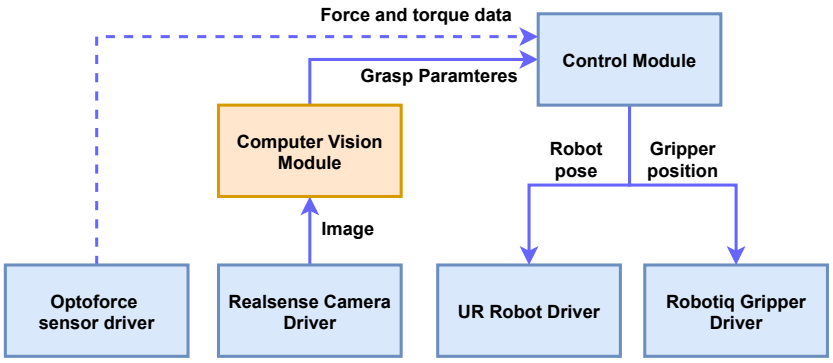


Figure 3.21: Information flow in the robotic application. The computer vision module, which is the main topic of this chapter, is highlighted in orange. In this application, the force sensor was not used (marked with the dashed line). The following software resources were used: [232]–[236].

For industrial usability, our S2R-ObjDet method is capable of detecting different classes and also differentiating among similar classes. Additionally, it works in real time, and the data generation takes less than 0.5s per image. According to our best knowledge, this is the first work thus far that mutually satisfies these constraints in this domain.

As recent works on transfer learning did not concentrate on object similarity (and diversity), we created the InO-10-190 dataset with 190 real annotated images of 920 objects of 10 classes of industrial workpieces. The dataset is publicly available and can serve as a benchmark for industrial object detection models.

Furthermore, we introduced the generalised confusion matrix (GCM) which is capable of quantifying misclassification, false positives, and false negatives in object detection problems. It has proven to be essential for finding the root cause of performance loss in the sim2real transfer.

The results presented in the chapter validate the strengths of our approach. We achieved 86.32% mAP_{50} in the case of zero-shot transfer, while with one-shot transfer, the best model scored 97.38% mAP_{50} on the test set. With these experiments, we also demonstrated how to diminish the performance loss caused by similar classes by introducing only one image from the target domain.

In a thorough ablation study, we showed that adding random texture and our post-processing domain randomization methods are crucial parts of the process. We also found that simulating gravity, random initial placement, and the all-point bounding box calcu-

lating method contribute significantly to the performance.

As a proof of concept, we showed that our model works reliably and in real time in a robotic pick-and-place application.

Both the sim2real data generation and training module, and the robot control framework can be used as a freely available, out-of-the-box solution to industrial problems.

Based on the findings presented in this chapter, our theses connected to the first research question regarding sim2real knowledge transfer for object detection are as follows:

Thesis I: The synthetic images generated by our sim2real domain randomization method (S2R-ObjDet) enable object detection models to learn general representations of the objects, thereby bridging the gap between simulation and real-world environments.

We propose S2R-ObjDet, a domain-randomization-based sim2real synthetic data generation method for object detection. The 3D models of the given objects are loaded in the simulator, each with a random texture or monochromatic colour. Both the number and types of objects are randomised. Simulating gravitational force, the objects are dropped to a plane where they end up in one of their stable positions. The camera extrinsic and intrinsic parameters are set randomly with some constraints to ensure that the given objects are in the field of view. After an image is rendered, a post-processing method is applied to it involving multi-colour pepper-and-salt noise, gaussian blur, and optionally rectangular, circular, and line cutouts. The ground truth annotations of each object are automatically computed based on all points of the objects instead of the 8-points of the axis-aligned bounding boxes of the objects. This process is repeated until the required number of images for the training dataset is generated. S2R-ObjDet is capable of shrinking the reality gap between simulation and the real world to a satisfactory level, achieving 86.32% and 97.38% mAP_{50} scores respectively in the case of zero-shot and one-shot transfers, on our publicly available manually annotated InO-10-190 dataset, containing 190 real images of 920 object instances of 10 classes. The class selection was simultaneously based on different and similar objects in order to test the robustness of the model in terms of detecting different classes and differentiating between similar objects. Our solution fits industrial needs as the data generation process requires less than 0.5s per image enabling a fast training process. The training pipeline is presented in Fig. 3.1. This thesis is associated with [1].

Thesis II: In object detection, misclassifications, false positives, and false negatives – factors not captured by traditional metrics – can be effectively quantified and evaluated using our generalised confusion matrix (GCM).

Our novel generalised confusion matrix (GCM) – depicted in Fig. 3.4 – is an adaptation of the classical confusion matrix to object detection. It addresses the limitations of the traditional precision-recall-based mAP and F_1 scores. Using the GCM, errors from misclassification, false positives, and false negatives can be effectively quantified and evaluated. Compared to the traditional confusion matrix $\boldsymbol{D} \in \mathbb{N}^{C \times C}$, where $C \in \mathbb{N}$ is the number of the classes, in our GCM $\boldsymbol{D}^{gen} \in \mathbb{N}^{C+1 \times C+1}$, one extra row and one extra column are added to the false positives and the false negatives cases. The correct detections are in the diagonal, $D^{gen}_{i,i}$, as in the case of the standard confusion matrix. $D^{gen}_{C+1,C+1} \doteq 0$. This thesis is associated with [1].

Chapter 4

Sim2real grasp pose estimation

Contents

4.1 Introduction 86
4.2 Problem statement 88
4.3 Related works 89
4.4 Approach 90
4.4.1 Object detection with S2R-ObjDet 90
4.4.2 ROI cropping 91
4.4.3 Orientation estimation with S2R-PosEst 91
4.4.4 Pattern matching (optional) 93
4.5 Robot control architecture 94
4.6 Results 96
4.6.1 Setting of the robotic experiments 96
4.6.2 Object detection 97
4.6.3 Orientation estimation 97
4.6.4 Robotic grasping 98
4.7 Conclusion 99

In Chapter 3, our sim2real domain randomization method was presented, focusing on object detection. Nevertheless, in robotic manipulation tasks, oftentimes, the poses of the objects need to be estimated as well. Even though the recent DL models show promising results, they require an immense dataset for training. In this chapter, we propose two vision-based, multi-object grasp pose estimation models (MOGPE) – the real-time MOGPE-RT and the high-precision MOGPE-HP. Furthermore, to diminish the reality gap and overcome the data shortage, our sim2real domain randomization method, presented in Chapter 3, is augmented to pose estimation (S2R-PosEst). Our methods yielded an 80% and a 96.67% success rate in a real-world robotic pick-and-place experiment, with the MOGPE-RT and the MOGPE-HP model respectively, using only limited real-world data. Our framework provides an industrial tool for fast data generation and model training, requiring minimal data from the target distribution. For a short presentation and additional materials, we refer the reader to the project page: www.danielhorvath.eu/mogpe.

4.1 Introduction

Building on the challenges posed by data-hungry DL models and their potential solutions discussed in Section 3.1 within the context of object detection, this Chapter shifts focus to pose estimation in the domain of robotic grasping

Robotic grasping is an unsolved problem and a critical challenge of adaptive robotics in which the model not only needs to identify and locate the different parts but also estimate its orientation to compute a viable grasp position. The contributions of this chapter are as follows:

- The proposed multi-object grasp pose estimation methods (MOGPE) – the real-time MOGPE-RT and the high-precision MOGPE-HP models – depicted in Fig. 4.1.
- The synthetic data generation process with sim2real domain randomization for grasp pose estimation (S2R-PosEst).
- Our freely available implementation of the S2R-PosEst method[1] and the robot control framework[2].

Our results:

[1] https://git.sztaki.hu/emi/grasping-pose-estimation
[2] https://git.sztaki.hu/emi/robot_control_framework

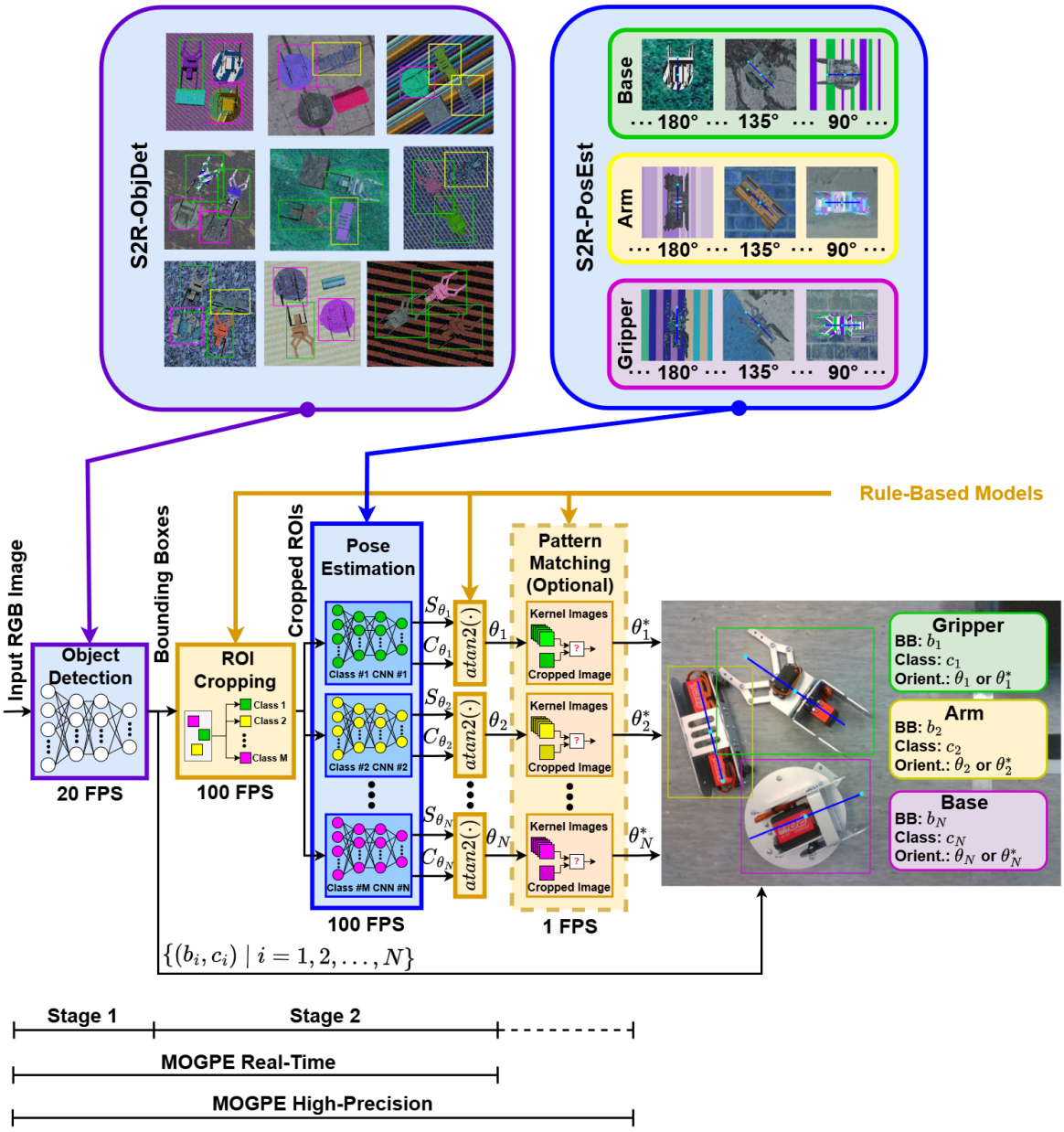


Figure 4.1: **Top.** Illustration of our S2R-ObjDet and S2R-PosEst methods. **Bottom.** The flowchart diagram of our multi-object grasp pose estimation (MOGPE) methods.

- In our case study, the object detection model with S2R-ObjDet yielded a 98.78% mAP_{50} score, while the orientation estimation models with S2R-PosEst achieved a 97.04% success rate on average.
- The (MOGPE-RT) model runs in real time. The object detection stage works at 20 FPS while the orientation estimation stage runs at 100 FPS.
- In a real-world experiment of robotic grasping, the MOGPE-RT model achieved an 80% while the MOGPE-HP model accomplished a 96.67% success rate. These results serve as a proof-of-concept of our approach.

4.2 Problem statement

In this section, the problem is briefly presented alongside our approach. For a more thorough overview of the field, the reader is referred to survey articles such as [237].

The problem is defined as a 3.5 DoF pick-and-place robot manipulation task. In Section 2.1.5, the output of an axis-aligned object detection model is described as $\boldsymbol{y} = \{(\boldsymbol{b}_i, c_i^{\text{class}}, p_i^{\text{con}}) \mid i = 1, 2, \ldots, N\}$, where $\boldsymbol{b}_i = [x_i, y_i, w_i, h_i]$ represents the AABB, c_i^{class} is the class label, and p_i^{con} is the confidence score of the i^{th} detection, while $N \in \mathbb{N}$ is the number of detected objects. One approach to introduce the orientation to object detection is to replace the AABBs to OBBs, however – for simplicity and transferability – we chose to keep the AABBs and augmented them with orientation angles. Mathematically, $\boldsymbol{y} = \{(\boldsymbol{b}_i, c_i^{\text{class}}, \theta_i) \mid i = 1, 2, \ldots, N\}$, where $\theta_i \in \mathbb{R}$ is the orientation angle of the i^{th} detection[3].

In our case, the characteristics of the problem are as follows. The position of the plane where the objects are placed must be known. The objects are recognised only from one of their stable positions. The parts are separated and all object classes are present at the training. Lastly, 3D models of the objects are available[4].

The given model needs to identify and locate all the different workpieces and then estimate the orientations of them. Additional challenges arise from the following circumstances. The environment is not controlled (no special illumination), and the background is not simplified (no monochromatic background). The model has access to only one RGB

[3]It is important to note that with this formulation, the confidence scores are already filtered. For more detail, see Section 4.4.

[4]In our solution, the 3D models of the objects are not used for 3D pattern matching but for synthetic data generation.

image, thus the 3D reconstruction of the scene is not possible. The grasp must be performed with a two-finger gripper and every object has only one grasp position.

Our solution is a two-stage, data-driven (supervised learning) method. 3D models of the objects are only utilised for the synthetic training dataset generation. As our aim is industrial usability, the assumption is that the availability of real-world data is limited. The majority of the training dataset is synthetic, generated by our sim2real domain randomization methods.

4.3 Related works

The related works focus mostly on two aspects of the robotic grasping challenge:

1. What is the optimal model to solve the problem?
2. How to generate training data and then transfer the knowledge to the real world?

Mahler et al. [52] introduced Dex-Net 4.0. They use a simulator to create a training dataset for their Grasp Quality Convolutional Neural Network. Even though this approach is relatively strong in bin-picking tasks, it is less optimal for pick-and-place operations with predefined grasping positions.

Tobin et al. [50] propose an autoregressive grasp planning method that gives a probability distribution over possible grasps. They used the YCB [221] dataset and in a real-world scenario, they achieved an 80% success rate.

Pashevich et al. [184] trained a model to learn manipulation policies in a simulation using depth images and sim2real transfer. They achieved 1.09 ± 0.73 cm positional error in the real world. Furthermore, in the tasks of cube picking, cube stacking, and cube placing tasks, they yielded 19, 18, and 15 successful attempts out of 20.

Zhang et al. [211]. presented how to efficiently transfer visuo-motor policies from simulation to real-world. In their case study, a velocity-controlled 7 DoF robot arm needed to reach a blue cuboid object in a table-top scenario. They achieved a 97.8% success rate and 1.8 cm control accuracy.

Zhang et al. [238] introduced a two-stage ROI-based robotic grasp detection model focusing object overlapping scenes. They yielded 92.5% and 83.8% success rate, respectively in single-object and multi-object scenes. Nevertheless, using real images, they did not focus on sim2real knowledge transfer.

It is challenging to compare the works above as many aspects of the problem are different. However, it can be noted that an 80% success rate is considered a good performance in the literature.

It is important to mention that, according to our best knowledge, even though there are existing industrial solutions for some types of robotic grasping, they cannot perform the task described in Section 4.2. In general, these tools either detect a tag on a palette and then move to predefined positions on the palette, exploit the controlled environment (such as special background or lightning conditions), are only capable of detecting one class of objects, or use many real-world images. For the aforementioned reasons, the comparison of such solutions is not feasible.

4.4 Approach

In our multi-object grasp pose estimation (MOGPE) methods, the problem is divided into two stages, depicted in Fig. 4.1. The first stage is object detection while the second stage is class-specific orientation estimations. As the plane coordinates and the 3D models of the objects are known, with the centre points and the orientations, the grasping position can be calculated. The learning-based models were trained primarily on synthetic data, generated by our sim2real methods.

The two main building blocks of the MOGPE models, the object detection model with S2R-ObjDet, and the orientation estimation model with S2R-PosEst are presented in Section 4.4.1 and 4.4.3. In Section 4.4.2, the *region of interest* (ROI) cropping algorithm is presented which connects the object detection and the orientation estimation models. The aforementioned blocks constitute the MOGPE-RT model. The MOGPE-HP model is an extension of the MOGPE-RT model with a rule-based pattern matching step, described in Section 4.4.4. The implementation is available at our git repository[5].

4.4.1 Object detection with S2R-ObjDet

In this section, the essentials of the S2R-ObjDet for MOGPE is presented. However, S2R-ObjDet is introduced and detailed in Section 3.3, thus for further details, we refer the reader to Chapter 3.

Depicted in Fig. 4.1, the object detection module is the first stage of our MOGPE method. Its input is the image itself and its output is described as $\boldsymbol{y} = \{(\boldsymbol{b}_i, c_i^{\text{class}}, p_i^{\text{con}}) \mid$

[5] https://git.sztaki.hu/emi/grasping-pose-estimation

$i = 1, 2, \ldots, N\}$. The detections with $p_i^{\text{con}} < \tau_{\text{con}}$ are filtered out. For further details on the problem formulation, the confidence score, and the confidence threshold, we refer the reader to Section 2.1.5.

For the object detection convolutional neural network, YOLOv4 [34] was chosen as, at the time of the research, it had the optimal accuracy-speed trade-off compared to the state-of-the-art. The generation of the synthetic training dataset and the training process of S2R-ObjDet are described in Section 3.3. Nevertheless, the detected objects are different compared to the InO-10-190 dataset, presented in Section 3.7, thus our results presented in this chapter also serve as further validation to our S2R-ObjDet method.

For industrial usability, it is important to note that the data generation lasts around 0.25 - 0.5s per image, while the training takes 12h on a GeForce RTX 2080 Ti GPU. The model prediction time is above 20 FPS on a GeForce RTX 3060 GPU.

4.4.2 ROI cropping

Between the object detection and the orientation estimation models, there is a rule-based ROI cropping algorithm that cuts out the specific ROIs of the objects from the input image according to the bounding box information. Then, it transforms them to the appropriate size while keeping the orientation of the objects (one object per image) and forwards them to the specific CNN of the second stage, depicted in Fig. 4.1 (after the object detection module) and detailed in Fig. 4.2. Assuming that there are $C \in \mathbb{Z}^+$ classes, an object that is detected on the image can be sent to C different CNNs.

As the next stage must estimate the orientation of the objects, it is crucial that the image transformation does not change the orientation. For this reason, the image is padded with zeros to a square and then resized to the expected input size of the neural network. In our case, it is 300x300 pixels.

4.4.3 Orientation estimation with S2R-PosEst

The second stage of the model is the orientation estimation which contains C CNNs (one for each class)[6]. Each of them takes a 300x300x3 image as input and outputs the sine and cosine representations of the $\theta \in [-\pi, \pi]$ orientation. Learning the sine and cosine values, rather than directly learning the angles, was chosen because these trigonometric functions

[6]The CNN architecture was chosen as it is easier to implement and train than ViT models.

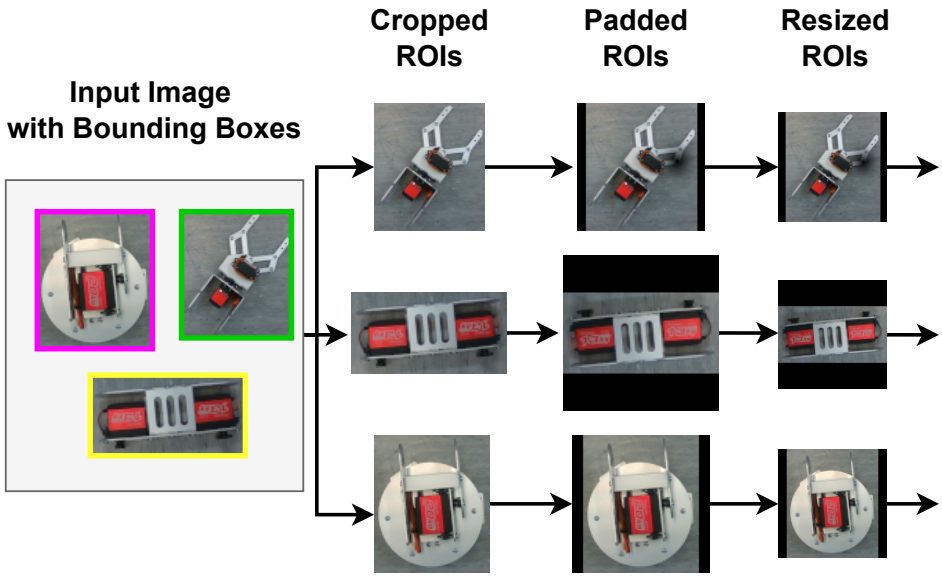


Figure 4.2: The data flow of the ROI cropping method.

are continuousan essential property for optimizing the loss function in regression problems. Having computed these values, the orientation can be calculated using the `atan2` function.

The architecture of the CNNs is shown in Fig.4.3. In the feature extractor, there are 4 convolutional layers with ReLU activation functions and each of them is followed by a MaxPooling layer. To compute the outputs, there are 4 fully connected layers in the head of the network. The models are trained from scratch, independently from each other, on class-specific synthetic and real examples.

The synthetic data were generated in PyBullet. The 3D model of the object is placed in the simulator and rotated around the z-axis (perpendicular to the plane where the object is placed) while random textures are added to the plane and to the object as well. All together, there are n_{rot} rotations. Mathematically, $n_{\text{rot}} = \lfloor \frac{2 \cdot \pi}{\beta_{\text{res}}} \rfloor$, where $n_{\text{rot}} \in \mathbb{N}$ is the number of rotation and $\beta_{\text{res}} \in \mathbb{R}$ is the resolution in radian. For each bit of rotation, an image is taken and the label is automatically generated with it. Some examples can be seen in Fig. 4.4. The data generation lasts around 0.25 - 0.5s per image, while the training, implemented in PyTorch, takes 2.5h per class on a GeForce RTX 2080 Ti GPU. The model prediction time is at 100 FPS on a GeForce RTX 3060 GPU.

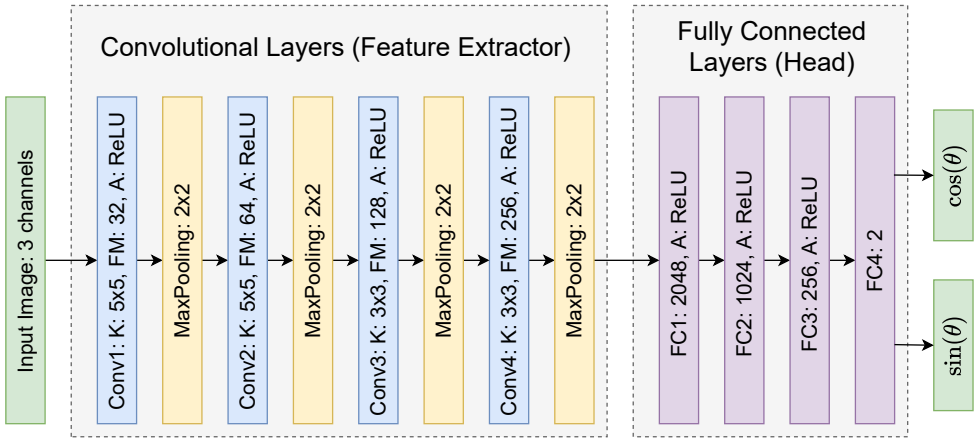


Figure 4.3: The proposed CNN architecture for orientation estimation. Abbreviations are Conv: convolutional layer, FC: fully connected layer K: kernel size, FM: number of feature maps, A: activation function.

4.4.4 Pattern matching (optional)

Computing the θ orientations for all the objects is the last step of the real-time MOGPE-RT model. To achieve higher precision, we also propose the high-precision MOGPE-HP model, which incorporates a rule-based pattern-matching algorithm executed following the orientation estimation, albeit at the expense of real-time performance. The pattern matching is performed locally, in the vicinity of the estimated orientation. With this addition, the model achieves higher precision at the cost of the extra computation.

The pattern-matching algorithm compares the image of the object with a set of pre-computed rotated kernel images. For one class, one real kernel image is rotated 359 times making 360 rotated kernel images[7].

Comparing two images takes around 13 ms, thus if the search is restricted for $\pm$ 10 degrees with a 1-degree resolution, it takes 0.29 seconds (including the angle zero). It is important to note that performing it in the whole range (without the orientation estimation by the CNN) would take 4.68 seconds. Moreover, this process must be performed for all detected objects. If parallel processing is not achievable, it can result in a substantial

[7]If the precision needs to be higher than 1 degree, this procedure can be done on a finer scale.

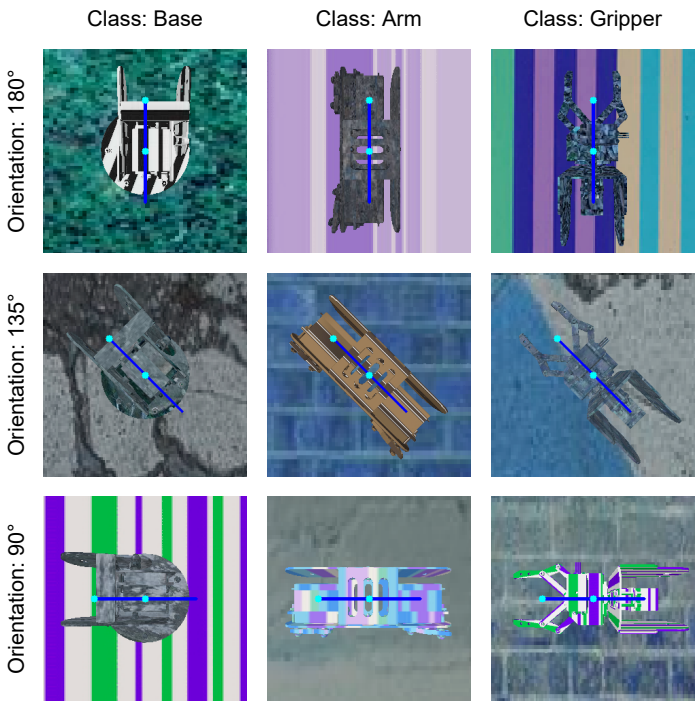


Figure 4.4: Examples of the generated synthetic training dataset.

increase in execution time.

It is important to note that the pattern-matching algorithm needs a good initialization, provided by the orientation estimation CNN. Otherwise, it frequently finds wrong orientations, especially in symmetric objects.

4.5 Robot control architecture

In this section, the robot control architecture is presented which shows how our computer vision models can be utilised in real-world robotic applications.

The robot control architecture is based on ROS (robot operating system) and is depicted in Fig. 4.5. The `camera driver` node publishes the images that are first read by the `object detection` node which then publishes the bounding box information. Based on these, the `orientation estimation` node predicts the orientation of the visible objects and sends this information to the `pattern matching` node which returns with the corrected

orientation estimate when the `get orientation service` is called. In the case of the MOGPE-RT model, the `get orientation service` returns with the original value of the `orientation estimation` node. The `camera frame broadcaster` node publishes the transformation between the camera frame and the end effector. With this information, the `pixel converter` node transforms pixel coordinates to the word frame when the `convert point service` is called. For motion planning, the `MoveIt` framework [239] was used. Our implementation is available at the project's git repository[8]

The camera is calibrated using the VISP library [240]. By taking some pictures of a known pattern (a chessboard in our case), the transformation between the robot's end effector frame and the camera frame is calculated. Since the position of the plane where the objects are placed and the 3D models of the objects are known, the inverse perspective projection equations can be used to transform object positions from the image frame to the camera frame, then transform them to the world frame using the transformation matrix obtained from the calibration.

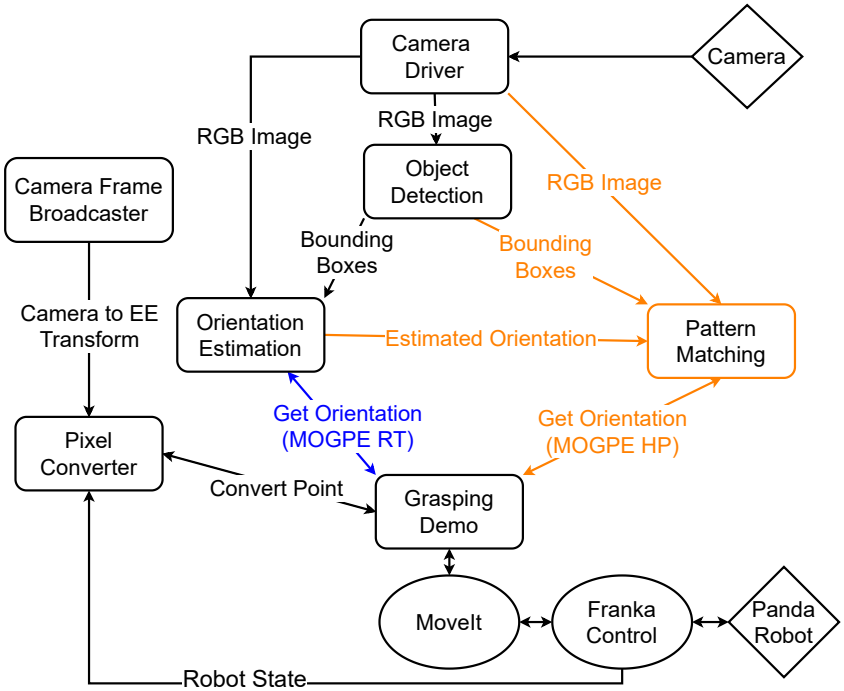


Figure 4.5: The robot control architecture. With blue colour, the version of the MOGPE-RT model, while with orange colour, the version of the MOGPE-HP model.

[8] https://git.sztaki.hu/emi/robot_control_framework

4.6 Results

In this section, the evaluation of our approach is presented. First, the settings of the robotic manipulation problem are described in Section 4.6.1. Then, the results of the object detection and the orientation estimation models are demonstrated in Section 4.6.2 and 4.6.3. Finally, the entire pipeline is validated in a real-world robotic grasping experiment, presented in Section 4.6.4.

4.6.1 Setting of the robotic experiments

For this robotic case study, three industrial parts were selected that are themselves parts of a simple robot arm, shown in Fig. 4.6. Synthetic samples of the parts are depicted in Fig. 4.4.

Initially, the parts are randomly placed in the starting area. The task of the robot is to pick and place the parts one by one from the starting area to the designated target positions using its two-finger gripper. Neither special illumination was applied nor monochromatic background.

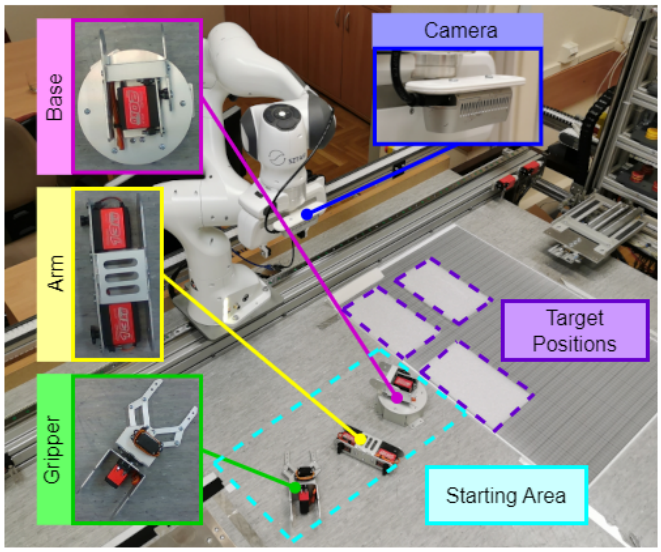


Figure 4.6: Experimental setup.

Table 4.1: The mAP_{50} scores of the object detection model.

	Dataset		
	Train	Valid	Test
Training #1	100%	100%	98.85%
Training #2	100%	100%	98.81%
Training #3	100%	99.81%	98.85%
Training #4	100%	100%	98.81%
Training #5	97.07%	95.26%	98.56%
AVG	**99.41%**	**99.01%**	**98.78%**
STD	**1.3103%**	**2.1001%**	**0.1224%**

4.6.2 Object detection

To train the model[9], 2000 synthetic images were generated using S2R-ObjDet, supplemented with a set of real images. The real images were multiplied to match the number of synthetic images. The batch size was set to 64, and the other hyper-parameters of the training were chosen according to the recommendation of [34].

The quantitative evaluation is shown in Tab. 4.1. The validation dataset was generated from the same distribution as the training dataset. On the other hand, the test dataset contains 59 real images, taken in different environmental and illumination conditions. Achieving an average of 98.78% mAP_{50} on the test dataset can be considered a robust performance. Having a reliable output of the object detection stage is crucial as this output is the input of the orientation estimation. As it is shown in Chapter 3, the S2R-ObjDet method works for more classes as well, and as in the orientation estimation stage, every class processed separately, our method can be easily scaled up to more classes.

For qualitative evaluation, Fig. 4.7 shows two accurate examples of object detection. For more qualitative evaluation, we refer the reader to our qualitative evaluation video[10].

4.6.3 Orientation estimation

To train the orientation estimation CNNs, 4320 synthetic annotated images were generated with our S2R-PosEst method. As a real dataset, 15, 12, and 12 real images were available

[9]Pre-trained on ImageNet.

[10]https://youtu.be/luwA6RDEaoA

Table 4.2: The success rate of the pose estimation model. An estimation is considered successful if it is within 10 degrees of the ground truth.

Dataset	Base	Arm	Gripper	**AVG**	**STD**
Train synthetic	99.76%	98.71%	99.54%	**99.34%**	**0.55%**
Train real	99.85%	99.75%	99.83%	**99.81%**	**0.05%**
Valid	100.0%	99.16%	99.72%	**99.63%**	**0.42%**
Test	99.17%	92.22%	99.72%	**97.04%**	**4.18%**

from the classes of the base, arm, and gripper. These real images were also augmented by rotating them 359 times which resulted in (with the original one) 5400, 4320, and 4320 images per class. 720 (2 times 360) real images were taken away per class for validation and testing. The loss function is MSE with Adam optimiser and the learning rate is 0.001. The batch size is 128, the training time is 100 epochs with early stopping. The loss function converged rapidly both on the training and on the validation set.

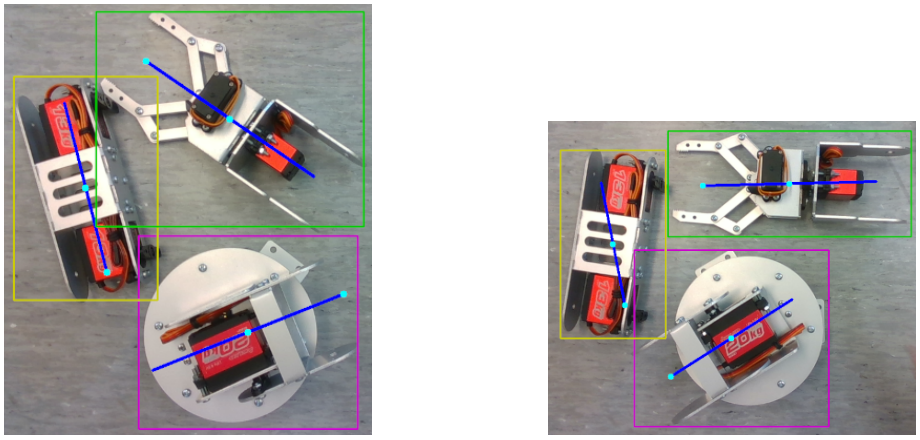
For quantitative evaluation, Tab. 4.2 shows the success rate of the models. An estimation is considered successful if it is within 10 degrees of the ground truth. In the case of the base and gripper objects, the success rate is above 99% in all datasets. In the case of the arm, the model achieves a 92% success rate on the test dataset. The primary reason behind this phenomenon might be the near 180-degree rotational symmetry of the arm object. Restricting the estimation range to 180 degrees could potentially enhance performance.

For qualitative evaluation, Fig. 4.7 shows an accurate and an inaccurate example of orientation estimation. For more qualitative evaluation, we refer the reader to our qualitative evaluation video[11].

4.6.4 Robotic grasping

Finally, the performance of our models was measured in a real-world robotic grasping experiment using a 7 DoF collaborative robot. Ten grasp attempts were made per class and per model (all in all 60 grasp attempts). The results of the experiments are summarised in Tab. 4.3. The real-time MOGPE-RT model worked well in the case of the arm and gripper classes. Nevertheless, it failed to reliably grasp the base class. On the other hand, the high-precision MOGPE-HP model could successfully grasp the objects most of the times,

[11] See footnote 10.



(a) Accurate example (b) Inaccurate example

Figure 4.7: An accurate (a) and an inaccurate (b) prediction. The orientation of the arm is slightly tilted in the latter case. Regarding the object detection, both examples are accurate.

yielding a 96.67 % success rate. Six grasp attempts are shown in our qualitative evaluation video[12].

Table 4.3: Results of the robotic grasping experiment.

Model	Base	Arm	Gripper	**Success rate**
MOGPE-RT	5/10	9/10	10/10	**80%**
MOGPE-HP	10/10	10/10	9/10	**96.67%**

4.7 Conclusion

In this chapter, robotic grasping was addressed, a critical challenge of adaptive robotics. Two vision-based, multi-object grasp pose estimation models were presented – the real-time MOGPE-RT and the high-precision MOGPE-HP – coupled with our sim2real knowledge transfer methods (S2R-ObjDet and S2R-PosEst) based on domain randomization to diminish the reality gap and to overcome the data shortage.

[12]See footnote 10.

Our framework provides an industrial tool for fast data generation and model training and requires minimal domain-specific data. In test time, the model does not only work fast (object detection 20 FPS, orientation estimation 100 FPS) but performs well (98.78% mAP_{50} score, and 97.04% success rate).

Our approach is validated not only on images but in a real-world robotic grasping experiment where the MOGPE-RT model achieved an 80%, while the MOGPE-HP model accomplished a 96.67% success rate.

Based on the findings presented in this chapter, our theses connected to the second research question regarding extending our S2R-ObjDet method to multi-object grasp pose estimation are as follows:

Thesis III: Our novel two-stage multi-object grasp pose estimation methods – the real-time MOGPE-RT and the high-precision MOGPE-HP – enable a modular training approach for multi-object grasp pose estimation by utilizing sequential phases of object detection and class-specific orientation estimation.

We propose two vision-based, multi-object grasp pose estimation models – the real-time MOGPE-RT and the high-precision MOGPE-HP – depicted in Fig. 4.1. Both models are built upon two core components: an object detection model and an orientation estimation model. The output of the object detection model is $\boldsymbol{y} = \{(\boldsymbol{b}_i, c_i^{class}, p_i^{con}) \mid i = 1, 2, \ldots, N\}$, *where* $\boldsymbol{b}_i = [x_i, y_i, w_i, h_i] \in [0, 1]^4$ *represents the axis-aligned bounding box of the* i^{th} *detection,* $c_i^{class} \in \mathbb{N}$ *is the class label of the* i^{th} *detection,* $p_i^{con} \in [0, 1]$ *is the confidence score of the* i^{th} *detection, and* $N \in \mathbb{N}$ *is the number of detected objects. The detections with* $p_i^{con} < \tau_{con}$ *are filtered out, where* $\tau_{con} \in [0, 1]$ *is the confidence threshold. The ROI cropping module extracts specific objects from the image and resizes them to the appropriate dimensions and shape. The class-specific orientation estimation models compute the* $\sin(\theta_i)$ *and* $\cos(\theta_i)$ *for all objects, where* $\theta_i \in [-\pi, \pi]$ *is the orientation angle. Then, with the atan2 function, the* θ_i *angles are computed which is the output of the MOGPE-RT model. In the case of the MOGPE-HP model, an additional local pattern-matching algorithm is incorporated, allowing for the estimation of a more precise* $\theta^* \in [-\pi, \pi]$ *at the expense of the extra computation. This thesis is associated with [4].*

Thesis IV: Our novel S2R-PosEst method facilitates rapid synthetic data generation for single-class orientation estimation models, effectively bridging the reality gap.

We propose S2R-PosEst, a sim2real domain randomization method for pose estimation, based on our S2R-ObjDet method. The 3D model of the given object is placed in the simulator and rotated around the z-axis – perpendicular to the plane where the object is placed – while random textures are added to the plane and to the object as well. All together, there are $n_{rot} = \lfloor \frac{2\pi}{\beta_{res}} \rfloor$ *rotations, where* $n_{rot} \in \mathbb{N}$ *is the number of rotation and* $\beta_{res} \in \mathbb{R}$ *is the resolution in radian. For each rotation, an image is taken and the label is automatically generated with it. The data generation requires 0.25–0.5s per image, making it suitable for industrial applications. This thesis is associated with [4].*

Chapter 5

Highlight experience replay

Contents

5.1 Introduction 103
5.2 Related works 105
5.2.1 Data exploitation 106
5.2.2 Data collection 106
5.3 Method 109
5.3.1 HiER 109
5.3.2 E2H-ISE 112
5.3.3 HiER+ 115
5.4 Results 115
5.4.1 Evaluation protocol 118
5.4.2 Aggregated results across all tasks 119
5.4.3 Panda-Gym 122
5.4.4 Gymnasium-Robotics Fetch 123
5.4.5 Gymnasium-Robotics PointMaze 123
5.4.6 Qualitative evaluation 128
5.4.7 HiER λ, HiER ξ, and E2H-ISE c versions 130
5.4.8 TD3 and DDPG 130
5.5 Conclusion 130

In Chapter 3 and Chapter 4, the main focus was on transferring knowledge from simulation to the real world in cases of supervised learning problems, namely object detection and pose estimation. Nevertheless, the endeavor for adaptive robots is coupled not only with transferability but universality as well. An important building block in this attempt might be reinforcement learning. Similarly to humans, RL algorithms learn from trial-and-error through interactions with the environment. Compared to supervised learning, RL is especially beneficial for robotic tasks that require a high level of dexterity. It is important to note that transferability and universality are not completely separate concepts as by definition, universal solutions are easily transferable to other tasks, robots, or domains. In this chapter, we focus on how to improve the state-of-the-art RL algorithms with curriculum learning.

Even though RL algorithms achieved superhuman performance in many domains, the field of robotics poses significant challenges as the state and action spaces are continuous, and the reward function is predominantly sparse. Furthermore, on many occasions, the agent is devoid of access to any form of demonstration. Inspired by human learning, in this chapter, we propose a method named highlight experience replay (HiER) that creates a secondary highlight replay buffer for the most relevant experiences. For the weights update, the transitions are sampled from both the standard and the highlight experience replay buffer. It can be applied with or without the techniques of *hindsight experience replay* (HER) and *prioritized experience replay* (PER). Our method significantly improves the performance of the state-of-the-art, validated on 8 tasks of three robotic benchmarks. Furthermore, to exploit the full potential of HiER, we propose HiER+ in which HiER is enhanced with an arbitrary data collection curriculum learning method. Our implementation, the qualitative results, and a video presentation are available on the project site: www.danielhorvath.eu/hier.

5.1 Introduction

Reinforcement learning methods, especially combined with neural networks (deep reinforcement learning), were proven to be superior in many fields such as achieving superhuman performance in chess [77], Go [78], or Atari games [79]. Nevertheless, in the field of robotics, there are significant challenges yet to overcome. Most importantly, the state and action spaces are continuous which intensifies the challenge of exploration. Oftentimes, discretization is not feasible due to loss of information or accuracy, preventing the application of tabular RL methods with high stability. Furthermore, the reward functions of robotic tasks are predominantly sparse which escalates the difficulty of exploration.

Introducing prior knowledge in the form of reward shaping could facilitate the exploration by guiding the agent toward the desired solution. However, 1) constructing a sophisticated reward function requires expert knowledge, 2) the reward function is task-specific, and 3) the agent might learn undesired behaviours. Another source of prior knowledge could be in the form of expert demonstrations. However, collecting demonstrations is oftentimes expensive (time and resources) or even not feasible. Furthermore, it constrains transferability as demonstrations are task-specific.

In parallel to constructing more efficient RL algorithms such as state-of-the-art actor-critic models (DDPG [80], [81], TD3 [82], and SAC [83]), another line of research focuses on improving existing RL algorithms by controlling the data collection [89]–[95], [241], [242] or the data exploitation [84]–[88], [243] process. Following [97], in this work, we consider both the data collection and the data exploitation methods as curriculum learning (CL) methods [96]–[98]. The former is oftentimes referred to as *traditional* and the latter as *implicit* CL.

Our aim is to improve the training of off-policy reinforcement learning agents, particularly in scenarios with continuous state and action spaces, sparse rewards, and the absence of demonstrations. These conditions pose significant challenges for state-of-the-art RL algorithms, due to the challenging problem of exploration. Our contribution to the field is as follows.

1. **HiER**: The highlight experience replay creates a secondary experience replay buffer to store the most relevant transitions. At training, the transitions are sampled from both the standard experience replay buffer and the highlight experience replay buffer. It can be added to any off-policy RL agent and applied with or without the techniques of hindsight experience replay (HER) [84] and prioritized experience replay (PER) [86]. If only positive experiences are stored in its buffer, HiER can be viewed as a special, automatic demonstration generator as well. Following [97], HiER is classified as a data exploitation or implicit curriculum learning method.

2. **HiER+**: The enhancement of HiER with an arbitrary data collection (traditional) curriculum learning method. The overview of HiER+ is depicted in Fig.5.1. Furthermore, as an example of the data collection CL method, we propose E2H-ISE, a universal, easy-to-implement *easy2hard* data collection CL method that requires minimal prior knowledge and controls the entropy of the initial state-goal distribution $\mathcal{H}(\mu_0)$ which indirectly controls the task difficulty[1]

[1]ISE stands for *initial state entropy*.

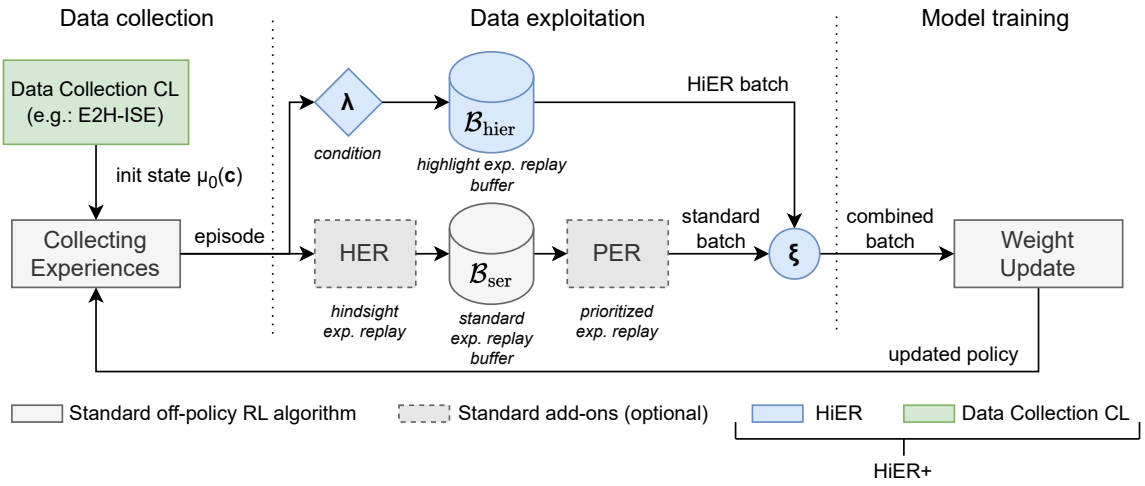


Figure 5.1: Overview of HiER and HiER+. For every episode, the initial state is sampled from μ_0. After every episode, the transitions are stored in $\mathcal{B}_{\text{ser}}$, and in case the λ condition is fulfilled then in $\mathcal{B}_{\text{hier}}$ as well. For training, the transitions are sampled from both $\mathcal{B}_{\text{ser}}$ and $\mathcal{B}_{\text{hier}}$ according to the ratio ξ. For a detailed description, see Algorithm 1 and 2.

To demonstrate the universality of our methods, HiER is validated on 8 tasks of three different robotic benchmarks [244]–[246] based on two different simulators [188], [192], while HiER+ is evaluated on the push, slide, and pick-and-place tasks of the Panda-Gym [244] robotic benchmark. Our methods significantly improve the performance of the state-of-the-art algorithms for each task.

The background of reinforcement learning and curriculum learning is presented in Section 2.3 and 2.4. This chapter is structured as follows: in Section 5.2, a literature review is presented. In Section 5.3 and 5.4, HiER, E2H-ISE, and HiER+ are detailed with the experimental results. Finally, the summary of our findings is provided in Section 5.5.

5.2 Related works

The summary of the related works is presented in Tab. 5.1. Following Section 2.4, the CL algorithms are categorised as data exploitation or data collection methods, presented in Section 5.2.1 and 5.2.2. Data exploitation methods either modify the transitions or control the transition selection. The structure and the performance of HiER are compared

with the state-of-the-art in Section 5.3.1 and 5.4. On the other hand, the data collection methods – presented in Section 5.2.2 – either control the initial state distribution or the goal distribution. The E2H-ISE method controls both the initial state distribution and the goal distribution as the state space is augmented with the goal, as described in Section 2.3.2. Comparison with the state-of-the-art is presented in Section 5.3.2 and 5.4.

5.2.1 Data exploitation

Schaul et al. [86] proposed the technique of prioritized experience replay (PER) which controls the transition selection by assigning priority (importance) scores to the samples of the replay buffer based on their last TD error [247] and thus, instead of uniformly, they are sampled according to their priority. Additionally, as high-priority samples would bias the training, importance sampling is applied.

As a form of prioritization, Oh et al. [87] introduced self-imitation learning (SIL) for on-policy RL. The priority is computed based on the discounted cumulative rewards. Furthermore, the technique of clipped advantage is utilised to incentivise positive experiences. By modifying the Bellmann optimality operator, Ferret et al. [243] introduced self-imitation advantage learning which is a generalised version of SIL for off-policy RL.

Wang et al. [88] presented the method of emphasising recent experience which is a transition selection technique for off-policy RL agents. It prioritises recent data without forgetting the past while ensuring that updates of new data are not overwritten by updates of old data.

Andrychowicz et al. [84] introduced the technique of hindsight experience replay (HER) which performs transition modification to augment the replay buffer by adding virtual episodes. After collecting an episode and adding it to the replay buffer, HER creates virtual episodes by changing the (desired) goal to the achieved goal at the end state (or to another state depending on the strategy) and relabeling the transitions before adding them to the replay buffer.

Bujalance and Moutarde [85] propose reward relabeling to guide exploration in sparse-reward robotic environments by giving bonus rewards for the last L transitions of the episodes.

5.2.2 Data collection

Florensa et al. [89] presented the reverse curriculum generation method to facilitate exploration for model-free RL algorithms in sparse-reward robotic scenarios. At first, the

Table 5.1: Summary of related works.

	What does CL control?	Work	Short description
Data exploitation	Transition modification	HER [84]	Creating virtual episodes by changing the desired goal to the achieved goal.
		R^2 [85]	Reward relabelling of last transitions of successful episodes.
	Transition selection	PER [86]	Transition selection is based on the last TD-error of the given transition.
		SIL [87]	Transition selection is based on the clipped advantage.
		ERE [88]	In transition selection, recent data is prioritized, without forgetting the old transitions.
		HiER (ours)	**Creating a secondary replay buffer for the most relevant experiences.**
Data collection	Initial state distribution	Reverse curriculum gen. [89]	Inititial state is close to the goal. The distance to the goal is gradually increased throughout the training.
		BaRC [90]	Generalization of [89] by approximating backward reaching sets.
		Salimans and Chen [91]	Initial state is sampled from human demonstration. The distance to the goal is gradually increased.
	Goals	Asymmetric self-play [92]	Training an agent against differently capable versions of itself to enhance its adaptability and robustness.
		Goal GAN [93]	A generator is trained to output new goals with appropriate (intermediate) difficulty.
		Skew-Fit [94]	Maximising the entropy of the goal-conditioned visited states by giving higher weights to rare samples.
		Racanière [95]	Setter agent generates goals for the solver agent considering goal validity, feasibility, and coverage.
	Initial state-goal dist.	**E2H-ISE (ours)**	**The initial state-goal distribution entropy is gradually increased throughout the training.**

environment is initialised close to the goal state. For new episodes, the distance between the initial state and the goal state is gradually increased. As prior knowledge, at least one goal state is required. To sample *nearby* feasible states, the environment is initialised in a

certain seed state (in the beginning at a goal state), and then, for a specific time, random Brownian motion is executed.

Ivanovic et al. [90] proposed the backward reachability curriculum method which is a generalization of [89] utilising prior knowledge of the simplified, approximate dynamics of the system. They compute the approximate backward reaching sets using the Hamilton-Jacobi reachability formulation and sample from them using rejection sampling.

Salimans and Chen [91] facilitate exploration by utilising one human demonstration. In their method, the initial states come from the demonstration. More precisely, until a timestep $t_D \in \mathbb{N}$, the agent copies the actions of the demonstration, and after t_D, it takes actions according to its policy. During the training, t_D is moved from the end of the demonstration to the beginning of the demonstration. Their method outperformed state-of-the-art methods in the Atari game Montezuma's Revenge. Nevertheless, arriving at the same state after a specific sequence of actions (as in the demonstration) is rather unlikely, especially when the transition function is profoundly stochastic, such as in robotics.

Sukhbaatar et al. [92] present automatic curriculum generation with asymmetric self-play of two versions of the same agent. One proposes tasks for the other to complete. With an appropriate reward structure, they automatically create a curriculum for exploration.

Florensa et al. [93] create a curriculum for multi-goal tasks by sampling goals of intermediate difficulty. First, the goals are labelled based on their difficulty, and then a generator is trained to output new goals with appropriate difficulty to efficiently train the agent.

Pong et al. [94] proposed Skew-Fit, an automatic curriculum that attempts to create a better coverage of the state space by maximising the entropy of the goal-conditioned visited states $\mathcal{H}(\mathcal{S}|\mathcal{G})$ by giving higher weights to rare samples. Skew-Fit converges to uniform distribution under specific conditions.

Racanière et al. [95] proposed an automatic curriculum generation method for goal-oriented RL agents by training a setter agent to generate goals for the solver agent considering goal validity, goal feasibility, and goal coverage.

The data collection CL methods are relatively disparate, however, some share specific characteristics. The methods that control the initial state distribution [89]–[91] attempt to reduce the task difficulty by proposing less challenging starting positions. Other algorithms [92], [95], utilise a secondary agent to train the protagonist. Instead of focusing on task difficulty, the E2H-ISE algorithm controls the entropy of the init-goal state distribution $\mathcal{H}(\mu_0)$. Among the considered methods, only Skew-Fit [94] controls the entropy but in that case, it is the entropy of the goal-conditioned visited states $\mathcal{H}(\mathcal{S}|\mathcal{G})$, not $\mathcal{H}(\mu_0)$.

5.3 Method

In this Section, our contributions are presented. First, HiER in Section 5.3.1, and then E2H-ISE and HiER+ in Section 5.3.2 and 5.3.3. The implementation is available at our git repository[2].

5.3.1 HiER

Humans remember certain events stronger than others and tend to replay them more frequently than regular experiences thus learning better from them [248]. As an example, an encounter with a lion or scoring a goal at the last minute will be engraved in our memory. Inspired by this phenomenon, HiER attempts to find these events and manage them differently than regular experiences. In this work, only positive experiences are considered with HiER, thus it can be viewed as a special, automatic demonstration generator as well.

HER and PER control what transitions to store in the experience replay buffer and how to sample from them. Contrary to them, HiER creates a secondary experience replay buffer. Henceforth, the former buffer is called standard experience replay buffer $\mathcal{B}_{\text{ser}}$, and the latter is referred to as highlight experience replay buffer $\mathcal{B}_{\text{hier}}$. At the end of every episode, HiER stores the transitions in $\mathcal{B}_{\text{hier}}$ if certain criteria are met. For updates, transitions are sampled both from the $\mathcal{B}_{\text{ser}}$ and $\mathcal{B}_{\text{hier}}$ based on a given sampling ratio. HiER is depicted in Fig. 5.1 marked in blue and detailed in Algorithm 1.

The criteria can be based on any type of performance measure. In our case, the undiscounted sum of rewards $G_0 = \sum_{i=0}^{T} r_i$ was chosen. For simplicity, henceforth, G_0 is referred to as G. The reward function r is formulated as in Eq. (2.10). Although more complex criteria are possible, in this work, we consider only one performance measure and one criterion: if G is greater than a threshold $\lambda \in \mathbb{R}$ then all the transitions of that episode are stored in $\mathcal{B}_{\text{hier}}$ and $\mathcal{B}_{\text{ser}}$, otherwise only in $\mathcal{B}_{\text{ser}}$. Nevertheless, λ can change in time, thus we define a λ_j for every j where $j \in \mathbb{N}$ is the index of the episode. In this work, the following λ modes were considered:

- `fix`: $\lambda_j = Z_\lambda$ for every j where $Z_\lambda \in \mathbb{R}$ is a constant.[3]
- `predefined`: λ is updated according to a predefined profile. Profiles could be arbitrary, such as linear, square-root, or quadratic. In this work only the linear profile

[2] https://github.com/sztaki-hu/hier

[3] We also tried a version with $m \in \mathbb{N}$ highlight buffers and m thresholds $\lambda^1, \lambda^2, \ldots, \lambda^m$. An episode is stored in the highlight buffer with the highest λ^i for which $G > \lambda^i$.

Algorithm 1 HiER

$\lambda,\ \xi,\ \theta,\ \phi$ ▷ *Initialise variables and model parameters*
$n_{\text{batch}},\ n_{\text{hier}} \leftarrow \xi \cdot n_{\text{batch}}$ ▷ *Initialise batch sizes*
$\mathcal{B}_{\text{ser}} \leftarrow [\,],\ \mathcal{B}_{\text{hier}} \leftarrow [\,]$ ▷ *Initialise buffers*
$N \leftarrow 100, \boldsymbol{G}^{\text{eval}} \leftarrow [\,],\ \boldsymbol{\nu}^{\text{eval}} \leftarrow [\,]$ ▷ *Initialise evaluation variables and metrics*
$\mathcal{E} \leftarrow [\,]$ ▷ *Initialise episode buffer*
$s \sim \mu_0$ ▷ *Initialise environment*
while Convergence **do**
  $a \leftarrow \pi_\theta(s)$ ▷ *Collecting data*
  $s', r, d \sim p(s, a)$
  $\mathcal{E} \leftarrow \mathcal{E} + [(s, a, s', r, d)]$
  $s \leftarrow s'$
  if Episode ends **then**
    $\mathcal{B}_{\text{ser}} \leftarrow \mathcal{B}_{\text{ser}} + [\mathcal{E}]$ ▷ *Store transitions of episode $\mathcal{E}$*
    $\mathcal{B}_{\text{ser}} \leftarrow \mathcal{B}_{\text{ser}} + [\mathcal{E}_1^V, \mathcal{E}_2^V, \ldots, \mathcal{E}_m^V]$ ▷ *HER (optional): Store virtual episodes*
    Update λ_j ▷ *HiER: Section 5.3.1*
    if $\lambda_j < \sum_{r_i \in \mathcal{E}} r_i$ **then**
      $\mathcal{B}_{\text{hier}} \leftarrow \mathcal{B}_{\text{hier}} + [\mathcal{E}]$
    end if
    $\mathcal{E} \leftarrow [\,]$
    $s \sim \mu_0$
  end if
  if Weight update **then**
    $\mathcal{D}_{\text{ser}} \leftarrow$ select $(n_{\text{batch}} - n_{\text{hier}})$ sample from $\mathcal{B}_{\text{ser}}$
    $\mathcal{D}_{\text{hier}} \leftarrow$ select n_{hier} sample from $\mathcal{B}_{\text{hier}}$
    $\mathcal{D} \leftarrow \mathcal{D}_{\text{ser}} + \mathcal{D}_{\text{hier}}$
    Update weights $\theta,\ \phi$ based on $\mathcal{D}$
    Update priorities in $\mathcal{B}_{\text{ser}}$ ▷ *PER (optional)*
    Update ξ_k ▷ *HiER: Section 5.3.1*
    $n_{\text{hier}} \leftarrow \xi_k \cdot n$
  end if
  if Evaluation **then**
    $\boldsymbol{E} = (\mathcal{E}_1^E, \mathcal{E}_2^E, \ldots, \mathcal{E}_N^E),\ \mathcal{E}_i^E = (s_{i,0}, a_{i,0}, r_{i,1}, \ldots, s_{i,T}) \sim \pi_\theta, p, s_{i,0} \sim \mu_0$
    $\boldsymbol{G}^{\text{eval}} \leftarrow \boldsymbol{G}^{\text{eval}} + \left[\frac{1}{N}\sum_{i=1}^{N}\left[\frac{1}{|r_i|}\sum_{j=1}^{|r_i|} r_{i,j}\right]\right]$ ▷ *Append with mean accumulated reward*
    $\boldsymbol{\nu}^{\text{eval}} \leftarrow \boldsymbol{\nu}^{\text{eval}} + \left[\frac{1}{N}\sum_{i=1}^{N} \mathbf{1}_{\left\{s_{i,T} \in \mathcal{S}^g\right\}}\right]$ ▷ *Append with mean success rate*
  end if
end while

with saturation was considered:

$$\lambda_j = (\lambda_{\text{sat}} - \lambda_0) \min\left(1,\ \frac{t}{T_{\text{total}} \cdot z_{\text{sat}}}\right) + \lambda_0, \tag{5.1}$$

where $t \in \mathbb{N}$ and $T_{total} \in \mathbb{N}$ are the actual, and the total timesteps of the training and $z_{sat} \in [0, 1]$ is a scaler, indicating the start of the saturation.[4] Furthermore, $\lambda_0 \in \mathbb{R}$ and $\lambda_{sat} \in \mathbb{R}$ is the initial value of λ, while $\lambda_{sat} \in \mathbb{R}$ is the λ value after saturation.

- `ama` (adaptive moving average): λ is updated according to:

$$\lambda_j = \begin{cases} \min\left(\lambda_{\max},\ M + \frac{1}{w}\sum_{i=1}^{w} G^{j-i}\right), & \text{if } j > w \\ \lambda_0, & \text{otherwise,} \end{cases} \tag{5.2}$$

where $\lambda_0 \in \mathbb{R}$ is the initial value of λ, while $\lambda_{\max} \in \mathbb{R}$ is the maximum value allowed for λ. Furthermore, G^j is the undiscounted return of the j^{th} episode, $w \in \mathbb{Z}^+$ is the window size and $M \in \mathbb{R}$ is a constant shift.[5]

Another relevant aspect of HiER is the sampling ratio between $\mathcal{B}_{\text{ser}}$ and $\mathcal{B}_{\text{hier}}$ for weight update, defined by $\xi \in [0, 1]$. It can change in time, updated after every weight update, thus we define a ξ_k for every k where $k \in \mathbb{N}$ is the index of the weight update. The following versions were considered:

- `fix`: $\xi_k = Z_\xi$ for every k where $Z_\xi \in \mathbb{R}$ is a constant.
- `prioritized`: ξ is updated according to:[6]

$$\xi_k = \frac{\mathcal{L}_{\text{hier},k}^{\alpha_p}}{\mathcal{L}_{\text{hier},k}^{\alpha_p} + \mathcal{L}_{\text{ser},k}^{\alpha_p}}, \tag{5.3}$$

where $\mathcal{L}_{\text{hier},k} \in \mathbb{R}$ and $\mathcal{L}_{\text{ser},k} \in \mathbb{R}$ are the TD errors of the training batches sampled from $\mathcal{B}_{\text{hier}}$ and $\mathcal{B}_{\text{ser}}$ at k. The parameter $\alpha_p \in [0, 1]$ determines how much prioritization is used.[7]

Sampling from $\mathcal{B}_{\text{hier}}$ and not only from $\mathcal{B}_{\text{ser}}$ introduce a bias towards the experiments collected in $\mathcal{B}_{\text{hier}}$. This bias is similar in nature to the case when demonstrations are utilised.

[4] In the equation, λ_j does not directly depend on j. However as t increases, so does j and λ_j with it.
[5] In an alternative version M is not a constant but relative to $\frac{1}{w}\sum_{i=1}^{w} G^{j-i}$.
[6] Similarly as in the case of PER.
[7] If $\alpha_p = 0$, then $\xi = 0.5$ regardless $\mathcal{L}_{\text{hier},k}$ and $\mathcal{L}_{\text{ser},k}$.

In that scenario, the expert demonstrations are sampled and combined with online experience, biasing the exploration towards the desired behaviour. In our case, as the agent is devoid of any form of demonstration, $\mathcal{B}_{\text{hier}}$ serves similarly as a demonstration buffer. This bias is essential for achieving enhanced performance (presented in Section 5.4). However, it might be useful to constrain the bias to avoid overfitting on experiences from $\mathcal{B}_{\text{hier}}$. On one hand, the `predefined` and the `ama` λ methods alleviate the bias by setting the entry of $\mathcal{B}_{\text{hier}}$ lower at the beginning and gradually increasing it resulting in a higher cardinality for $\mathcal{B}_{\text{hier}}$ and higher similarity between $\mathcal{B}_{\text{hier}}$ and $\mathcal{B}_{\text{ser}}$. Furthermore, the presented `prioritized` ξ method prevents overfitting on the data of $\mathcal{B}_{\text{hier}}$ as low $\mathcal{L}_{\text{hier}}$ loss reduces ξ – see Eq. (5.3). On the other hand, the bias could be further reduced by gradually decreasing ξ over time, or the gradient of the data from $\mathcal{B}_{\text{hier}}$ could be scaled, similarly to importance sampling in the case of PER [86].

Another relevant aspect worth detailing is the difference between the `prioritized` ξ method and PER. While PER changes the probability distribution of selecting specific transitions from $\mathcal{B}_{\text{ser}}$ based on their individual TD error, the `prioritized` ξ method controls sampling between $\mathcal{B}_{\text{ser}}$ and $\mathcal{B}_{\text{hier}}$ based on the mean TD error of the data selected from $\mathcal{B}_{\text{ser}}$ and $\mathcal{B}_{\text{hier}}$. Thus, the sampling distribution of PER has $|\mathcal{B}_{\text{ser}}|$ outputs while the sampling distribution of the `prioritized` ξ method has two outputs, one for $\mathcal{B}_{\text{ser}}$ and one for $\mathcal{B}_{\text{hier}}$. Another relevant difference is that in the `prioritized` ξ method, contrary to PER, the gradients are not scaled, similar to a standard demonstration buffer.

It is important to note that the formulation of HiER is fundamentally different from [84]–[88], [243], not only but most importantly because of the idea of the secondary experience replay.

5.3.2 E2H-ISE

A key attribute of HiER is that it learns from relevant positive experiences, described in Section 5.3.1. However, if these experiences are scarce in the first place, $\mathcal{B}_{\text{hier}}$ would be considerably limited or even empty. Thus, HiER could benefit from an *easy2hard* data collection CL method by having access to more positive experiences.

E2H-ISE is a data collection CL method based on controlling the entropy of the initial state-goal distribution $\mathcal{H}(\mu_0)$ and with it, indirectly, the task difficulty. In general, μ_0 is constrained to one point (zero entropy) and moved towards the uniform distribution on the possible initial space (max entropy). Even though certain E2H-ISE versions allow decreasing the entropy, in general, they move μ_0 towards max entropy.

To formalise E2H-ISE, the parameter $c \in [0, 1]$ is introduced as the scaling factor of the uniform μ_0, assuming that the state space, including the goal space, is continuous and

bounded. The visualization of the scaling factor c is depicted in Fig. 5.2. If $c = 1$ there is no scaling, while $c = 0$ means that μ_0 is deterministic and returns only the centre point of the space. To increase or decrease $\mathcal{H}(\mu_0)$, c changes in time, thus we define c_j for every j where $j \in \mathbb{N}$ is the index of the episode. At the start of the training, c is initialised and it is updated at the beginning of every training episode before s_0 is sampled from μ_0.[8] The following versions are proposed for updating c:[9]

- `predefined`: c changes according to a predefined profile similar as in the case of λ `predefined` (see Section 5.3.1). In this work, only the linear profile with saturation was considered.

- `self-paced`: c is updated according to:

$$c_j = \begin{cases} \min(1,\ c_{j-1} + \delta_{\text{step}}), & \text{if } \overline{\boldsymbol{\nu}}^{\text{train,W}} > \Psi_{\text{high}} \\ \max(0,\ c_{j-1} - \delta_{\text{step}}), & \text{if } \overline{\boldsymbol{\nu}}^{\text{train,W}} < \Psi_{\text{low}} \\ c_{j-1}, & \text{otherwise,} \end{cases} \tag{5.4}$$

  where $\bar{\boldsymbol{\nu}}^{\text{train},W} \in [0,1]$ is the mean of the last $W \in \mathbb{N}$ training success rates in $\boldsymbol{\nu}^{\text{train}} = [\nu_1^{\text{train}}, \nu_2^{\text{train}}, \ldots, \nu_{n_{\text{train}}}^{\text{train}}] \in [0,1]^{n_{\text{train}}}$. Mathematically, $\bar{\boldsymbol{\nu}}^{\text{train},W} = \frac{1}{W} \sum_{i=n_{\text{train}}-W+1}^{n_{\text{train}}} \nu_i^{\text{train}}$. Furthermore, $\delta_{\text{step}} \in [0,1]$ is the step size, and $\Psi_{\text{high}},\ \Psi_{\text{low}} \in [0,1]$ are threshold values.[10] After any update on c, $\boldsymbol{\nu}^{\text{train}}$ is emptied[11], and the update on c is restarted after W episodes.

- `control`: c is updated according to:

$$c_j = \begin{cases} \min(1,\ c_{t-j} + \delta_{\text{step}}), & \text{if } \overline{\boldsymbol{\nu}}^{\text{train,W}} \geq \psi \\ \max(0,\ c_{t-j} - \delta_{\text{step}}), & \text{if } \overline{\boldsymbol{\nu}}^{\text{train,W}} < \psi, \end{cases} \tag{5.5}$$

  where $\psi \in [0,1]$ is the target. The algorithm attempts to move and keep $\overline{\boldsymbol{\nu}}^{\text{train}}$ at ψ. Updates are executed only if $j > W$.

- `control adaptive`: This method is similar to `control` but the target success rate ψ is not fixed but computed from the mean evaluation success rate:

$$\psi_j = \min\left(\psi_{\max},\ \Delta + \overline{\boldsymbol{\nu}}^{\text{eval},V}\right), \tag{5.6}$$

[8] For evaluation, the environment is always initialised according to the unchanged μ_0.

[9] We have experimented with a 2-stage version where μ_0 and μ_G (initial goal distribution) were separated.

[10] If $\Psi_{\text{low}} = 0$, then c can only increase.

[11] The circular buffer storing the success rates.

where $\bar{\boldsymbol{\nu}}^{\text{eval},V} \in [0,1]$ is the mean of the last $V \in \mathbb{N}$ evaluation success rates in $\boldsymbol{\nu}^{\text{eval}} = [\nu_1^{\text{eval}}, \nu_2^{\text{eval}}, \ldots, \nu_{n_{\text{eval}}}^{\text{eval}}] \in [0,1]^{n_{\text{eval}}}$. Mathematically, $\bar{\boldsymbol{\nu}}^{\text{eval},V} = \frac{1}{V}\sum_{i=n_{\text{eval}}-V+1}^{n_{\text{eval}}} \nu_i^{\text{eval}}$. Furthermore, $\Delta \in [0,1]$ is a constant shift (as we want to target a better success rate than the current) and $\psi_{\max} \in \mathbb{R}$ is the maximum value allowed for ψ.[12] Updates are executed only if $j > V$.

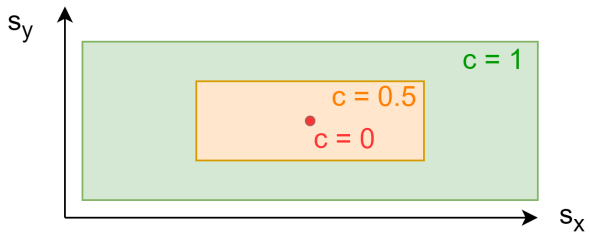


Figure 5.2: Visualization of the effect of parameter c on μ_0 in a 2D case where state $s = [s_x, s_y]$. The initial state $s_0 = [s_{0,x}, s_{0,y}]$ is sampled from the probability distribution $\mu_0(c)$.

Sampling from $\mu_0(c \neq 1)$ introduces bias to the states within the probability distribution of $\mu_0(c)$. This bias is reduced as c increases. Furthermore, as the buffers are circular, once they reach their capacity, the old experiences are replaced with new ones. On the other hand, we conducted experiments on dynamically subtracting the centre of $\mu_0(c)$ to counterbalance the sampling bias, e.g.: $\mu_0(c) = \mu_0(c_1) - \mu_0(c_2)$ where $c_1 > c_2$. However, they did not result in any improvement. Our experimental results, presented in Section 5.4, show that accepting the bias and starting with c close to zero is beneficial as HiER+ further improves the performance of HiER.

It is important to note that our E2H-ISE formulation is inherently different from [89]–[91] as our solution does not concentrate on goal difficulty but the entropy of μ_0. In our case, the *easy2hard* attribute derives from the magnitude of the entropy and not from the goal difficulty. It is also disparate from [94] as their solution maximises the entropy of goal-conditioned visited states $\mathcal{H}(\mathcal{S}|\mathcal{G})$ and not $\mathcal{H}(\mu_0)$. Nevertheless, the E2H-ISE method is only an example of data collection CL methods that can be utilised in HiER+. It is proposed in this work, as it is significantly easier to implement than the presented, more sophisticated, state-of-the-art methods, while it is universal and requires minimal prior knowledge. Thus, the full potential of HiER+ can be presented conveniently with the

[12] It is important to note that contrary to the training, in the evaluation, we sample from the unrestricted μ_0 ($c = 1$), thus the eval success rate represents the real success rate of the agent. Consequently, c can be set to keep the training to a success rate that is just (by Δ) above the eval success rate.

E2H-ISE method. Comparing different data collection CL methods in HiER+ is out of the scope of this work.

5.3.3 HiER+

In this section, HiER+ is presented which is an enhancement of HiER with an arbitrary data collection CL method. Even though in this work, we present HiER+ with E2H-ISE, it is important to note that the fundamental architecture of HiER+ would remain consistent when paired with alternative data collection CL approaches. It can be added to any off-policy RL algorithm with or without HER and PER, as depicted in Fig. 5.1 and presented in Algorithm 2. Having initialised the variables and the environment (Lines 1–6), the training loop starts. After collecting an episode, its transitions are stored in $\mathcal{B}_{\text{ser}}$, and if HER is active then virtual experiences are added as well (Lines 13–14).[13] Then the λ parameter of HiER is updated and if the given condition is met, the episode is stored in $\mathcal{B}_{\text{hier}}$ as well (Lines 15–18). In the next steps, the c parameter of E2H-ISE is updated and the environment is reinitialised (Line 19–21), thus the agent can start collecting the next episode. At a given frequency, the weights of the models are updated (Line 23–31). The batches of $\mathcal{D}_{\text{ser}}$ and $\mathcal{D}_{\text{hier}}$ are sampled and combined (Lines 24–26). After the weight update (Line 27), if PER is active, the priorities in $\mathcal{B}_{\text{ser}}$ are updated (Line 28). Finally, the ξ parameter and with it the batch size of HiER is updated (Lines 29–30). In the periodic evaluation, N episodes are simulated with evaluation settings – meaning that $c = 1$ (Line 33). Then, the relevant metrics are computed and stored in the corresponding lists (Lines 34–35). It is important to note that if the parameter $c = 1$ throughout the training (changing Line 1 and removing Line 19), then Algorithm 2 is equivalent to Algorithm 1 which describes HiER.

It is worth mentioning that HiER+ utilizes a data collection CL method which might require setting the initial states of the episodes. This constraint can be challenging to meet in certain closed simulators and may pose difficulties in real-world robotic training environments. In contrast, HiER does not have this constraint.

5.4 Results

Our contributions were validated on 8 tasks of three robotic benchmarks. The tasks are the push, slide, and pick-and-place tasks of the Panda-Gym [244] and the Gymnasium-

[13] $\mathcal{B}_{\text{ser}}$ and $\mathcal{B}_{\text{hier}}$ are circular buffers, thus once they are full, the new transitions are replacing the old ones.

Algorithm 2 HiER+

$c_0 \leftarrow 0,\ \lambda,\ \xi,\ \theta,\ \phi$ ▷ *Initialise variables and model parameters*
$n_{\text{batch}},\ n_{\text{hier}} \leftarrow \xi \cdot n_{\text{batch}}$ ▷ *Initialise batch sizes*
$\mathcal{B}_{\text{ser}} \leftarrow [\,],\ \mathcal{B}_{\text{hier}} \leftarrow [\,]$ ▷ *Initialise buffers*
$N \leftarrow 100, \boldsymbol{G}^{\text{eval}} \leftarrow [\,],\ \boldsymbol{\nu}^{\text{eval}} \leftarrow [\,]$ ▷ *Initialise evaluation variables and metrics*
$\mathcal{E} \leftarrow [\,]$ ▷ *Initialise episode buffer*
$s \sim \mu_0(c_0)$ ▷ *Initialise environment*
while Convergence **do**
    $a \leftarrow \pi_\theta(s)$ ▷ *Collecting data*
    $s', r, d \sim p(s, a)$
    $\mathcal{E} \leftarrow \mathcal{E} + [(s, a, s', r, d)]$
    $s \leftarrow s'$
    if Episode ends **then**
        $\mathcal{B}_{\text{ser}} \leftarrow \mathcal{B}_{\text{ser}} + [\mathcal{E}]$ ▷ *Store transitions of episode $\mathcal{E}$*
        $\mathcal{B}_{\text{ser}} \leftarrow \mathcal{B}_{\text{ser}} + [\mathcal{E}_1^V, \mathcal{E}_2^V, \ldots, \mathcal{E}_m^V]$ ▷ *HER (optional): Store virtual episodes*
        Update λ_j ▷ *HiER: Section 5.3.1*
        if $\lambda_j < \sum_{r_i \in \mathcal{E}} r_i$ **then**
            $\mathcal{B}_{\text{hier}} \leftarrow \mathcal{B}_{\text{hier}} + [\mathcal{E}]$
        end if
        Update c_j ▷ *E2H-ISE: Section 5.3.2*
        $\mathcal{E} \leftarrow [\,]$
        $s \sim \mu_0(c_j)$
    end if
    if Weight update **then**
        $\mathcal{D}_{\text{ser}} \leftarrow$ select $(n_{\text{batch}} - n_{\text{hier}})$ sample from $\mathcal{B}_{\text{ser}}$
        $\mathcal{D}_{\text{hier}} \leftarrow$ select n_{hier} sample from $\mathcal{B}_{\text{hier}}$
        $\mathcal{D} \leftarrow \mathcal{D}_{\text{ser}} + \mathcal{D}_{\text{hier}}$
        Update weights $\theta,\ \phi$ based on $\mathcal{D}$
        Update priorities in $\mathcal{B}_{\text{ser}}$ ▷ *PER (optional)*
        Update ξ_k ▷ *HiER: Section 5.3.1*
        $n_{\text{hier}} \leftarrow \xi_k \cdot n$
    end if
    if Evaluation **then**
        $\boldsymbol{E} = (\mathcal{E}_1^E, \mathcal{E}_2^E, \ldots, \mathcal{E}_N^E),\ \mathcal{E}_i^E = (s_{i,0}, a_{i,0}, r_{i,1}, \ldots, s_{i,T}) \sim \pi_\theta, p, s_{i,0} \sim \mu_0(c = 1)$
        $\boldsymbol{G}^{\text{eval}} \leftarrow \boldsymbol{G}^{\text{eval}} + \left[\frac{1}{N}\sum_{i=1}^{N}\left[\frac{1}{|r_i|}\sum_{j=1}^{|r_i|} r_{i,j}\right]\right]$ ▷ *Append with mean accumulated reward*
        $\boldsymbol{\nu}^{\text{eval}} \leftarrow \boldsymbol{\nu}^{\text{eval}} + \left[\frac{1}{N}\sum_{i=1}^{N} \mathbf{1}_{\left\{s_{i,T} \in \mathcal{S}^g\right\}}\right]$ ▷ *Append with mean success rate*
    end if
end while

Robotics Fetch benchmarks [245], and two mazes of the Gymnasium-Robotics PointMaze environment [246] – depicted on Fig. 5.12. The Panda-Gym Environment is based on the PyBullet [192] physics engine while the Gymnasium-Robotics Fetch and PointMaze environments are based on MuJoCo [188]. The state and action spaces vary depending on the task. In general, the state space contains the kinematic information of the scene and the goal description while the action space represents the displacement or the force applied on the robot. It is important to note that for all tasks, the state and action spaces are continuous, and the reward function is sparse without any reward shaping. Furthermore, the agent is devoid of access to any form of demonstration. These constraints, significantly exacerbate the difficulty of exploration. It should be emphasised that solving these tasks with rule-based algorithms requires substantial effort and precise engineering due to the continuous nature of the spaces and the inherent stochasticity of the environment. For further details on the benchmarks, the reader is directed to the relevant sections – Section 5.4.3, Section 5.4.4, and Section 5.4.5 –their corresponding papers [244]–[246] and their documentations [249]–[251].

The naming convention of the algorithms is the following: Algorithm [Components]. The algorithm can be either Baseline, HiER, or HiER+, and the options for the components are HER and PER [14]. Thus, Baseline [HER & PER] means that the base (SAC, TD3, or DDPG) RL algorithm was applied with HER and PER. On the other hand, HiER [HER] means that the base RL algorithm was applied with our HiER method and HER but without PER. HiER+ is HiER with E2H-ISE.

First and foremost, we present our evaluation protocol in Section 5.4.1 which is essential for result reproducibility. Then, the aggregate performance (across all tasks) of HiER is shown compared to their corresponding baselines in Section 5.4.2. Subsequently, HiER and HiER+ (with E2H-ISE) are thoroughly evaluated on the push, slide, and pick-and-place tasks of the Panda-Gym robotic benchmark in Section 5.4.3. Furthermore, HiER is evaluated on the push, slide, and pick-and-place tasks and two mazes – depicted on Fig. 5.12 – of the Gymnasium-Robotics Fetch and PointMaze benchmarks in Section 5.4.4 and Section 5.4.5. Then, the qualitative results of all tasks are evaluated in Section 5.4.6. Additionally, the comparisons of the different ξ, λ, and c methods are presented in Section 5.4.7. Finally, our method is validated with DDPG and TD3 in Section 5.4.8.

For our experiments, the SAC RL algorithm was chosen, except in Section 5.4.8. The standard hyperparameters are set as default in [252] except the discount factor $\gamma = 0.95$ as in [244], and the SAC entropy maximization term $\alpha = 0.1$. The buffer size of $\mathcal{B}_{\text{hier}}$ was set to 10^6.

[14] With the exception of Fig. 5.17.

In all the experiments with the exception of Section 5.4.7, HiER was applied with the `predefined` λ method and with the `prioritized` version of ξ when PER was active and with the `fix` version with $\xi = 0.5$ otherwise. Furthermore, in HiER+, the E2H-ISE method was employed with the `self-paced` option. The aforementioned settings were selected according to our comparison presented in Section 5.4.7.

5.4.1 Evaluation protocol

For results reproducibility, it is important to disclose the evaluation protocol. Each algorithm (configuration) and task pair is trained in 10 independent runs with different random seeds. For every run, at a specified frequency, the evaluation performance of the model is measured, presented at Lines 31–35 of of Algorithm 1 which is equivalent to the Lines 32–36 of Algorithm 2. The two most relevant performance metrics are the evaluation success rate (ν^{eval}) and the evaluation accumulated reward (G^{eval}), henceforth success rate and reward. In this work, the performance is measured 50 times during a single training, and each time, the evaluation score is computed by taking the mean of 100 episodes. At the end of the training, all evaluation metrics – most importantly $\boldsymbol{\nu}^{\text{eval}}$ and $\boldsymbol{G}^{\text{eval}}$ – are saved and stored. For the evaluation presented in this paper, in the case of success rates, the best scores of each run were the base datapoints – $[\max \boldsymbol{\nu}^{\text{eval}}_{\text{run1}}, \max \boldsymbol{\nu}^{\text{eval}}_{\text{run2}}, \ldots, \max \boldsymbol{\nu}^{\text{eval}}_{\text{run10}}]$. From these data points (10 per algorithm-task pair), the mean, median, IQM, and OG scores were computed. This evaluation protocol follows [79], [253], [254] and the idea is similar to the method of early stopping.

In the following sections, the primary basis of evaluation is the success rate which was chosen for the following reasons:

- Our main objective is to solve the tasks with the highest success rate. As we focus on sparse reward scenarios with Eq. (2.10), the only additional information in the reward score is how fast the agent solved the task which is less relevant in our case.
- The success rate is an already normalised scale between zero and one. Reward scores of Eq. (2.10) with different time horizons are significantly disparate.
- The reward value depends on the reward function itself. The same task can be executed with a different reward function, whose results are not comparable.
- The success rate could be seen as a specific reward function giving zero reward to every non-goal state, and one for every goal state.

Nevertheless, we report our reward scores, for the aggregated results, presented in Tab. 5.2, and for the results of HiER and HiER+ on the Panda-Gym environment, displayed in Tab. E.2. In the cases of reward scores, instead of the best, the last values of each run were utilised. Our aim is to show that our methods outperform the state-of-the-art not only in the chosen evaluation protocol but in other protocols as well.

In general, we present our results with the mean, median, interquartile mean (IQM), and optimality gap (OG) metrics. For the former three, higher values are better, while for OG, the lower score is better. In the case of the success rate, the desired target is 1.0 which is the maximum achievable score[15]. For displaying the amount of uncertainty, in the graphs, 95% confidence intervals (CIs) were applied.

For plotting the figures of aggregated results, the performance profiles, and the probability improvements, the *rliable* [158] library was utilised. Having 10 runs was sufficient, thus we present our results without task bootstrapping (as default in *rliable*).

5.4.2 Aggregated results across all tasks

Prior to showing the experimental results on each of the three robotic benchmarks, this section provides a summary of the aggregated results across all tasks, focusing on HiER and HiER [HER].

Our experimental results are presented in Tab. 5.2 and Fig. 5.3. The results indicate that both HiER versions outperform their corresponding baseline, and HiER [HER] yields the best performance in all metrics. In terms of point estimates, while Baseline [HER] yields 0.56 and -43.7 IQM success rate and IQM reward, HiER [HER] achieves 0.83 and -32.48 scores which are increments of 0.27 and 11.22, respectively. Moreover, regarding the uncertainty, both HiER and HiER [HER] are superior to their corresponding baselines as the confidence intervals do not overlap.

Additionally, the performance profile graph, presented in Fig. 5.4, displays the run-score and the average-score distributions of the aforementioned algorithms. It shows that both HiER and HiER [HER] have stochastic dominance over their baselines.

Finally, Fig. 5.5 shows that both HiER and HiER [HER] outperform their baselines with 0.85 and 0.88 probability[16]. Additionally, HiER [HER] surpasses HiER with a probability of 0.76.

[15] As the desired target is 1.0 which is in itself the highest possible number, these results are redundant as the mean is also presented. Nevertheless, to facilitate comparison, we preferred to keep them in the graphs.

[16] It is important to note that these probabilities could be significantly higher if the easy tasks were removed.

Table 5.2: HiER compared to the state-of-the-art across all tasks. For the reward, there is no universal desirable target, thus there is no OG value. The column-wise best results are marked in bold. Both HiER version outperform their corresponding baseline. HiER [HER] yields the best performance in all metrics.

	HER	HiER	Success rate				Reward		
			Mean ↑	Median ↑	IQM ↑	OG ↓	Mean ↑	Median ↑	IQM ↑
Baselines	-	-	0.19	0.10	0.09	0.81	-111.56	-48.2	-48.91
	✓	-	0.57	0.50	0.56	0.43	-87.50	-43.19	-43.70
HiER	-	✓	0.44	0.38	0.38	0.56	-98.72	-40.96	-42.28
	✓	✓	**0.75**	**0.80**	**0.83**	**0.25**	**-73.14**	**-31.35**	**-32.48**

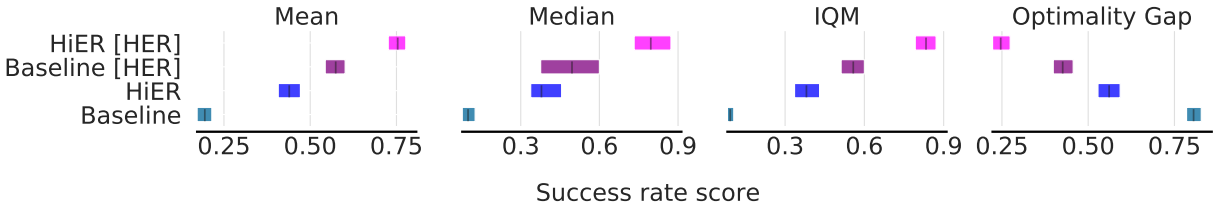


Figure 5.3: HiER compared to the state-of-the-art across all tasks with 95% CIs. Both HiER version outperform their corresponding baseline. HiER [HER] yields the best performance in all metrics. The point estimates are presented in Tab. 5.2.

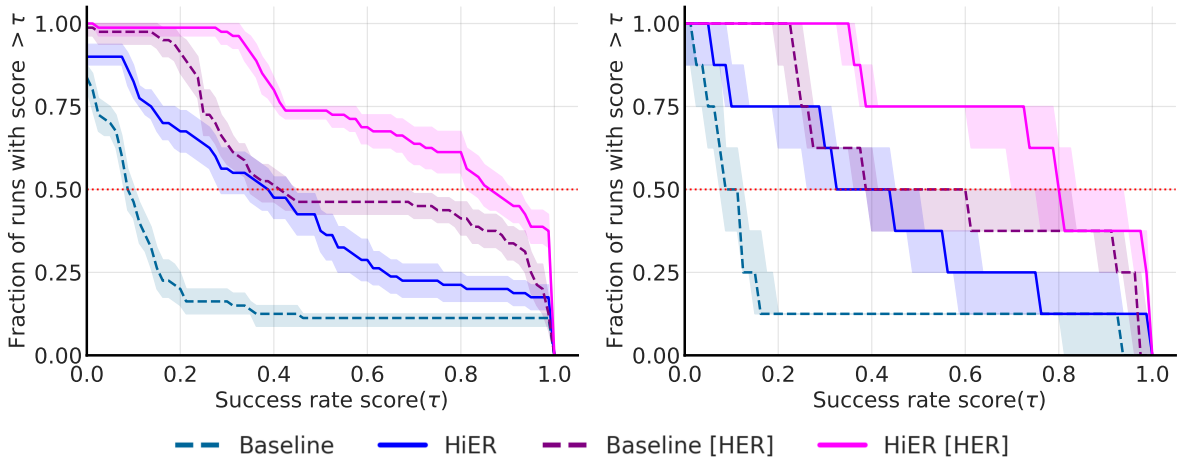


Figure 5.4: Performance profiles across all tasks with 95% CIs. **Left**: run-score distribution, **right**: average-score distribution. The red-dotted line shows the median values while the areas under the performance profiles correspond to the mean values (comparing with Tab. 5.2, the average-score distribution needs to be examined). Both HiER and HiER [HER] have stochastic dominance over their corresponding baselines.

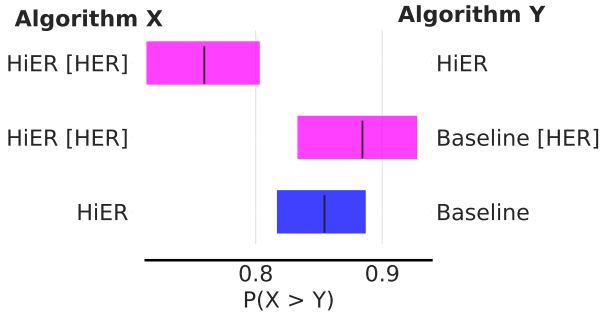


Figure 5.5: Probability of improvement of HiER versions compared to their corresponding baselines and themselves across all tasks with 95% CIs. The average probabilities from top to bottom are the following: 0.76, 0.88, and 0.85.

5.4.3 Panda-Gym

Having presented the aggregated results on HiER, we present our results on the Panda-Gym robotic benchmark with more details and task-specific results. Additionally, we demonstrate how HiER+ with E2H-ISE can further improve the performance of HiER. From the Panda-Gym robotic benchmark, three robotic manipulation tasks were considered:

- `PandaPush-v3`: A block needs to be pushed to a target. Both the block starting position and the target position are within the reach of the robot.
- `PandaSlide-v3`: A puck needs to be slid to a target position outside of the reach of the robot.
- `PandaPickAndPlace-v3`: A block needs to be moved to a target that is oftentimes in the air thus the robot needs to grasp the block.

The starting position of the block (or the puck) and the goal position are sampled from the corresponding distributions. For all three tasks, the state space consists of the kinematic information of the block object and the gripper, and the position of the desired goal. The action space represents the Cartesian displacement (dx, dy, and dz) of the gripper and the closing and opening of the gripper. Regarding the `PandaPush-v3` and `PandaSlide-v3` tasks, the state space $\mathcal{S} \subset \mathbb{R}^{21}$ and the action space $\mathcal{A} \subset \mathbb{R}^3$ (the gripper is closed) while for the `PandaPickAndPlace-v3` task, the state space $\mathcal{S} \subset \mathbb{R}^{22}$ and the action space $\mathcal{A} \subset \mathbb{R}^4$. The reward function is sparse, as described in Eq. (2.10). The tasks are depicted in Fig. 5.6. For further details, we refer the reader to [244] and [249].

The aggregated results are presented in Fig. 5.7, while the performance profiles of the algorithms are demonstrated in Fig. 5.8. Our experimental results show that HiER (blue) and both versions of HiER+ (purple and magenta) significantly outperform the baselines (grey), while E2H-ISE alone could only slightly improve the performance. Moreover, Fig. 5.9 shows at least a 0.99 average probability of improvement for our methods compared to the baselines.

Regarding the specific tasks, the learning curves of the selected configurations are depicted in Fig. 5.6. For all cases, HiER and HiER+ significantly outperform the baselines. Moreover, Tab. 5.3 presents a simplified summary of the performance of the algorithms on the specific tasks. Our results show that HiER [HER] enhances its baseline by an increment of 0.03, 0.44, and 0.12 IQM score on the `PandaPush-v3`, `PandaSlide-v3`, and `PandaPickAndPlace-v3` tasks. Nevertheless, HiER+ [HER] further improves the performance, achieving 1.0, 0.82, and 0.71 IQM scores. The detailed results for all configurations

are presented in Appendix E. Tab. E.1 displays the results based on success rate, while Tab. E.2 shows the results based on rewards.

5.4.4 Gymnasium-Robotics Fetch

In this section, HiER is evaluated on the `FetchPush-v2`, `FetchSlide-v2`, and `FetchPick-AndPlace-v2` tasks of the MuJoCo-based Gymnasium-Robotics Fetch environment.

Even though the tasks are similar to the Panda-Gym robotic benchmark, the robot configuration, the observation space, and the environment dynamic (different simulator) are disparate. For all three tasks, the state space $\mathcal{S} \subset \mathbb{R}^{28}$ consists of the kinematic information of the block object and the gripper, and the position of the desired goal. The action space $\mathcal{A} \subset \mathbb{R}^4$ represents the Cartesian displacement (dx, dy, and dz) of the gripper and the closing and opening of the gripper. Our goal with these experiments is to demonstrate that HiER does not uniquely work for the Panda-Gym robotic benchmark. The tasks are depicted in Fig. 5.10. For more details, we refer the reader to [245] and [250].

In this section, HiER and HiER [HER] are compared with their corresponding baselines. Our experiment results are presented in Tab. 5.4 and depicted in Fig. 5.10 and Fig. 5.11. In all cases, the HiER versions outperform their corresponding baselines. Regarding the `FetchPush-v2` task, HiER [HER] improves the IQM score of the Baseline [HER] method by 0.06 (increasing from 0.92 to 0.98). In the case of the `FetchSlide-v2` task, HiER achieves the best result with a 0.56 IQM score, yielding a 0.54 increase compared to its baseline with 0.02. Interestingly, adding HER worsens the performance. Nevertheless, HiER [HER] still outperforms Baseline [HER]. Finally, for the `FetchPickAndPlace-v2` task, HiER [HER] achieves a 0.73 IQM score. Compared to the Baseline [HER] method with 0.24, it yields a 0.49 improvement. Interesting to note that for the latter two tasks, both HiER versions outperform both baselines.

5.4.5 Gymnasium-Robotics PointMaze

In this section, HiER is evaluated on the `PointMazeWall-v3` and `PointMaze-S-v3` tasks of the MuJoCo-based Gymnasium-Robotics PointMaze environment to show the universality of our approach in a fundamentally different problem.

In these tasks, a ball – placed in a 2D maze – needs to move from the start position to the goal position in a continuous state and action space. The start and the target positions are generated randomly with some constraints. For both tasks, the state space $\mathcal{S} \subset \mathbb{R}^6$ consists of the x and y coordinates and velocities of the force-actuated green ball and the

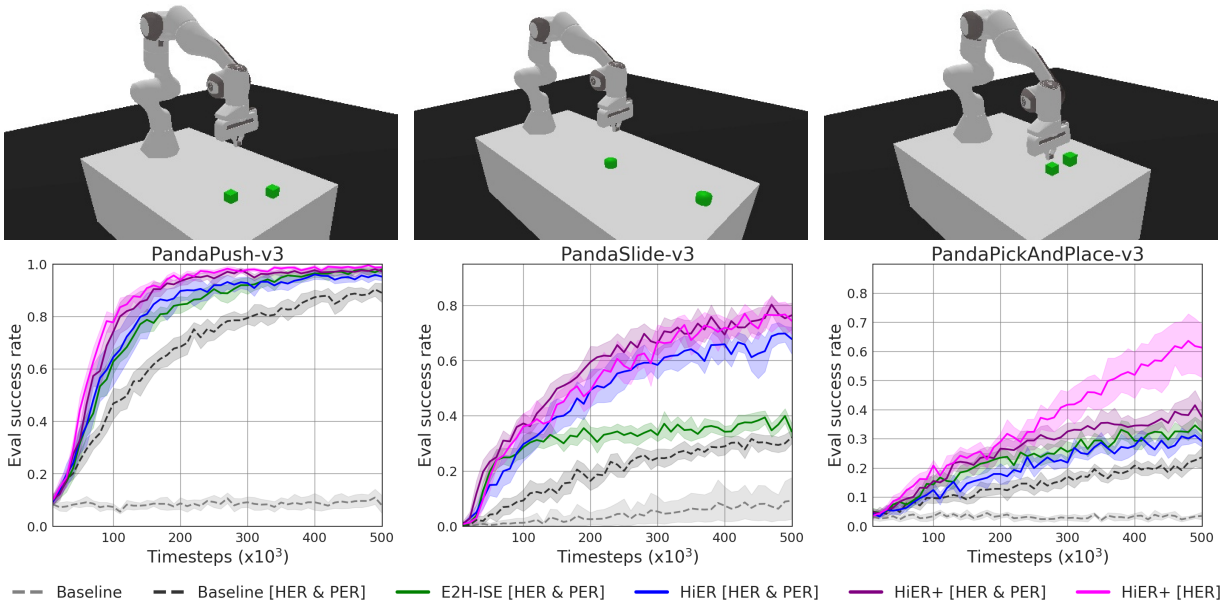


Figure 5.6: Learning curves of HiER and HiER+ with E2H-ISE compared to the state-of-the-art based on success rates on the push, slide, and pick-and-place tasks of the Panda-Gym robotic benchmark with 95% CIs.

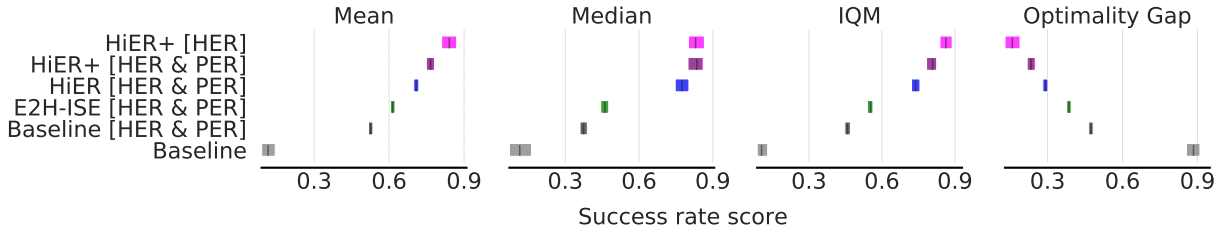


Figure 5.7: Aggregate metrics on the push, slide, and pick-and-place tasks of the Panda-Gym robotic benchmark with 95% CIs. HiER (blue) and both versions of HiER+ (purple and magenta) significantly outperform the baselines (gray). E2H-ISE alone could slightly improve the performance of the baseline.

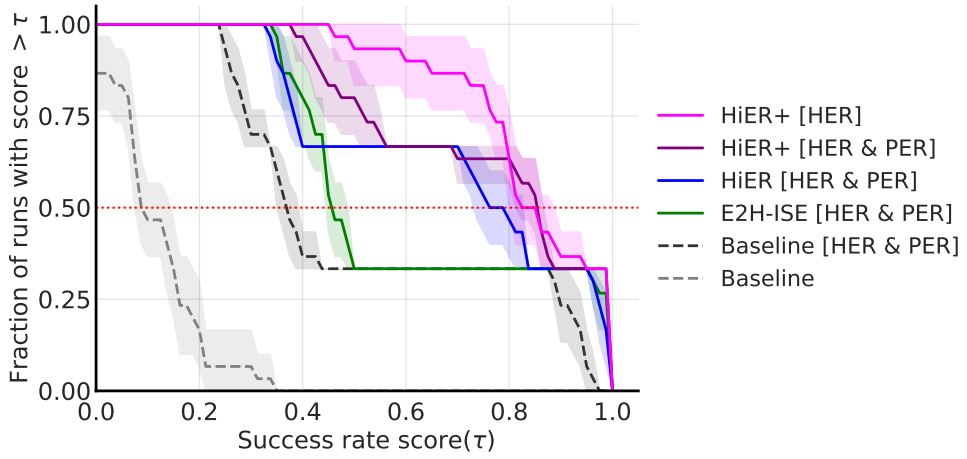


Figure 5.8: Performance profiles (run-score distribution) on the push, slide, and pick-and-place tasks of the Panda-Gym robotic benchmark with 95% CIs.

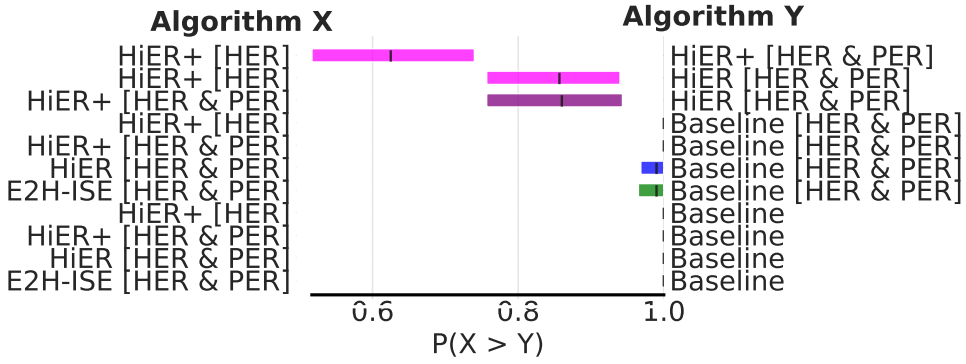


Figure 5.9: Probability of improvement on the push, slide, and pick-and-place tasks of the Panda-Gym robotic benchmark with 95% CIs. The average probabilities from top to bottom: 0.625, 0.857, 0.86, 1.0, 1.0, 0.99, 0.99, 1.0, 1.0, and 1.0.

Table 5.3: Simplified summary of our results on the push, slide, and pick-and-place tasks of the Panda-Gym robotic benchmark based on success rates. The column-wise best results are marked in bold. The full table with all the configurations is presented in Tab. E.1.

	`PandaPush-v3` | `PandaSlide-v3` | `PandaPickAndPlace-v3`		
	Mean ↑	Median ↑	IQM ↑
Baseline [HER]	0.97 | 0.38 | 0.27	0.98 | 0.37 | 0.28	0.97 | 0.37 | 0.27
HiER [HER]	**1.00** | 0.79 | 0.39	**1.00** | **0.81** | 0.39	**1.00** | 0.81 | 0.39
HiER+ [HER]	**1.00** | **0.83** | **0.69**	**1.00** | **0.81** | **0.74**	**1.00** | **0.82** | **0.71**
	OG ↓	Max ↑	STD ↓
Baseline [HER]	0.03 | 0.62 | 0.73	0.99 | 0.45 | 0.32	0.02 | **0.04** | 0.03
HiER [HER]	**0.00** | 0.21 | 0.61	**1.00** | 0.91 | 0.42	**0.00** | 0.09 | **0.02**
HiER+ [HER]	**0.00** | **0.17** | **0.31**	**1.00** | **0.95** | **0.90**	**0.00** | 0.05 | 0.14

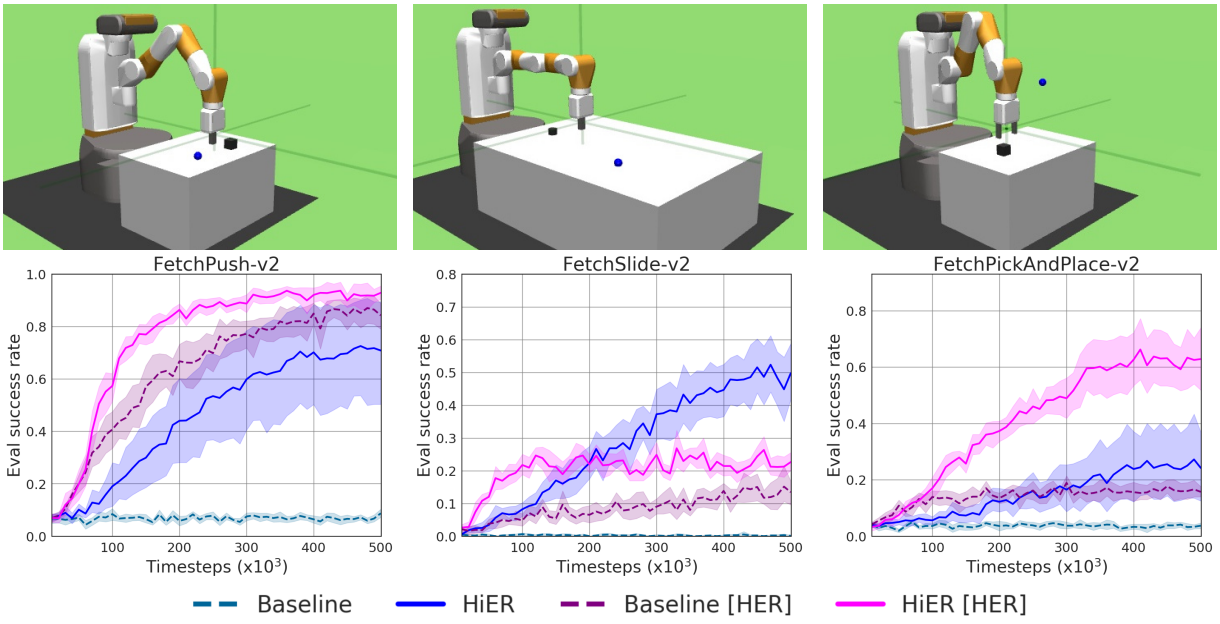


Figure 5.10: Learning curves of HiER compared with its baselines on push, slide, and pick-and-place tasks of the Gymnasium-Robotics Fetch benchmark with 95% CIs.

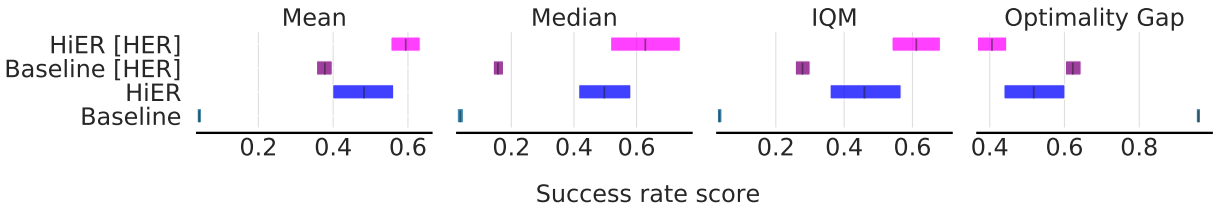


Figure 5.11: Aggregate metrics on the push, slide, and pick-and-place tasks of the Gymnasium-Robotics Fetch benchmark with 95% CIs. Both HiER (blue) and HiER [HER] (magenta) significantly outperform the baselines (light blue and purple).

Table 5.4: HiER compared to the state-of-the-art based on success rates on push, slide, and pick-and-place tasks of the Gymnasium-Robotics Fetch benchmark. The column-wise best results are marked in bold.

	HER HiER	`FetchPush-v2` | `FetchSlide-v2` | `FetchPickAndPlace-v2`		
		Mean ↑	Median ↑	IQM ↑
Baselines	- -	0.12 | 0.02 | 0.08	0.12 | 0.02 | 0.08	0.12 | 0.02 | 0.08
	✓-	0.92 | 0.23 | 0.24	0.93 | 0.22 | 0.23	0.92 | 0.22 | 0.24
HiER	- ✓	0.76 | **0.56** | 0.32	0.93 | **0.55** | 0.17	0.83 | **0.56** | 0.26
	✓✓	**0.98** | 0.35 | **0.73**	**0.99** | 0.36 | **0.77**	**0.98** | 0.36 | **0.73**
		OG ↓	Max ↑	STD ↓
Baselines	- -	0.88 | 0.98 | 0.92	0.14 | 0.04 | 0.10	**0.01** | **0.01** | **0.01**
	✓-	0.08 | 0.77 | 0.76	0.98 | 0.39 | 0.30	0.05 | 0.07 | 0.04
HiER	- ✓	0.24 | **0.44** | 0.68	**1.00** | **0.80** | 0.76	0.29 | 0.13 | 0.22
	✓✓	**0.02** | 0.65 | **0.27**	**1.00** | 0.39 | **0.93**	0.02 | 0.03 | 0.14

x and y coordinates of the final goal ball (blue). The action space $\mathcal{A} \subset \mathbb{R}^2$ represents the linear force exerted on the green ball in the x and y directions. In addition, the ball velocity is clipped in a range of $[-5, 5]$ m/s to prevent that it grows unbounded. For more details, we refer the reader to [246] and [251].

In our experiments, two different maze layouts were considered as depicted in Fig. 5.12. The reward function is changed to Eq. (2.10). As the tasks take longer to execute, the horizon is 500 timestep which is tenfold compared to the robotic manipulation tasks. Thus, for these experiments, the discount factor γ was set to one[17].

The results of our experiments are presented in Tab. 5.5 and depicted in Fig. 5.13. In the case of the `PointMaze-Wall-v3` task, the results are close to the optimal 1.0 success rate, thus there is no significant difference, even though HiER still performs equally or better than the baselines depending on the metrics and the configurations. Regarding the more challenging `PointMaze-S-v3` task, HiER [HER] outperforms Baseline [HER] by 0.2 IQM score, rising from 0.69 to 0.89.

5.4.6 Qualitative evaluation

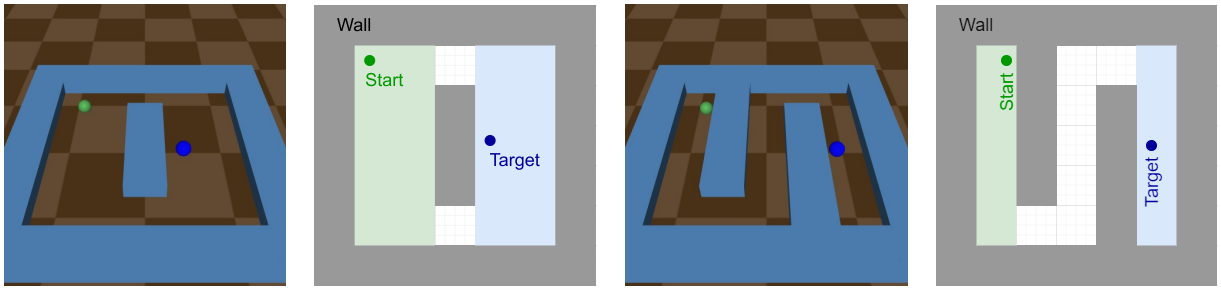
In this section, the qualitative evaluation of the aforementioned tasks is presented. We refer the reader to the project site[18] to watch our results compared with the baselines.

Regarding the Panda-Gym and the Gymnasium-Robotics Fetch environment, on many occasions, the baseline appears to be disoriented and incapable of completing the task. It appears that, during the training process, the agent became entrapped in a local minimum as a result of the challenging exploration problem caused by the continuous state and action space, the sparse reward, and the lack of demonstrations. This phenomenon is significantly less frequent in the case of HiER and HiER+ which solve the tasks with a considerably higher success rate, in correlation with the presented quantitative evaluation.

In the case of the Gymnasium-Robotics PointMaze environment, the qualitative evaluation does not show relevant differences. The primary reason is that while the mean, median, and IQM success rate score is considerably higher in the case of HiER [HER], both HiER [HER] and Baseline [HER] managed to obtain a perfect success rate of 100% at least once in the PointMaze environment (see Tab. 5.5).

[17] Not having a discount on future reward does not pose a problem as the reward function is formulated with -1 reward in every timestep, described in Eq. (2.10). Thus, the agent aims to solve the task as fast as possible.

[18] http://www.danielhorvath.eu/hier/#bookmark-qualitative-eval



(a) `PointMaze-Wall-v3` (b) Layout of `PointMaze-Wall-v3` (c) `PointMaze-S-v3` (d) Layout of `PointMaze-S-v3`

Figure 5.12: The tasks of Gymnasium-Robotics PointMaze environment [246]. The mazes were custom-made, thus we named them accordingly. The layouts (b) and (d) show the placement of the walls and the possible start and target positions from a top view. The environment is based on the MuJoCo simulator [188].

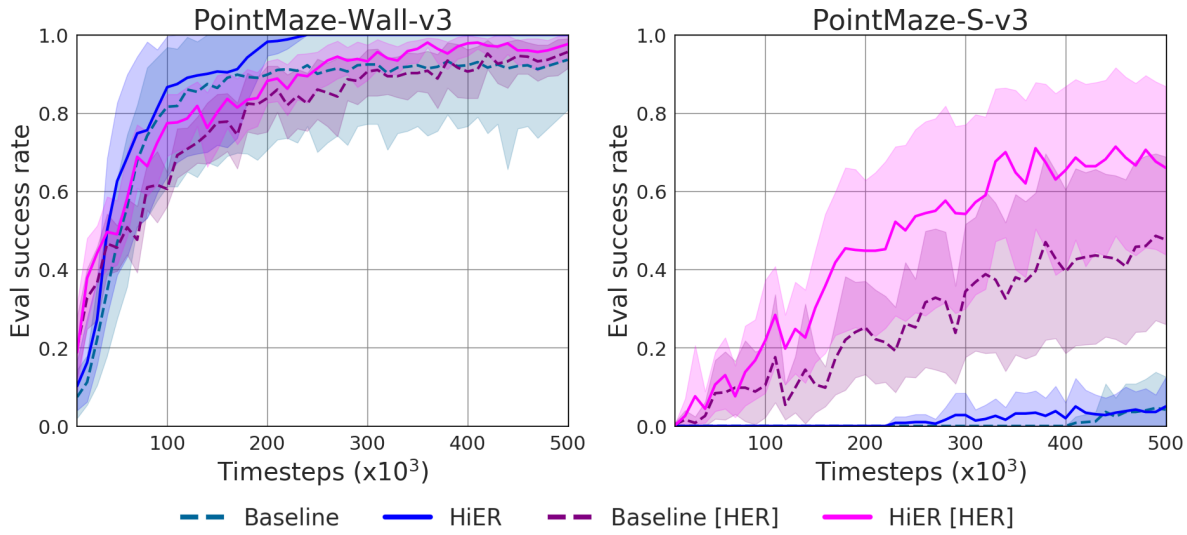


Figure 5.13: Learning curves of HiER compared with its baselines on the Gymnasium-Robotics PointMaze environment with 95% CIs.

5.4.7 HiER λ, HiER ξ, and E2H-ISE c versions

The comparison of the different HiER λ methods are depicted in Fig. 5.14a and 5.14b and Fig. 5.15. The experiments were executed without HER, PER, and E2H-ISE. In these settings, the `predefined` λ method outperforms the other variants, although its CI overlaps the CI of the `fix` λ method. The λ profiles are presented in Fig. 5.14b.

The impact of HiER ξ method is shown in Fig. 5.14c and Fig. 5.16. The experiments were executed with HER and E2H-ISE but without PER. In these settings, the `fix` $\xi = 0.25$, $\xi = 0.5$, and the `prioritized` ξ method appear to be the best versions in this order, although their CIs overlap[19]. It is important to note, that when PER is active, it scales the gradient proportionally to the probability of the samples, thus `prioritized` ξ mode is recommended to counterbalance this effect.

The different E2H-ISE c methods are presented in Tab. 5.6 and displayed in Fig. 5.17. The experiments were executed without PER. The ranking of E2H-ISE versions is relatively sensible for the applied methods (HER and HiER). Without HiER, there is no significant difference between the c methods. With HiER but without HER the `control` and the `control adaptive` c methods yield the highest performance, although their CIs overlap with the other versions. With HiER and HER, the `control adaptive` and `self-paced` c methods achieve the best performance. Nevertheless, further optimization, or possibly another version of E2H-ISE could improve the performance.

5.4.8 TD3 and DDPG

To validate our methods not only with SAC, Fig. 5.18 and Tab.5.7 show our results in the case of DDPG and TD3. In both cases, HiER+ improved the results of the baseline. In the case of TD3 (blue), the improvement is more significant as the CIs do not overlap. In the case of DDPG (magenta), although there is a considerable improvement, the CIs overlap.

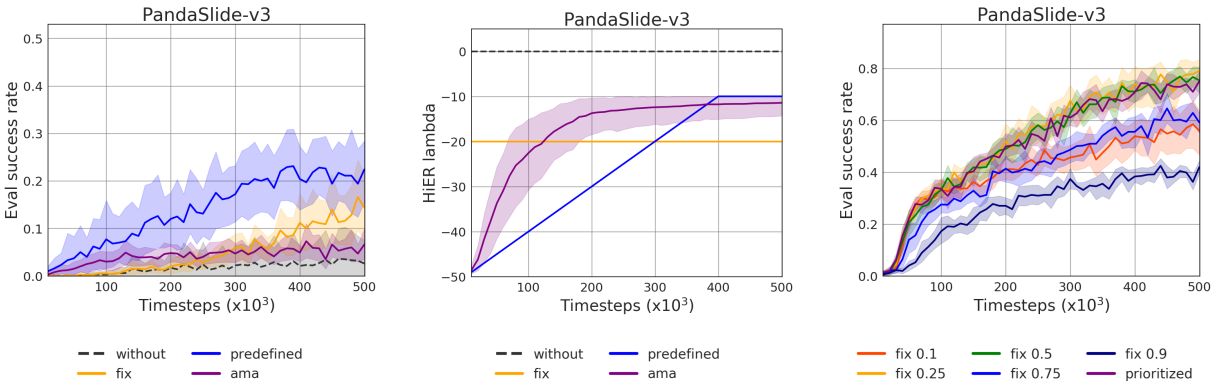
5.5 Conclusion

In this work, we introduced a novel technique called the highlight experience replay (HiER) to facilitate the training of off-policy reinforcement learning agents in a robotic, sparse-reward environment with continuous state and action spaces. Furthermore, the agent is

[19] In other settings, we found $\xi = 0.5$ slightly better than the others.

Table 5.5: HiER compared to the state-of-the-art based on success rates on the Gymnasium-Robotics PointMaze environment. The column-wise best results are marked in bold.

	HER HiER	`PointMaze-Wall-v3` | `PointMaze-S-v3` Mean ↑	Median ↑	IQM ↑
Baselines	- -	0.94 | 0.05	**1.00** | 0.00	**1.00** | 0.00
	✓-	0.97 | 0.61	**1.00** | 0.76	0.99 | 0.69
HiER	- ✓	**1.00** | 0.05	**1.00** | 0.00	**1.00** | 0.00
	✓✓	**1.00** | **0.80**	**1.00** | **0.91**	**1.00** | **0.89**
		OG ↓	Max ↑	STD ↓
Baselines	- -	0.06 | 0.95	**1.00** | 0.46	0.19 | 0.14
	✓-	0.03 | 0.39	**1.00** | **1.00**	0.05 | 0.36
HiER	- ✓	**0.00** | 0.95	**1.00** | 0.28	**0.00** | **0.11**
	✓✓	**0.00** | **0.20**	**1.00** | **1.00**	0.01 | 0.29



(a) The effect of HiER λ versions on the success rate.

(b) The change of λ values over time.

(c) The effect of HiER ξ methods. HiER λ is set to `predefined`.

Figure 5.14: The analysis of HiER λ versions (a) and (b), and HiER ξ versions (c). ξ is fixed at 0.5 for (a) and (b). HiER λ `ama` parameters: $\lambda_0 = -50$, $\lambda_{\text{max}} = -10$ $M = 0$ and $w = 20$. The `without` version indicates that HiER was not used.

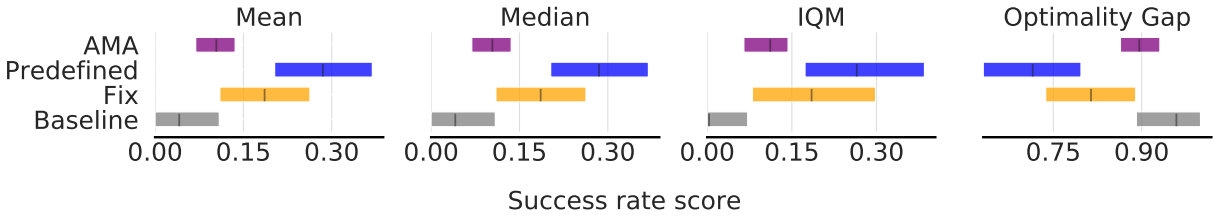


Figure 5.15: Comparison of different HiER λ methods on the slide task of the Panda-Gym benchmark with 95% CIs. The `predefined` λ method is seemingly superior, although the CIs with the `fix` λ method overlap. HiER λ `ama` parameters: $\lambda_0 = -50$, $\lambda_{\max} = -10$ $M = 0$ and $w = 20$. The profiles of HiER λ are depicted on Fig. 5.14 (b).

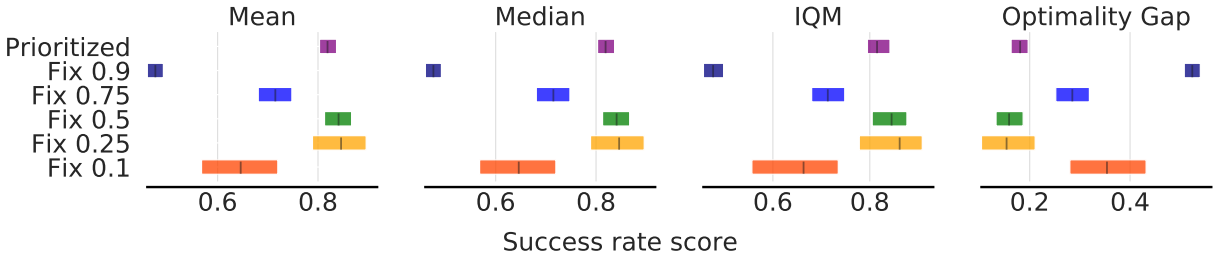


Figure 5.16: Comparison of different HiER ξ methods on the slide task of the Panda-Gym benchmark with 95% CIs. The `fix` $\xi = 0.25$, $\xi = 0.5$, and the `prioritized` appear to be the best versions in this order, although their CIs overlap.

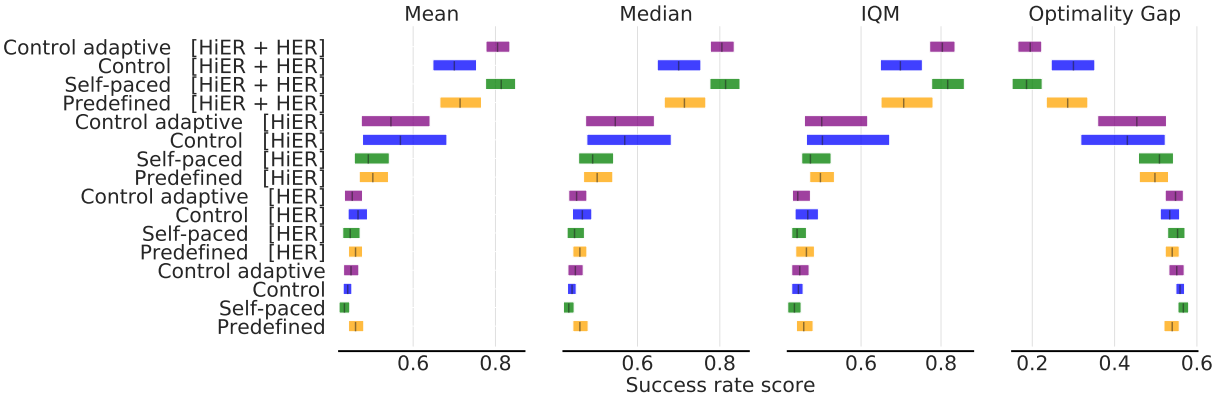


Figure 5.17: Comparison of different E2H-ISE c methods on the slide task of the Panda-Gym benchmark with 95% CIs. The parameters of the methods and the point estimates are presented in Tab. 5.6.

Table 5.6: The effect of the E2H-ISE c methods on the success rates on the `PandaSlide-v3` task. HiER parameters: λ mode `predefined` and ξ `fix` with $\xi = 0.5$. E2H-ISE parameters: `self-paced` $\Psi_{\text{low}} = 0.2$, $\Psi_{\text{high}} = 0.8$ and $\delta = 0.05$; `control`: $\psi = 0.8$ and $\delta = 0.01$; `control adaptive`: $\Delta = 0.2$, $\psi_{\max} = 0.9$, and $\delta = 0.01$. The row-wise best results are marked in bold.

Components		`predefined`			`self-paced`			`control`			`control adaptive`		
HER	HiER	Max ↑	Mean ↑	STD ↓	Max ↑	Mean ↑	STD ↓	Max ↑	Mean ↑	STD ↓	Max ↑	Mean ↑	STD ↓
-	-	**0.52**	**0.46**	0.03	0.46	0.43	0.02	0.46	0.44	0.02	0.51	0.45	0.03
✓	-	0.49	0.46	0.03	**0.54**	0.45	0.03	0.53	**0.47**	0.04	**0.54**	0.45	0.04
-	✓	0.63	0.50	0.06	0.69	0.49	0.07	**0.95**	**0.57**	0.17	0.90	0.55	0.14
✓	✓	0.85	0.71	0.08	**0.90**	**0.81**	0.06	0.87	0.70	0.08	**0.90**	0.80	0.05

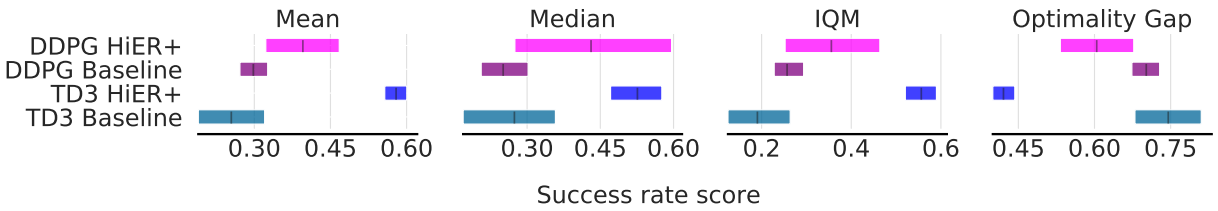


Figure 5.18: Comparison of the TD3 and DDPG versions of HiER+ with their baselines on the push, slide, and pick-and-place tasks of the Panda-Gym benchmark with 95% CIs. The point estimates are presented in Tab. 5.7.

Table 5.7: HiER+ compared to the state-of-the-art based on success rates on the Panda-Gym robotic benchmark in the case of TD3 and DDPG. The column-wise best results for TD3 and DDPG separately are marked in bold.

		`PandaPush-v3` | `PandaSlide-v3` | `PandaPickAndPlace-v3`		
	RL Algorithm	Mean ↑	Median ↑	IQM ↑
DDPG	Baseline	0.25 | 0.56 | 0.08	0.23 | 0.56 | 0.08	0.24 | 0.56 | 0.08
	HiER+	**0.43** | **0.63** | **0.13**	**0.32** | **0.68** | **0.12**	**0.38** | **0.68** | **0.12**
TD3	Baseline	0.40 | 0.27 | 0.09	0.28 | 0.36 | 0.09	0.36 | 0.30 | 0.09
	HiER+	**0.93** | **0.53** | **0.28**	**0.94** | **0.56** | **0.30**	**0.94** | **0.54** | **0.29**
		OG ↓	Max ↑	STD ↓
DDPG	Baseline	0.75 | 0.44 | 0.92	0.42 | 0.74 | 0.11	**0.08** | **0.11** | **0.01**
	HiER+	**0.57** | **0.37** | **0.87**	**0.91** | **0.83** | **0.20**	0.27 | 0.22 | 0.03
TD3	Baseline	0.60 | 0.73 | 0.91	0.86 | 0.44 | 0.11	0.28 | 0.16 | **0.01**
	HiER+	**0.07** | **0.47** | **0.72**	**0.99** | **0.63** | **0.35**	**0.04** | **0.08** | 0.05

devoid of access to any form of demonstration. These constraints, significantly exacerbate the difficulty of exploration.

In our method, a secondary replay buffer is created to store the most relevant experiences based on some criteria. At training, the transitions are sampled from both the standard experience replay buffer and the highlight experience replay buffer. Similarly to the hindsight experience replay (HER) and prioritized experience replay (PER), HiER can be added to any off-policy reinforcement learning algorithm. Following [97], HiER is classified as a data exploitation (or implicit) curriculum learning method.

To demonstrate the universality of HiER, it was validated on 8 tasks of three different robotics benchmarks [244]–[246] based on two different simulators [188], [192]. On one hand, among the 8 tasks, 3-3 were the same in principle (push, slide, and pick-and-place) but the robot configurations, the state spaces, and the dynamics of the environments were disparate. On the other hand, the last 2 tasks were fundamentally different as a ball needed to find a target in different mazes.

In all of the experiments, HiER significantly improved the state-of-the-art methods. Our experimental results show that HiER is especially beneficial in hard-to-solve tasks such as `PandaSlide-v3`, `FetchPickAndPlace-v2`, or `PointMaze-S-v3`.

HiER collects and stores positive experiences to improve the training process. With HiER+, we showed how HiER can benefit from a traditional, data collection curriculum learning method. Lack of general and easy-to-implement solutions, we proposed E2H-ISE, an *easy2hard* data collection CL method that requires minimal prior knowledge and controls the entropy of the initial state-goal distribution $\mathcal{H}(\mu_0)$ which indirectly controls the task difficulty. Nevertheless, applying more sophisticated CL methods in place of E2H-ISE might be beneficial in future research.

HiER+ was validated on the `PandaPush-v3`, `PandaSlide-v3`, and `PandaPickAndPlace-v3` tasks of the Panda-Gym [244] robotic benchmark. Our results show that HiER+ could further improve the performance of HiER.

Furthermore, we presented our experiments on the different λ, ξ, and c methods of HiER and E2H-ISE. On one hand, we found that in the case of HiER λ, the `predefined` version was superior. On the other hand, the rankings of the ξ and c methods are more unambiguous and depend on the applied configuration. We also showed that HiER+ improves the baselines not only with SAC but with TD3 and DDPG as well.

Additionally, the qualitative analysis revealed that HiER and HiER+ showed a reduced tendency to be trapped in local minima compared to the vanilla baseline methods.

Based on the findings presented in this chapter, our theses connected to the third

research question regarding improving state-of-the-art reinforcement learning algorithms with curriculum learning are as follows:

Thesis V: Our novel highlight experience replay (HiER) method enhances the training of reinforcement learning agents by separately storing and replaying the most relevant experiences, leading to a significant improvement in state-of-the-art performance.

Inspired by human learning, we propose HiER, the highlight experience replay method. A secondary experience replay buffer is created to store the most relevant transitions. At training, the transitions are sampled from both the standard experience replay buffer and the highlight experience replay buffer. It can be added to any off-policy RL agent and applied with or without the techniques of hindsight experience replay (HER) and prioritized experience replay (PER). HiER is depicted in Fig. 5.1 and detailed in Algorithm 1. If only positive experiences are stored in its buffer, HiER can be viewed as a special, automatic demonstration generator as well. HiER is classified as a data exploitation or implicit curriculum learning method. HiER significantly improves the performance of RL baselines, having stochastic dominance over the state-of-the-art, validated on 8 tasks of three robotic benchmarks. This thesis is associated with [2].

Thesis VI: Our novel HiER+ approach enhances our highlight experience replay (HiER) method by increasing the availability of positive experiences – achieved through controlling task difficulty – particularly during the early stages of the training.

We propose HiER+ which is an enhancement of HiER with an arbitrary data collection (traditional) curriculum learning method. The overview of HiER+ is depicted in Fig.5.1 and detailed in Algorithm 2. Furthermore, as an example of the data collection CL method, we propose E2H-ISE, a universal, easy-to-implement easy2hard data collection CL method that requires minimal prior knowledge and controls the initial state-goal entropy (ISE) distribution $\mathcal{H}(\mu_0)$ which indirectly controls the task difficulty. Our experimental results show that HiER+ further improves HiER's performance. Moreover, HiER+ demonstrates stochastic dominance over HiER, based on the results from three robotic tasks of the Panda-Gym benchmark. This thesis is associated with [2].

Chapter 6

Conclusions

Contents

6.1 Summary of thesis achievements 138
6.2 Future work 139

6.1 Summary of thesis achievements

As outlined in Section 1.3, the thesis is focused on three main research topics:

- **How to transfer knowledge from simulation to the real world in the case of object detection?** This research question is associated with our work in [1], where we designed a sim2real transfer learning method based on domain randomization for object detection (S2R-ObjDet) with which labelled synthetic datasets of arbitrary size and object types can be automatically generated. Subsequently, an object detection model is trained to detect the different types of industrial objects. With the proposed domain randomization method, we could shrink the reality gap to a satisfactory level, achieving 86.32% and 97.38% mAP_{50} scores respectively in the case of zero-shot and one-shot transfers, on our manually annotated and public dataset containing 190 real images of 920 objects (InO-10-190). Our solution fits for industrial use as the data generation process takes less than 0.5 s per image and the training lasts only around 12 h, on a GeForce RTX 2080 Ti GPU. Furthermore, it can reliably differentiate similar classes of objects by having access to only one real image for training. To our best knowledge, this was the first work satisfying these constraints. Moreover, we proposed the generalised confusion matrix (GCM) which is an adaptation of the traditional confusion matrix to object detection. It offers a solution to the shortcomings of the classical precision-recall-based mAP and F_1 score. With GCM, the misclassification error can be quantified and evaluated. For a short presentation and additional materials, we refer the reader to the project page `https://www.danielhorvath.eu/sim2real`.

- **How to extend our S2R-ObjDet method to multi-object grasp pose estimation?** This research question relates to our work in [4], where we propose two vision-based, multi-object grasp pose estimation models (MOGPE) – the real-time MOGPE-RT and the high-precision MOGPE-HP – that enable a modular training approach for multi-object grasp pose estimation by utilizing sequential phases of object detection and class-specific orientation estimation. Moreover, with the S2R-PosEst method – an extension of our S2R-ObjDet approach – we can automatically generate synthetic data for orientation estimation. Our methods yielded an 80% and a 96.67% success rate in a real-world robotic pick-and-place experiment, with the MOGPE-RT and the MOGPE-HP model respectively, using only limited real-world data. Our framework provides an industrial tool for fast data generation and model training and requires minimal data from the target distribution. For a short presentation and additional materials, we refer the reader to the project page `https://www.danielhorvath.eu/mogpe`.

- **How to improve the training process of state-of-the-art reinforcement learning algorithms with curriculum learning?** This research question is connected to our work in [2], where we proposed a data exploitation curriculum learning method, named the highlight experience replay (HiER) that creates a secondary highlight replay buffer for the most relevant experiences. For the weights update, the transitions are sampled from both the standard and the highlight experience replay buffer. It can be applied with or without the techniques of hindsight experience replay (HER) and prioritized experience replay (PER). Our method significantly improves the performance of the state-of-the-art, validated on 8 tasks of three robotic benchmarks. Furthermore, to exploit the full potential of HiER, we proposed HiER+ in which HiER is enhanced with an arbitrary data collection curriculum learning method. Furthermore, as an example of the data collection CL method, we introduced E2H-ISE, a universal, easy-to-implement *easy2hard* data collection CL method that requires minimal prior knowledge and controls the initial state-goal entropy (ISE) distribution $\mathcal{H}(\mu_0)$ which indirectly controls the task difficulty. Our implementation, the qualitative results, and a video presentation are available on the project site: http://www.danielhorvath.eu/hier/

6.2 Future work

In future work, several intriguing ideas merit exploration. The most relevant among them are as follows:

- **How can HiER contribute to knowledge transfer?** We demonstrated how HiER is beneficial in learning different robotic tasks. Nevertheless, it is worth exploring, how HiER can contribute to knowledge transfer. We are particularly interested in training two RL agents parallelly in simulation and the real world with creating a shared highlight buffer.
- **Can HiER be beneficial for multi-agent RL training?** In another line of research, HiER might be utilised in multi-agent RL training. In this scenario, instead of one RL agent, multiple RL agents are trained (in parallel) and combined in the hope of converging to a better solution. Prior to HiER, we conducted some experiments on this topic, although as our preliminary results did not show any improvement, we abandoned the idea. However, HiER might be able to connect these agents and thus aid their training.

- **Could low-reward episodes also contribute to RL training with HiER?** Low-reward episodes might be useful in learning to avoid undesirable behaviour, especially in the presence of failure states. An interesting line of research would be to either include them in HiER by changing HiER's storing condition or to dedicate an additional experience replay buffer to these episodes.

- **Is it possible to extend the S2R-ObjDet and the S2R-PosEst methods to feature detection?** Our S2R-ObjDet and S2R-PosEst methods are tailored to robotic manipulation tasks where the poses of specific objects need to be detected. Nevertheless, there are other types of robotic tasks, where specific features or reference points must be found in order to position the robot itself, e.g., finding reference points on printed circuit boards for precise positioning. Creating synthetic training data could be beneficial for such applications. Expanding our data generation methods for feature detection would offer a solution to the needs of the electronics industry or medical robotics.

- **Is it possible to increase the performance of sim2real knowledge transfer based on the feedback of the generalised confusion matrix (GCM)?** Our generalised confusion matrix (GCM) was proven to be beneficial for detecting misclassification, false positives, and false negatives in object detection. This information could also utilised not only for evaluation but for the training process as well. It is important to note, that it could work only if the misclassifications happen among the synthetic images as well – error from the test dataset cannot be used for training. Even though it appears that the misclassification error is due to the knowledge transfer, detecting them in the synthetic domain could significantly facilitate the knowledge transfer by actively modifying the synthetic training dataset. It could be a form of adversarial training or automatic curriculum learning.

- **How would the S2R-ObjDet and the S2R-PosEst methods perform in scenarios with significant occlusions?** Validating the S2R-ObjDet and S2R-PosEst methods in scenarios with significant occlusions were out of the scope of this research. Therefore, it would be valuable to conduct additional experiments to assess their performance in such scenarios.

- **Is it possible to increase the performance of sim2real knowledge transfer with textual scene description?** The textual representation of the scene could provide a shared contextual space between simulation and reality, aiding the knowledge transfer. A large language model (LLM) could be employed to generate these descriptions, and through multimodal learning, a visual-language model (VLM) [255] could be trained to establish the necessary mapping.

APPENDICES

Appendix A

Machine learning

A.1 Context

Given the lack of a universally accepted definition for human or artificial intelligence, their meanings remain relatively vague, which fuels the scientific debate around them. In this Appendix, we aim to briefly clarify the field and certain subfields of AI, as the terminology is often misused, even in scientific discussions. For moral dilemmas and on the ethical use of AI, we refer the reader to [256], and the European Union Artificial Intelligence Act [257].

Following Russel and Norvig [258], the definitions of AI can be grouped into four schools of thought along two dimensions, presented in Tab. A.1. In this thesis, following [258], we concentrate on the *acting rationally* approach.

As the definitions show, the field of AI is overly general. Thus, it is beneficial to categorise them into *rule-based* and *(machine-)learning-based* (ML) methods. The latter is also referred to as *data-driven* methods. Instead of explicitly programming them, ML focuses on developing methods that can learn from data. Commonly, an ML pipeline involves data collection, data preprocessing, model selection, training, validation, hyperparameter tuning, testing, and deployment.

In traditional or shallow ML, the feature selection is performed by an expert, meaning that the relevant input features are selected and transformed for the ML algorithm. However, in many cases, feature selection is the most challenging aspect of the problem. *Deep learning* (DL) is a subset of ML where the features of different abstraction levels are learned hierarchically, typically with a *deep neural network* (DNN). The landscape of AI is depicted in Fig. A.1.

Table A.1: Some definitions of artificial intelligence, organised into four schools of thought from [258].

Thinking Humanly: The Turing Test approach. E.g., '[The automation of] activities that we associate with human thinking, activities such as decision-making, problem-solving, learning...' [259]	**Thinking Rationally: The 'laws of thought' approach.** E.g., 'The study of mental faculties through the use of computational models.' [260]
Acting Humanly: The cognitive modeling approach. E.g., 'The art of creating machines that perform functions that require intelligence when performed by people.' [261]	**Acting Rationally: The rational agent approach.** E.g., "Computational Intelligence is the study of the design of intelligent agents." [262]

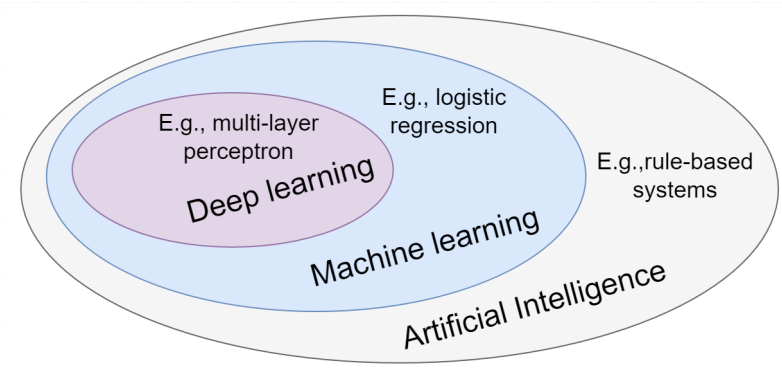


Figure A.1: Field of artificial intelligence, following [108].

A.2 Formulation

There are three fundamentally different branches of machine learning: *supervised*, *unsupervised*, and *reinforcement learning*.

In supervised learning, the algorithm learns from labelled data, meaning that the training data consists of input-output ($\boldsymbol{x}$-$\boldsymbol{y}$) pairs. The labelled output is also referred to as the *ground-truth* (GT) or the *target values*. The aim is to find the true mapping from input $\boldsymbol{x}$ to the output $\boldsymbol{y}$ as $f^*(\boldsymbol{x}) = \boldsymbol{y}$. The training dataset $(\boldsymbol{x}_{\text{train}}, \boldsymbol{y}_{\text{train}})$ should be *independent and identically distributed* (i.i.d.) and representative for the validation/test scenario. *Capacity* (the ability to fit a wide variety of functions) is an important aspect of ML methods. When the model's capacity is insufficiently low, the model struggles to approximate the

training set (*underfitting*). On the other hand, models with high capacity can learn the noise of the training data by memorising the training data points (*overfitting*), causing a gap between the training and validation results. Overfitting is a perpetual problem of DL models. Supervised learning problems can be categorised as classification or regression. In the former, the input is classified into classes, while in the latter, the target value(s) is/are real number(s).

Unsupervised learning algorithms learn patterns and structures from unlabelled data. This means that the ground truth ($\boldsymbol{y}_{\text{train}}$) is not provided for the training, only $\boldsymbol{x}_{\text{train}}$. Unsupervised learning can be applied to find the underlying (lower-level) structure of the data (e.g., principal component analysis) or perform a preprocessing step for supervised learning algorithms. Additionally, feature extraction or data normalization can be two other applications of unsupervised learning.

Reinforcement learning is formalised with a *Markov decision process* (MDP) where an *agent* interacts with an *environment*, trying to maximise its *reward* (received from the environment). Even though reinforcement learning shares similarities with supervised learning, the main differences are 1.) the challenge of delayed reward, 2.) the dilemma of exploration and exploitation, and 3.) that the training data is generated by the agent, thus it is not i.i.d. Reinforcement learning is introduced in detail in Section 2.3.

A.3 Neural networks

Artificial neural networks or *neural networks* (NNs) are a class of fundamental ML models, inspired by the human brain. In general, the goal of NNs is to approximate some function f^*. A neural network is formulated as $\boldsymbol{y} = f(\boldsymbol{x}; \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is the learnable parameter vector of the NN. Structurally, NNs consist of interconnected nodes, called (artificial) neurons, structured in layers. The outputs of the neurons in a layer are passed to the inputs of the neurons in the next layer[1].

Artificial neurons are computing system, described as $y = g\left(\boldsymbol{x}^T \boldsymbol{w}\right)$ where $\mathbf{x} \in \mathbb{R}^n$ is the input vector, $y \in \mathbb{R}$ is the output, $\mathbf{w} \in \mathbb{R}^n$ is the trainable weight vector, and $g : \mathbb{R} \to \mathbb{R}$ is a non-linear activation function. The non-linear activation function is essential to solve non-linear function approximations, e.g., XOR problem. Common activation functions are the *sigmoid*, the *hyperbolic tangent*, and the *rectified linear unit* (ReLU) along with its variations.

[1]Here, for simplicity, feedforward networks are presented.

Multilayer perceptron (MLPs) are a set of neurons structured into a network where the input of a layer is the output of the previous layer. The first and the last layers are called, the input and output layers, while the layers between them are called hidden layers. From another point of view, a layer in an MLP is function $f : \mathbb{R}^n \to \mathbb{R}^m$, where the n-dimensional input vector is $(x_1, x_2, \ldots, x_n)$ and the m-dimensional output vector is $(y_1, y_2, \ldots, y_m)$. When the layers are connected sequentially, the resulting function is formulated as $f(\boldsymbol{x}) = f^{(3)}(f^{(2)}(f^{(1)}(\boldsymbol{x})))$, in the case of 3 layers, where $f^{(i)}$ is the function of the i^{th} layer. In essence, the *universal approximation theorem* states that a feedforward neural network with a single hidden layer and sufficient neurons can approximate any continuous function on a compact domain, given appropriate weights and activation functions [108].

When training MLPs, the error signal is derived from the problem-specific loss function. The *gradient* is computed using *backpropagation*, which relies on the chain rule of calculus. Subsequently, the weight update is controlled by some version of the *stochastic gradient descent* (SGD) algorithm.

In parallel with writing this work, Liu et al. [263] proposed the *Kolmogorov-Arnold networks* (KANs), a promising alternative to MLPs. Contrary to MLPs, KANs are not based on the universal approximation theorem but the *Kolmogorov-Arnold representation theorem* [264] which states – in essence – that any multivariate continuous function can be represented as a finite sum of univariate continuous functions and their compositions, under specific conditions. In consequence, in KAN, the linear weights of MLPs are replaced with spline-based learnable univariate functions on the edges of the network and the nodes became additions (without any non-linear activation). Nevertheless, in this work, the term NN hereafter refers specifically to MLPs.

A.4 Deep learning

To exploit the potential of NNs, it is beneficial to increase their capacity by introducing more nodes in the network. Increasing the depth (number of hidden layers) is superior to increasing the width (number of neurons per layer) as it enables the NN to learn complex concepts by building on top of simple ones. Thus, the model learns a hierarchical representation of the input data. This strategy is referred to as deep learning, and with it, complex functions can be approximated. E.g., the feature selection can be automatised in computer vision applications, the first layers might learn low-level edge detection, then subsequent layers detect corners and shapes, and finally, the layers at the end learn high-level features such as face recognition.

"This [deep learning] solution is to allow computers to learn from experience and understand the world in terms of a hierarchy of concepts, with each concept defined through its relation to simpler concepts." [108]

As DL models, in general, have significantly higher capacity than needed for their problems, there is a high risk of overfitting the training data which could lead to inferior generalization. The main idea is to allow high capacity and apply various methods to avoid overfitting, e.g., *parameter regularization* when an extra term is added to the loss function controlling the magnitude of the weight vector or the technique of *dropout* when neurons are randomly turned deactivated during the training process.

Appendix B

Control theory and reinforcement learning

Reinforcement learning and control theory are closely related fields, as both aim to develop agents (controllers) that make decisions to optimize performance over time. Control theory traditionally focuses on designing controllers to regulate dynamic systems (like robots or vehicles) by minimizing deviations from a desired trajectory, using models to predict and correct errors in real time. On the other hand, an RL agent learns from interactions with the environment gathered through trial-and-error, maximizing its rewards, often without prior knowledge of the system dynamics. Generally, both control theory and RL formulate the problem as closed-loop control based on feedback from the environment. This appendix is based on [265].

As the terms and their formulations are different in control theory and RL, Tab. B.1 shows a comparison of key terms and their formulations. It is important to note that, we have preserved the original notations from the literature: terms from control theory are shown in bold to indicate their vector nature, while terms from reinforcement learning remain non-bold.

In control theory, a nonlinear, discrete-time dynamical system is generally considered, represented as in Eq. (B.1).

$$\boldsymbol{x}_{k+1} = \boldsymbol{f}_k(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{w}_k), \tag{B.1}$$

where $k \in \mathbb{N}$ is the discrete-time index, $\boldsymbol{x}_k \in \mathcal{X}$ is the state of state space $\mathcal{X}$, $\boldsymbol{u}_k \in \mathcal{U}$ is the input[1] of input space $\mathcal{U}$, $\boldsymbol{w}_k \in \mathcal{W}$ is the process noise, and $\boldsymbol{f}_k$ is the dynamic model of

[1] It is important to note, as it takes a model-based approach, the robot action is called input.

Table B.1: Comparison of the most relevant terms and their formulations of control theory and reinforcement learning.

Control theory		Reinforcement learning	
Term	Formulation	Term	Formulation
State	$\boldsymbol{x} \in \mathcal{X}$	State	$s \in \mathcal{S}$
Input	$\boldsymbol{u} \in \mathcal{U}$	Action	$a \in \mathcal{A}$
Process noise	$\boldsymbol{w} \in \mathcal{W}$		
Cost	$l : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$	Reward	$r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$
System model	$\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, \boldsymbol{w})$	Transition prob.	$p(s', r \mid s, a)$
Prior sys. model	$\bar{\boldsymbol{f}}(\boldsymbol{x}, \boldsymbol{u})$		
Model uncertainty	$\hat{\boldsymbol{f}}(\boldsymbol{x}_k, \boldsymbol{u}, \boldsymbol{w})$		

the robot. In addition, $\boldsymbol{f}$ is traditionally decomposed into a component representing prior knowledge and an unknown component as in Eq. (B.2).

$$\boldsymbol{f}_k(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{w}_k) = \bar{\boldsymbol{f}}_k(\boldsymbol{x}_k, \boldsymbol{u}_k) + \hat{\boldsymbol{f}}_k(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{w}_k), \tag{B.2}$$

where $\bar{\boldsymbol{f}}$ is the prior dynamics model and $\hat{\boldsymbol{f}}$ represents the uncertain dynamics. A general assumption is that $\bar{\boldsymbol{f}}$ and $\hat{\boldsymbol{f}}$ are Lipschitz continuous – meaning that a bounded change in input results a bounded change in output – helping in ensuring that solutions to the system do not diverge uncontrollably. According to this formulation, control theory leverages our prior knowledge and the data collected from the system to find an optimal controller. For analyzing and ensuring the stability of the control system, the Lyapunov functions are often utilized. A Lyapunov function is a scalar function $V^L(x)$ which maps the system's states to non-negative *energy-like* values typically satisfying $V^L(x) > 0$ for all $x \neq 0$ and $V^L(0) = 0$. The Lyapunov function represents how far is the system from the equilibrium. The system moves towards or remains at equilibrium rather than diverging if $\dot{V}^L(x) \leq 0$ (Lyapunov stability). For asymptotic stability, $\dot{V}^L(x) < 0$ for all $x \neq 0$ ensuring that the system state will eventually return to the equilibrium.

On the other hand, in RL, the time index is usually annotated with t, the state with $s_t \in \mathcal{S}$, the action (input) with $a_t \in \mathcal{A}$, and the transition function (system model) with $p(s', r \mid s, a)$. RL tackles the problem with sequential decision-making in a Markov decision process (MDP) under uncertainty. Opposite to traditional control theory, RL generally does not rely on the priory system model $\bar{\boldsymbol{f}}$ (the knowledge of the transition function). The RL agent directly interacts with the environment ($\boldsymbol{f}$) by initially sampling random actions (inputs) in an attempt to find the optimal policy (controller). The detailed description of the RL formalisation is presented in Section 2.3 and Appendix D.

The optimisation is based on the cost function $l : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$ (control theory) or the reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ (RL). The main difference between them is that the cost function is usually to be minimized while the reward function is to be maximized.

As it is apparent from the formalization, control theory takes a model-driven approach while RL attempts to tackle the same problem with a data-driven approach. Furthermore, while traditional control theory focuses on minimizing immediate deviations from a desired state, RL is oriented toward maximizing long-term rewards, making it versatile for applications where outcomes unfold over time. Nevertheless, alternative advanced control methods – such as model predictive control (MPC) – optimize the control of dynamic processes over a given time horizon, predicting future states by modelling the behaviour of the system and solving an optimization problem at each control step [266]. As a result of recent trends such as MPC and the introduction of more data-driven solutions to control theory, the boundary between the fields of control theory and RL has become less distinct.

Appendix C

Safe reinforcement learning

A key challenge of learning in robotics is the guarantee of the safety of the robot's behaviour. The robot must avoid causing harm to its surroundings – especially when there is human presence – and should also protect itself from potential hardware failures. The most significant obstacle to industrial use of RL algorithms is their safety guarantees. Therefore, this Appendix presents some research directions on safe learning. For a more comprehensive review, readers are encouraged to consult survey articles, such as [265], which served as the foundation for this Appendix.

The safety guarantees are derived from the assumptions and structure embodied in the problem formalization. Recent works addressing safe learning in robotics can be grouped into the fields of control theory and reinforcement learning, although their boundaries are becoming increasingly blurred, as discussed in Appendix B. Traditionally, the former takes a model-driven approach while the latter attempts to tackle the same problem with a data-driven approach – depcted on Fig. C.1 [265].

In order to ensure safe operation, safety constraints are introduced which might include:

- State constraints. $\mathcal{X}_c \subset \mathcal{X}$ (control theory) or $\mathcal{S}_c \subset \mathcal{S}$ (RL).
- Input or action constraints. $\mathcal{U}_c \subset \mathcal{U}$ (control theory) or $\mathcal{A}_c \subset \mathcal{A}$ (RL).
- Other stability guarantees.

The safety constraints might be defined with n_c constraints functions $\boldsymbol{c}_k(\boldsymbol{x}_k, \boldsymbol{u}_k, \boldsymbol{w}_k) \in \mathbb{R}^{n_c}$. Brunke et al. [265] defines three levels of constraints: soft (safety level I), probabilistic (safety level II), and hard (safety level III) – depicted in Fig. C.2.

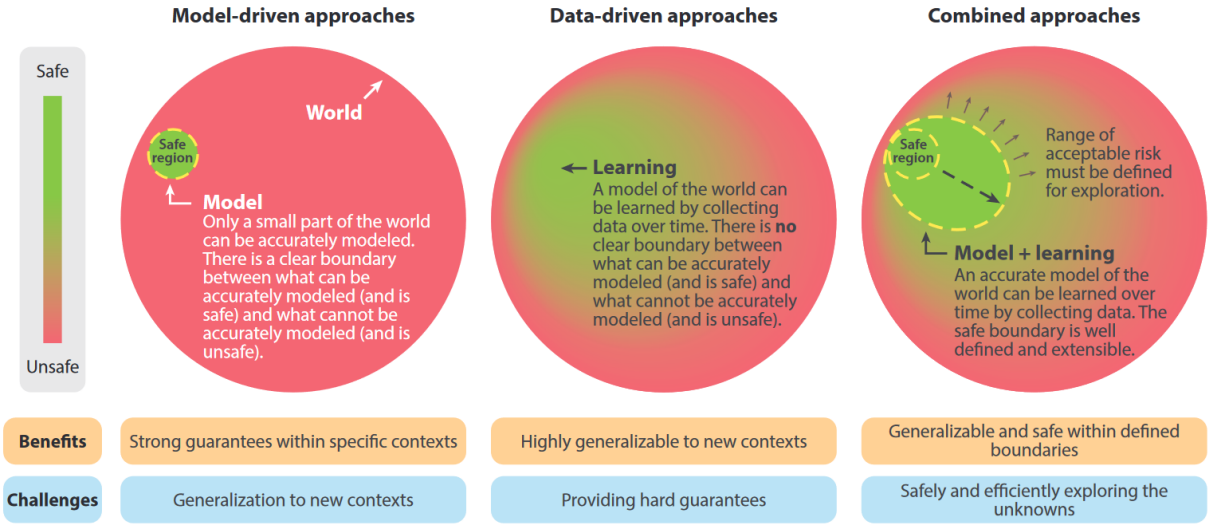


Figure C.1: A comparison of model-driven, data-driven, and combined approaches from [265].

In control theory, a typical assumption is that a prior model of the system is available. In *adaptive control* the controller parameters are modified online based on observed system behaviour. In *robust control* – such as H_2 and H_∞ methods – the aim is to find a suitable controller for all possible disturbances (assuming the worst-case scenario). In robust control, having the initial design, the controller is kept unchanged [267]. Model predictive control (MPC) optimize the control of dynamic processes over a given time horizon, predicting future states by modelling the behaviour of the system and solving an optimization problem at each control step. Robust MPC ensures state and input constraints $\boldsymbol{x}_k \in \mathcal{X}$ and $\boldsymbol{u}_k \in \mathcal{U}$, for all possible bounded disturbance. It solves a constrained optimization problem over a control input sequence and applies the first optimal control input to the system [266]. A typical approach in robust MPC is the tube-based MPC [268].

On the other hand, most state-of-the-art RL algorithms in robotics are generally model-free meaning that they do not have any prior knowledge of the system dynamics (transition probability function). In RL, constrained MDPs (CMDPs) [269] or robust MDPs [270] can be utilized to add safety constraint to traditional MDPs. The objective of a traditional RL agent in an MDP is to optimize the discounted reward which is noted as G_t^{disc} in the main text. However, to emphasize its dependence on the policy, here, it is noted as $\mathcal{J}(\pi)$

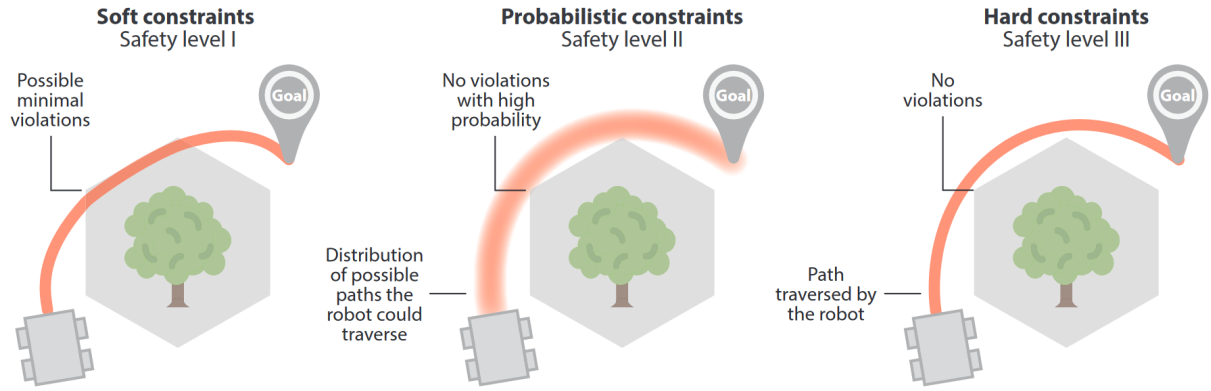


Figure C.2: Safety levels from [265].

and formulated as in Eq. (C.1).

$$\mathcal{J}(\pi) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t r(s_t, a_t, s_{t+1})\right] \tag{C.1}$$

The formulation of a constrained CMDPs extend traditional MDPs ($\mathcal{S}$,$\mathcal{A}$, r, γ, p, μ_0) with a set of cost functions, C_1, C_2, ..., C_n where $C_i : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathcal{R}$. The set of feasible stationary policies for CMDP is then defined as in Eq. (C.2).

$$\Pi_C = \{\pi \in \Pi \mid \forall i, \mathcal{J}_{C_i}(\pi) \leq d_i\}, \tag{C.2}$$

where $d_i \in \mathbb{R}$. In CMDP, the objective is formulated as in Eq. (C.3).

$$\max_\theta \mathcal{J}(\pi_\theta), \text{ s.t. } \pi_\theta \in \Pi_C \tag{C.3}$$

State-wise constrained Markov decision process (SCMDP) [271] is a special type of CMDP and requires the cost for every state action transition to satisfy a hard constraint as described in Eq. (C.4).

$$\Pi_{SC} = \{\pi \in \Pi \mid \forall (s_t, a_t, s_{t+1}) \sim \tau, \forall i, C_i(s_t, a_t, s_{t+1}) \leq w_i\}, \tag{C.4}$$

where $w_i \in \mathbb{R}$. The optimization problem is as in Eq. (C.5).

$$\max_\theta \mathcal{J}(\pi_\theta), \text{ s.t. } \pi_\theta \in \Pi_{SC} \tag{C.5}$$

Dalal et al. [272] – while applying a SCMDP – add a safety layer to analytically correct the actions of a policy. Each safety signal C_i is approximated with a linear model, thus their value of the next state can be computed and the action can be corrected before it is taken.

In another line of research, a safety critic is learnt which is an action-value function Q^{π}_{safe} predicting if a proposed action can lead to unsafe conditions [273]–[275]. Srinivasan et al. [273] propose safety Q-functions for reinforcement learning (SQRL) in which a critic is learnt that evaluates whether a state-action pair leads to unsafe behaviour, under a policy that is constrained by the safety-critic itself. This approach attempts to improve universality compared to manually programmed safety constraints. In their method – depicted on Fig. C.3 – the task-agnostic models of safety are learnt first, in controlled pre-training – akin to children learning to walk – and then use these models when learning new tasks – e.g, running. In the pre-training phase, the agent is allowed to explore and learn about unsafe behaviours ("an agent can more safely learn to drive a car if it already knows how to avoid collisions" [273]) while the safety-critic and the policy are trained in parallel. In the fine-tuning phase, the policy is constrained by the safety critic. Thananjeyan et al [275] propose the recovery RL method which balances the task performance and safety by separating a task and a recovery policy. The agent follows the task policy unless it is unsafe. Safety is based on critic Q^{π}_{safe} that is pre-trained on offline data – depicted in Fig. C.4.

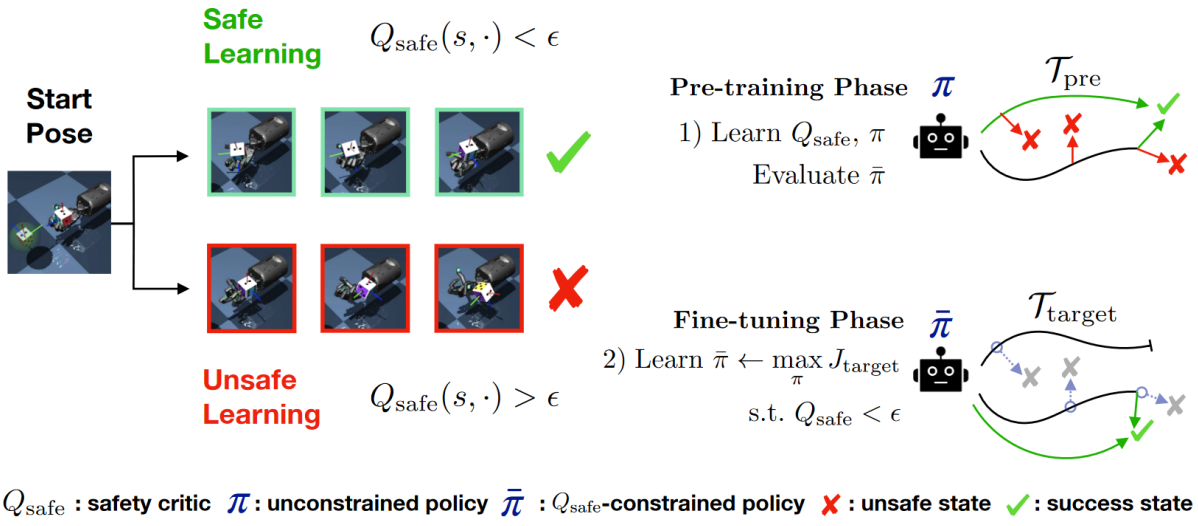


Figure C.3: The SQRL approach from [273].

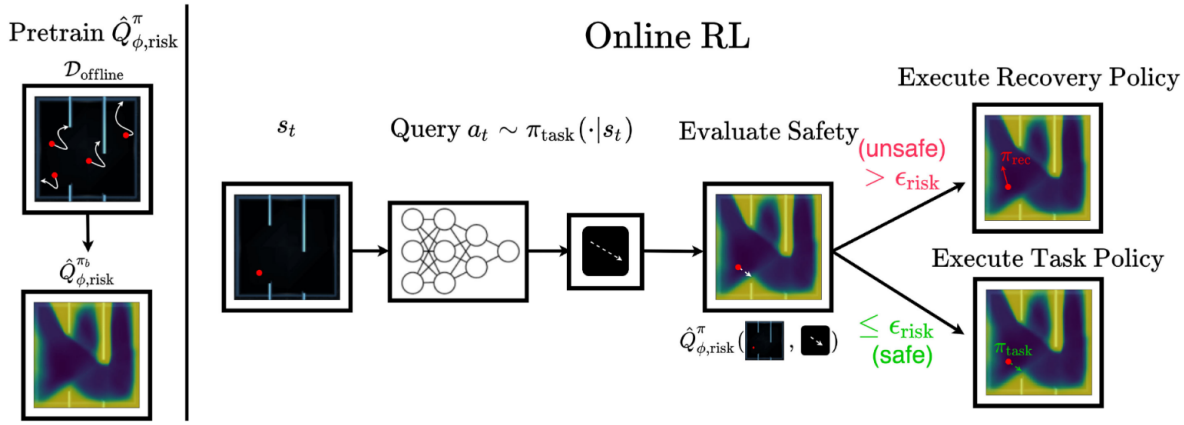


Figure C.4: The recovery RL approach from [275].

Wabersich et al. [276] introduced a model predictive safety certification (MPSC) for linear systems, which was later extended to non-linear systems with continuous state and input (action) spaces in [277], [278]. They propose the MPC-based predictive safety filter (PSF) method [278] which transforms safety-critical systems into safe systems – depicted in Fig. C.5. Compared to traditional MPC, the PSF verifies the input (action) proposed by the agent and modifies (filters) it – as little as possible – if it is necessary to ensure safe operation. It is worth mentioning that the method can be utilized with arbitrary RL algorithms or even with human agents (e.g., as a driving assistant). Their method provides safety certificates for RL algorithms. In their formulation: "a learning-based input action is certified as safe if it leads to a safe state, i.e., a state for which a potentially low-performance, but online computable and safe backup controller exists for all future times." [277].

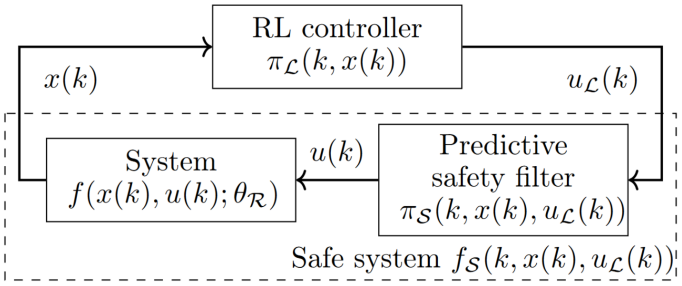


Figure C.5: The predictive safety filter from [278].

Finally, a remaining issue is when to enforce safety assurance. Following [271], there are three main cases depicted in Fig. C.6. When we only need the model to converge to a safe policy, soft constraints might be sufficient as there is no need for hard constraints throughout the training – e.g., the robot is trained in a simulator before real-world deployment. On the other hand, there are cases when the soft constraints are not sufficient – e.g., real-world training. In these scenarios, either having fixed hard constraints or starting from a set of safe states (and actions) and gradually extending it based on information gained by the training.

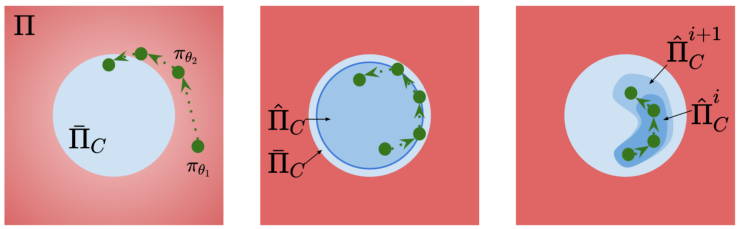


Figure C.6: Illustrations of three different notions of state-wise safety from [271]. **Left.** safety after convergence. **Middle.** safety during training with hard constraints. **Right.** safety during training with progressive safe exploration.

Appendix D

Tabular reinforcement learning

In tabular RL, typically value-based RL methods are utilised with the *general policy iteration* method which involves perpetually repeating *policy evaluation* and *policy improvement* until convergence. Policy evaluation is a technique to approximate the value function of a policy $v_\pi(s)$ or $q_\pi(s,a)$, while the policy improvement is to improve the policy $\pi(a|s)$ based on its value function.

In the easiest case, when the state and action spaces are small enough, and the dynamics of the environment (the $p(s_{t+1}, r_t \mid s_t, a_t)$ transition probability function) are known, then *dynamic programming* (DP) [143] can be applied to calculate $v_\pi(s)$ or $q_\pi(s,a)$ with a technique called *bootstrapping*, in which the value of a state (or state-action pair) is computed based on the Bellman equation, presented in Eq. (D.1). The value of a state is calculated based on its neighbor states. When $v_\pi(s)$ or $q_\pi(s,a)$ is computed, the agent chooses the greedily the best actions. The new policy is given by the Bellman optimality equation, see in Eq. (D.2). It is important to note that in DP there is no real interaction between the agent and the environment, everything is computed knowing the dynamics of the environment. In practice, many problems are not solvable by DP methods because either the environmental dynamics are unknown or the state and action spaces are excessively large.

$$v_\pi(s) \doteq \sum_a \pi(a \mid s) \sum_{s',r} \left[p(s', r \mid s, a) \left[r + \gamma v_\pi(s')\right]\right] \tag{D.1}$$

$$\pi(s) \doteq \arg\max_a \sum_{s',r} \left[p(s', r \mid s, a) \left[r + \gamma v_\pi(s')\right]\right] \tag{D.2}$$

Monte Carlo (MC) methods [73] is another set of algorithms to solve tabular RL problems. In this case, the agent learns the $v_\pi(s)$ or $q_\pi(s,a)$ with trial-and-error. It is beneficial to learn $q_\pi(s,a)$, especially when the transition probability function is unknown. Starting from an arbitrary random policy, the agent interacts with the environment, collects rewards, and updates its value function based on its experience. The MC return is computed as in Eq. (D.3). It is important to note, that in MC methods, the agent does not need to know the transition probability function (the dynamics of the environment), but it must go until the end of an episode to update its value function and it does not use bootstrapping. Two main versions are every-visit and first-visit MC methods [279].

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \ldots + \gamma^{T-t-1} R_T \tag{D.3}$$

Temporal difference (TD) [247] methods combine the advantages of DP and MC. On one hand, they learn with trial-and-error and do not require access to the $p(s_{t+1}, r_t \mid s_t, a_t)$ transitional probability function, similar to MC methods. On the other hand, they do not wait until the end of the episode as they apply bootstrapping, similar to DP methods. The simplest method is called one-step TD or TD(0). The Q value update is presented in Eq. (D.4). Furthermore, n-step TD methods or TD(λ) methods enable bootstrapping to occur over multiple steps. The return is computed as presented in Eq. (D.5). The TD(∞) methods are equivalent to the MC methods. The most famous TD methods are SARSA, Q-learning, and Expected SARSA [280]. To avoid the *maximization bias*, *double learning* (e.g., Double Q-learning) can be utilised. In double learning, there are two parallel Q functions Q_1 and Q_2. For updating Q_1 the bootstrapping uses de value of Q_2, and for updating Q_2, the bootstrapping is based on Q_1.

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t) \right] \tag{D.4}$$

$$G_t \doteq R_{t+1} + \gamma R_{t+2} + \ldots + \gamma^{n-1} R_{t+n} + \gamma^n Q_{t+n-1}(S_{t+n}, A_{t+n}) \tag{D.5}$$

Appendix E

Detailed results of HiER and HiER+

In this appendix, the detailed results of HiER and HiER+ on the Panda-Gym robotic benchmark are presented. Tab. E.1 displays the results based on success rate, while Tab. E.2 shows the results based on rewards.

Table E.1: HiER and HiER+ compared to the state-of-the-art based on success rates on the Panda-Gym robotic benchmark. On the left side of the header, the components of the specific algorithm are displayed (HER, PER, ISE, HiER). The column-wise best results are marked in bold.

	HER PER ISE HiER	`PandaPush-v3` | `PandaSlide-v3` | `PandaPickAndPlace-v3`				
		Mean ↑	Median ↑	IQM ↑	OG ↓	Max ↑
Baselines	- - - -	0.16 | 0.12 | 0.07	0.15 | 0.05 | 0.07	0.15 | 0.08 | 0.07	0.84 | 0.88 | 0.93	0.21 | 0.34 | 0.09
	✓- - -	0.97 | 0.38 | 0.27	0.98 | 0.37 | 0.28	0.97 | 0.37 | 0.27	0.03 | 0.62 | 0.73	0.99 | 0.45 | 0.32
	- ✓- -	0.26 | 0.25 | 0.08	0.25 | 0.27 | 0.09	0.25 | 0.27 | 0.08	0.74 | 0.75 | 0.92	0.43 | 0.42 | 0.10
	✓✓- -	0.93 | 0.37 | 0.28	0.94 | 0.37 | 0.28	0.93 | 0.37 | 0.27	0.07 | 0.63 | 0.72	0.97 | 0.43 | 0.33
HiER	- - - ✓	0.44 | 0.29 | 0.09	0.44 | 0.28 | 0.09	0.44 | 0.29 | 0.09	0.56 | 0.71 | 0.91	0.57 | 0.39 | 0.11
	✓- - ✓	**1.00** | 0.79 | 0.39	**1.00** | 0.81 | 0.39	**1.00** | 0.81 | 0.39	**0.00** | 0.21 | 0.61	**1.00** | 0.91 | 0.42
	- ✓- ✓	0.80 | 0.41 | 0.13	0.88 | 0.42 | 0.13	0.83 | 0.44 | 0.13	0.20 | 0.59 | 0.87	0.98 | 0.66 | 0.16
	✓✓- ✓	0.98 | 0.78 | 0.37	0.99 | 0.78 | 0.37	0.99 | 0.78 | 0.37	0.02 | 0.22 | 0.63	**1.00** | 0.83 | 0.39
ISE	- - ✓-	0.85 | 0.45 | 0.25	0.85 | 0.45 | 0.25	0.86 | 0.45 | 0.25	0.15 | 0.55 | 0.75	0.95 | 0.47 | 0.30
	✓- ✓-	**1.00** | 0.45 | 0.42	**1.00** | 0.45 | 0.43	**1.00** | 0.45 | 0.43	**0.00** | 0.55 | 0.58	**1.00** | 0.52 | 0.53
	- ✓✓-	0.83 | 0.44 | 0.31	0.83 | 0.44 | 0.30	0.83 | 0.44 | 0.30	0.17 | 0.56 | 0.69	0.89 | 0.52 | 0.36
	✓✓✓-	0.99 | 0.46 | 0.39	**1.00** | 0.46 | 0.38	**1.00** | 0.46 | 0.39	0.01 | 0.54 | 0.61	**1.00** | 0.50 | 0.44
HiER+	- - ✓✓	0.98 | 0.53 | 0.33	0.99 | 0.48 | 0.32	0.99 | 0.49 | 0.32	0.02 | 0.47 | 0.67	**1.00** | 0.76 | 0.39
	✓- ✓✓	**1.00** | 0.83 | **0.69**	**1.00** | 0.81 | **0.74**	**1.00** | 0.82 | **0.71**	**0.00** | 0.17 | **0.31**	**1.00** | **0.95** | **0.90**
	- ✓✓✓	0.98 | 0.51 | 0.41	0.98 | 0.49 | 0.40	0.98 | 0.50 | 0.40	0.02 | 0.49 | 0.59	**1.00** | 0.65 | 0.50
	✓✓✓✓	**1.00** | **0.84** | 0.47	**1.00** | **0.86** | 0.45	**1.00** | **0.85** | 0.46	**0.00** | **0.16** | 0.53	**1.00** | 0.88 | 0.55

Table E.2: HiER and HiER+ compared to the state-of-the-art based on the evaluation rewards on the Panda-Gym robotic benchmark. On the left side of the header, the components of the specific algorithm are displayed (HER, PER, ISE, HiER). The desired performance scores for the OG metric are -10, -20, and -30 for the push, slide, and pick-and-place tasks respectively. The column-wise best results are marked in bold.

	HER PER ISE HiER	`PandaPush-v3` | `PandaSlide-v3` | `PandaPickAndPlace-v3`														
		Mean ↑			Median ↑			IQM ↑			OG ↓			Max ↑		
Baselines	- - - -	-46.2	-46.7	-48.2	-46.2	-49.0	-48.5	-46.4	-48.0	-48.3	36.2	26.7	18.2	-41.0	-37.6	-46.5
	✓- - -	-11.4	-39.2	-41.8	-11.2	-38.5	-41.1	-11.2	-38.7	-41.5	1.5	19.2	11.8	-9.6	-36.5	-39.5
	- ✓- -	-40.8	-43.5	-48.8	-41.6	-43.4	-48.5	-41.4	-43.0	-48.7	30.8	23.5	18.8	-32.6	-38.3	-48.0
	✓✓- -	-12.7	-38.5	-40.4	-11.9	-38.7	-39.8	-12.3	-38.6	-40.1	2.7	18.5	10.4	-9.7	-35.5	-36.6
HiER	- - - ✓	-34.1	-42.0	-47.6	-35.0	-42.0	-47.8	-34.4	-42.5	-47.7	24.1	22.0	17.6	-27.0	-35.6	-46.2
	✓- - ✓	-7.0	-23.6	-37.2	**-6.8**	-22.6	-36.6	**-6.9**	-23.1	-36.9	**0.0**	4.0	7.2	-6.1	-17.8	-34.5
	- ✓- ✓	-17.2	-38.2	-46.8	-14.5	-36.6	-46.7	-15.7	-36.8	-46.7	7.2	18.2	16.8	-9.9	-32.9	-45.1
	✓✓- ✓	-8.4	-25.4	-37.7	-8.3	-24.9	-37.3	-8.3	-25.1	-37.5	0.1	5.4	7.7	-6.4	-21.2	-36.2
ISE	- - ✓-	-14.9	-37.3	-41.6	-15.0	-37.8	-41.6	-15.1	-37.4	-41.4	5.0	17.3	11.6	-8.3	-33.6	-38.8
	✓- ✓-	-8.1	-35.9	-34.1	-8.0	-35.9	-34.6	-8.0	-35.9	-34.4	**0.0**	15.9	4.3	-6.7	-32.7	-27.2
	- ✓✓-	-16.3	-37.3	-39.1	-15.7	-37.5	-39.0	-16.1	-37.5	-38.9	6.3	17.3	9.1	-12.1	-34.8	-35.8
	✓✓✓-	-8.1	-37.6	-36.1	-7.8	-37.7	-36.6	-8.0	-37.7	-36.5	**0.0**	17.6	6.1	-6.4	-34.9	-31.6
HiER+	- - ✓✓	-8.8	-33.4	-38.1	-8.2	-33.1	-38.5	-8.5	-33.8	-38.3	0.3	13.4	8.1	-7.1	-26.6	-34.8
	✓- ✓✓	**-6.9**	-22.5	**-24.2**	-7.0	-22.9	**-23.2**	-7.0	-22.8	**-23.0**	**0.0**	3.0	**1.0**	**-5.9**	**-17.4**	**-15.0**
	- ✓✓✓	-8.6	-34.9	-34.6	-8.4	-35.9	-34.8	-8.5	-35.3	-34.6	**0.0**	14.9	4.6	-7.8	-30.5	-30.9
	✓✓✓✓	-7.7	**-21.7**	-33.9	-7.4	**-21.5**	-34.4	-7.5	**-21.5**	-34.0	**0.0**	**2.3**	4.1	-7.0	**-17.4**	-28.0

References

Author's journal papers

[1] **D. Horváth**, G. Erdős, Z. Istenes, T. Horváth, and S. Földi, "Object Detection Using Sim2Real Domain Randomization for Robotic Applications," *IEEE Transactions on Robotics*, vol. 39, no. 2, pp. 1225–1243, Apr. 2023, ISSN: 1941-0468. DOI: 10.1109/TRO.2022.3207619.

[2] **D. Horváth**, J. Bujalance Martín, F. Gábor Erdős, Z. Istenes, and F. Moutarde, "HiER: Highlight Experience Replay for Boosting Off-Policy Reinforcement Learning Agents," *IEEE Access*, vol. 12, pp. 100 102–100 119, Jul. 2024, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2024.3427012.

[3] G. Erdős, K. Abai, R. Beregi, *et al.*, "Enabling Technologies for Autonomous Robotic Systems in Manufacturing," *Transactions of Nanjing University of Aeronautics and Astronautics*, vol. 41, no. 4, pp. 403–431, Aug. 2024, ISSN: 1005-1120. DOI: 10.16356/j.1005-1120.2024.04.001.

Author's conference papers

[4] **D. Horváth**, K. Bocsi, G. Erdős, and Z. Istenes, "Sim2Real Grasp Pose Estimation for Adaptive Robotic Applications," in *the 22nd IFAC World Congress*, ser. IFAC-PapersOnLine, vol. 56, 2023, pp. 5233–5239. DOI: 10.1016/j.ifacol.2023.10.121.

[5] G. Erdős, **D. Horváth**, and G. Horváth, "Visual Servo Guided Cyber-Physical Robotic Assembly Cell," in *the 17th IFAC Symposium on Information Control Problems in Manufacturing (INCOM)*, ser. IFAC-PapersOnLine, vol. 54, Jan. 2021, pp. 595–600. DOI: 10.1016/j.ifacol.2021.08.068.

[6] M. Hajós and **D. Horváth**, "Robotos Pakolási Feladat Megoldása Környezetérzékelés Segítségével," in *Nemzetközi Gépészeti Konferencia (OGÉT)*, Apr. 2020, pp. 305–308. [Online]. Available: https://ojs.emt.ro/oget/article/view/156.

[7] Z. Kemény, R. Beregi, J. Nacsa, C. Kardos, and **D. Horváth**, "Human–Robot Collaboration in the MTA SZTAKI Learning Factory Facility at Győr," in *the 8th CIRP Sponsored Conference on Learning Factories (CLF)*, ser. Procedia Manufacturing, vol. 23, Jan. 2018, pp. 105–110. DOI: 10.1016/j.promfg.2018.04.001.

[8] Z. Kemény, R. Beregi, J. Nacsa, C. Kardos, and **D. Horváth**, "Example of a Problem-to-Course Life Cycle in Layout and Process Planning at the MTA SZTAKI Learning Factories," in *the 9th Conference on Learning Factories (CLF)*, ser. Procedia Manufacturing, vol. 31, Jan. 2019, pp. 206–212. DOI: 10.1016/j.promfg.2019.03.033.

Other references

[9] Y. N. Harari, *Sapiens: A Brief History of Humankind*. Harper, 2015, ISBN: 9780062316097.

[10] J. Zhou, Y. Zhou, B. Wang, and J. Zang, "Human–Cyber–Physical Systems (HCPSs) in the Context of New-Generation Intelligent Manufacturing," *Engineering*, vol. 5, no. 4, pp. 624–636, Aug. 2019, ISSN: 2095-8099. DOI: 10.1016/j.eng.2019.07.015.

[11] H. P. Moravec, *Mind Children: The Future of Robot and Human Intelligence*. Cambridge, Mass. : Harvard University Press, 1988, ISBN: 9780674576162.

[12] G. A. Bekey, *Autonomous Robots: From Biological Inspiration to Implementation and Control*. The MIT Press, 2005, ISBN: 9780262025782.

[13] Y. LeCun, *Language is Low Bandwidth*, X, Mar. 2024. [Online]. Available: https://x.com/ylecun/status/1766498677751787723 (visited on 09/10/2024).

[14] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," *arXiv*, May 2015. DOI: 10.48550/arXiv.1505.04597.

[15] M. A. Mazurowski, M. Buda, A. Saha, and M. R. Bashir, "Deep Learning in Radiology: An Overview of the Concepts and a Survey of the State of the Art with Focus on MRI," *Journal of Magnetic Resonance Imaging*, vol. 49, no. 4, pp. 939–954, Apr. 2019, ISSN: 1522-2586. DOI: 10.1002/jmri.26534.

[16] N. Gogin, M. Viti, L. Nicodème, *et al.*, "Automatic Coronary Artery Calcium Scoring from Unenhanced-ECG-gated CT using Deep Learning," *Diagnostic and Interventional Imaging*, vol. 102, no. 11, pp. 683–690, Nov. 2021, ISSN: 2211-5684. DOI: 10.1016/j.diii.2021.05.004.

[17] E. Gawehn, J. A. Hiss, and G. Schneider, "Deep Learning in Drug Discovery," *Molecular Informatics*, vol. 35, no. 1, pp. 3–14, Jan. 2016, ISSN: 1868-1751. DOI: 10.1002/minf.201501008.

[18] H. Chen, O. Engkvist, Y. Wang, M. Olivecrona, and T. Blaschke, "The Rise of Deep Learning in Drug Discovery," *Drug Discovery Today*, vol. 23, no. 6, pp. 1241–1250, Jun. 2018, ISSN: 1359-6446. DOI: 10.1016/j.drudis.2018.01.039.

[19] C. Kerepesi, B. Daróczy, Á. Sturm, T. Vellai, and A. Benczúr, "Prediction and Characterization of Human Ageing-Related Proteins by Using Machine Learning," *Scientific Reports*, vol. 8, no. 1, p. 4094, Mar. 2018, ISSN: 2045-2322. DOI: 10.1038/s41598-018-22240-w.

[20] C. Su, Z. Xu, J. Pathak, and F. Wang, "Deep Learning in Mental Health Outcome Research: A Scoping Review," *Translational Psychiatry*, vol. 10, no. 1, pp. 1–26, Apr. 2020, ISSN: 2158-3188. DOI: 10.1038/s41398-020-0780-3.

[21] D. Kuang and L. He, "Classification on ADHD with Deep Learning," in *2014 International Conference on Cloud Computing and Big Data*, Nov. 2014, pp. 27–32. DOI: 10.1109/CCBD.2014.42.

[22] S. Laksshman, R. R. Bhat, V. Viswanath, and X. Li, "DeepBipolar: Identifying Genomic Mutations for Bipolar Disorder via Deep Learning," *Human Mutation*, vol. 38, no. 9, pp. 1217–1224, Jun. 2017, ISSN: 1098-1004. DOI: 10.1002/humu.23272.

[23] A. Jan, H. Meng, Y. F. B. A. Gaus, and F. Zhang, "Artificial Intelligent System for Automatic Depression Level Analysis Through Visual and Vocal Expressions," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 10, no. 3, pp. 668–680, Sep. 2018, ISSN: 2379-8939. DOI: 10.1109/TCDS.2017.2721552.

[24] J. Kim, J. Lee, E. Park, and J. Han, "A Deep Learning Model for Detecting Mental Illness from User Content on Social Media," *Scientific Reports*, vol. 10, no. 1, p. 11 846, Jul. 2020, ISSN: 2045-2322. DOI: 10.1038/s41598-020-68764-y.

[25] J. Huang, J. Chai, and S. Cho, "Deep Learning in Finance and Banking: A Literature Review and Classification," *Frontiers of Business Research in China*, vol. 14, no. 1, p. 13, Jun. 2020, ISSN: 1673-7431. DOI: 10.1186/s11782-020-00082-6.

[26] A. M. Ozbayoglu, M. U. Gudelek, and O. B. Sezer, "Deep Learning for Financial Applications : A Survey," *Applied Soft Computing*, vol. 93, p. 106 384, Aug. 2020, ISSN: 1568-4946. DOI: 10.1016/j.asoc.2020.106384.

[27] K. Khare, O. Darekar, P. Gupta, and V. Z. Attar, "Short Term Stock Price Prediction Using Deep Learning," in *2017 2nd IEEE International Conference on Recent Trends in Electronics, Information and Communication Technology (RTEICT)*, May 2017, pp. 482–486. DOI: 10.1109/RTEICT.2017.8256643.

[28] Z. Jiang, D. Xu, and J. Liang, "A Deep Reinforcement Learning Framework for the Financial Portfolio Management Problem," *arXiv*, Jul. 2017. DOI: 10.48550/arXiv.1706.10059.

[29] G. Zioviris, K. Kolomvatsos, and G. Stamoulis, "Credit Card Fraud Detection Using a Deep Learning Multistage Model," *The Journal of Supercomputing*, vol. 78, no. 12, pp. 14 571–14 596, Aug. 2022, ISSN: 1573-0484. DOI: 10.1007/s11227-022-04465-9.

[30] W. Guettala and L. Gulyás, "On the Power of Graph Neural Networks and Feature Augmentation Strategies to Classify Social Networks," in *Intelligent Information and Database Systems*, Springer Nature, 2024, pp. 287–301, ISBN: 9789819749850. DOI: 10.1007/978-981-97-4985-0_23.

[31] E. Suel, J. W. Polak, J. E. Bennett, and M. Ezzati, "Measuring Social, Environmental and Health Inequalities Using Deep Learning and Street Imagery," *Scientific Reports*, vol. 9, no. 1, p. 6229, Apr. 2019, ISSN: 2045-2322. DOI: 10.1038/s41598-019-42036-w.

[32] J. Amankwah-Amoah, S. Abdalla, E. Mogaji, A. Elbanna, and Y. K. Dwivedi, "The Impending Disruption of Creative Industries by Generative AI: Opportunities, Challenges, and Research Agenda," *International Journal of Information Management*, vol. 79, p. 102 759, Dec. 2024, ISSN: 0268-4012. DOI: 10.1016/j.ijinfomgt.2024.102759.

[33] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems*, vol. 25, Curran Associates, Inc., 2012. [Online]. Available: https://papers.nips.cc/paper_files/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html (visited on 09/10/2024).

[34] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," *arXiv*, Apr. 2020. DOI: 10.48550/arXiv.2004.10934.

[35] A. Barisic, F. Petric, and S. Bogdan, "Sim2Air - Synthetic Aerial Dataset for UAV Monitoring," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 3757–3764, Apr. 2022, ISSN: 2377-3766. DOI: 10.1109/LRA.2022.3147337.

[36] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation," *arXiv*, Oct. 2016. DOI: 10.48550/arXiv.1511.00561.

[37] C. Henry, S. M. Azimi, and N. Merkle, "Road Segmentation in SAR Satellite Images With Deep Fully Convolutional Neural Networks," *IEEE Geoscience and Remote Sensing Letters*, vol. 15, no. 12, pp. 1867–1871, Dec. 2018, ISSN: 1558-0571. DOI: 10.1109/LGRS.2018.2864342.

[38] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to Sequence Learning with Neural Networks," in *Advances in Neural Information Processing Systems*, vol. 27, Curran Associates, Inc., 2014. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2014/hash/a14ac55a4f27472c5d894ec1c3c743d2-Abstract.html (visited on 09/10/2024).

[39] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is All you Need," in *Advances in Neural Information Processing Systems*, vol. 30, Curran Associates, Inc., 2017. [Online]. Available: https://papers.nips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html (visited on 09/10/2024).

[40] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Association for Computational Linguistics, Jun. 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423.

[41] OpenAI, J. Achiam, S. Adler, *et al.*, "GPT-4 Technical Report," *arXiv*, Mar. 2024. DOI: 10.48550/arXiv.2303.08774.

[42] J.-E. Deschaud, "IMLS-SLAM: Scan-to-Model Matching Based on 3D Data," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 2480–2485. DOI: 10.1109/ICRA.2018.8460653.

[43] S. Horache, J.-E. Deschaud, and F. Goulette, "3D Point Cloud Registration with Multi-Scale Architecture and Unsupervised Transfer Learning," in *2021 International Conference on 3D Vision (3DV)*, Dec. 2021, pp. 1351–1361. DOI: 10.1109/3DV53792.2021.00142.

[44] B. Mildenhall, P. P. Srinivasan, M. Tancik, *et al.*, "NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis," *arXiv*, Aug. 2020. DOI: 10.48550/arXiv.2003.08934.

[45] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3D Gaussian Splatting for Real-Time Radiance Field Rendering," *arXiv*, Aug. 2023. DOI: 10.48550/arXiv.2308.04079.

[46] J. Sanchez, J.-E. Deschaud, and F. Goulette, "Domain Generalization of 3D Semantic Segmentation in Autonomous Driving," in *International Conference on Computer Vision*, 2023, pp. 18 077–18 087. [Online]. Available: https://openaccess.thecvf.com/content/ICCV2023/html/Sanchez_Domain_Generalization_of_3D_Semantic_Segmentation_in_Autonomous_Driving_ICCV_2023_paper.html (visited on 09/10/2024).

[47] L. Soum-Fontez, J.-E. Deschaud, and F. Goulette, "MDT3D: Multi-Dataset Training for LiDAR 3D Object Detection Generalization," *arXiv*, Aug. 2023. DOI: 10.48550/arXiv.2308.01000.

[48] A. Brunetto, S. Hornauer, and F. Moutarde, "NeRAF: 3D Scene Infused Neural Radiance and Acoustic Fields," *arXiv*, May 2024. DOI: 10.48550/arXiv.2405.18213.

[49] S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen, "Learning Hand-Eye Coordination for Robotic Grasping with Deep Learning and Large-Scale Data Collection," *The International Journal of Robotics Research*, Mar. 2016, ISSN: 0278-3649. DOI: 10.1177/0278364917710318.

[50] J. Tobin, L. Biewald, R. Duan, *et al.*, "Domain Randomization and Generative Models for Robotic Grasping," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2018, pp. 3482–3489. DOI: 10.1109/IROS.2018.8593933.

[51] OpenAI, I. Akkaya, M. Andrychowicz, *et al.*, "Solving Rubik's Cube with a Robot Hand," *arXiv*, Oct. 2019. DOI: 10.48550/arXiv.1910.07113.

[52] J. Mahler, M. Matl, V. Satish, *et al.*, "Learning Ambidextrous Robot Grasping Policies," *Science Robotics*, vol. 4, no. 26, Jan. 2019. DOI: 10.1126/scirobotics.aau4984.

[53] A. Moreau, N. Piasco, D. Tsishkou, B. Stanciulescu, and A. d. L. Fortelle, "LENS: Localization Enhanced by NeRF Synthesis," in *Proceedings of the 5th Conference on Robot Learning*, PMLR, Jan. 2022, pp. 1347–1356. [Online]. Available: https://proceedings.mlr.press/v164/moreau22a.html (visited on 09/10/2024).

[54] T. Gilles, S. Sabatini, D. Tsishkou, B. Stanciulescu, and F. Moutarde, "THOMAS: Trajectory Heatmap Output with Learned Multi-Agent Sampling," in *the 10th International Conference on Learning Representations (ICLR)*, Jan. 2022. DOI: 10.48550/arXiv.2110.06607.

[55] T. Gilles, S. Sabatini, D. Tsishkou, B. Stanciulescu, and F. Moutarde, "GOHOME: Graph-Oriented Heatmap Output for future Motion Estimation," in *2022 International Conference on Robotics and Automation (ICRA)*, May 2022, pp. 9107–9114, ISBN: 9781728196817. DOI: 10.1109/ICRA46639.2022.9812253.

[56] S. H. Tóth, Z. J. Viharos, Á. Bárdos, and Z. Szalay, "Sim-to-Real Application of Reinforcement Learning Agents for Autonomous, Real Vehicle Drifting," *Vehicles*, vol. 6, no. 2, pp. 781–798, Jun. 2024, ISSN: 2624-8921. DOI: 10.3390/vehicles6020037.

[57] R. Chekroun, H. Wang, J. Lee, *et al.*, "Mesoscale Traffic Forecasting for Real-Time Bottleneck and Shockwave Prediction," *arXiv*, Mar. 2024. DOI: 10.48550/arXiv.2402.05663.

[58] J. Botzheim, T. Obo, and N. Kubota, "Human Gesture Recognition for Robot Partners by Spiking Neural Network and Classification Learning," in *The 6th International Conference on Soft Computing and Intelligent Systems, and The 13th International Symposium on Advanced Intelligence Systems*, Nov. 2012, pp. 1954–1958. DOI: 10.1109/SCIS-ISIS.2012.6505305.

[59] J. Gesnouin, S. Pechberti, B. Stanciulcscu, and F. Moutarde, "TrouSPI-Net: Spatio-Temporal Attention on Parallel Atrous Convolutions and U-GRUs for Skeletal Pedestrian Crossing Prediction," in *the 16th IEEE International Conference on Automatic Face and Gesture Recognition*, Dec. 2021, pp. 01–07. DOI: 10.1109/FG52635.2021.9666989.

[60] L. Wang, R. Gao, J. Váncza, *et al.*, "Symbiotic Human-Robot Collaborative Assembly," *CIRP Annals*, vol. 68, no. 2, pp. 701–726, Jan. 2019, ISSN: 0007-8506. DOI: 10.1016/j.cirp.2019.05.002.

[61] B. Nagy and P. Korondi, "Deep Learning-Based Recognition and Analysis of Limb-Independent Dog Behavior for Ethorobotical Application," *IEEE Access*, vol. 10, pp. 3825–3834, Jan. 2022, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2022.3140513.

[62] F. Zorić, A. Milas, T. Petrović, Z. Kovačić, and M. Orsag, "AI-Enhanced Structural Health Monitoring with a Multi-Rotor Aerial Vehicle," in *2024 International Conference on Unmanned Aircraft Systems (ICUAS)*, Jun. 2024, pp. 1170–1176. DOI: 10.1109/ICUAS60882.2024.10556849.

[63] T. Selimović, M. Peti, and S. Bogdan, "Multi-Agent Active Perception Based on Reinforcement Learning and POMDP," *IEEE Access*, vol. 12, pp. 48 004–48 016, Apr. 2024, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2024.3383544.

[64] A. I. Károly, P. Galambos, J. Kuti, and I. J. Rudas, "Deep Learning in Robotics: Survey on Model Structures and Training Strategies," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 1, pp. 266–279, Jan. 2021, ISSN: 2168-2232. DOI: 10.1109/TSMC.2020.3018325.

[65] Y. Bao, Y. Li, S.-L. Huang, *et al.*, "An Information-Theoretic Approach to Transferability in Task Transfer Learning," in *2019 IEEE International Conference on Image Processing (ICIP)*, Sep. 2019, pp. 2309–2313. DOI: 10.1109/ICIP.2019.8803726.

[66] S. Zhou, M. K. Helwa, A. P. Schoellig, A. Sarabakha, and E. Kayacan, "Knowledge Transfer Between Robots with Similar Dynamics for High-Accuracy Impromptu Trajectory Tracking," in *2019 18th European Control Conference (ECC)*, Jun. 2019, pp. 1–8. DOI: 10.23919/ECC.2019.8796140.

[67] E. Salvato, G. Fenu, E. Medvet, and F. A. Pellegrino, "Crossing the Reality Gap: A Survey on Sim-to-Real Transferability of Robot Controllers in Reinforcement Learning," *IEEE Access*, vol. 9, pp. 153 171–153 187, Nov. 2021, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2021.3126658.

[68] S. J. Pan and Q. Yang, "A Survey on Transfer Learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010, ISSN: 1558-2191. DOI: 10.1109/TKDE.2009.191.

[69] K. Weiss, T. M. Khoshgoftaar, and D. Wang, "A Survey of Transfer Learning," *Journal of Big Data*, vol. 3, no. 1, p. 9, May 2016, ISSN: 2196-1115. DOI: 10.1186/s40537-016-0043-6.

[70] J. Deng, W. Dong, R. Socher, *et al.*, "ImageNet: A Large-Scale Hierarchical Image Database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[71] T.-Y. Lin, M. Maire, S. Belongie, *et al.*, "Microsoft COCO: Common Objects in Context," *arXiv*, Feb. 2015. DOI: 10.48550/arXiv.1405.0312.

[72] S. Dasari, F. Ebert, S. Tian, *et al.*, "RoboNet: Large-Scale Multi-Robot Learning," *arXiv*, Jan. 2020. DOI: 10.48550/arXiv.1910.11215.

[73] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. A Bradford Book, 2018, ISBN: 9780262039246.

[74] M. Naeem, S. T. H. Rizvi, and A. Coronato, "A Gentle Introduction to Reinforcement Learning and its Application in Different Fields," *IEEE Access*, vol. 8, pp. 209 320–209 344, Nov. 2020, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.3038605.

[75] M. Q. Mohammed, K. L. Chung, and C. S. Chyi, "Review of Deep Reinforcement Learning-Based Object Grasping: Techniques, Open Challenges, and Recommendations," *IEEE Access*, vol. 8, pp. 178 450–178 481, Sep. 2020, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.3027923.

[76] Y. LeCun, "A Path Towards Autonomous Machine Intelligence," version 0.9.2,, Jun. 2022. [Online]. Available: https://openreview.net/pdf?id=BZ5a1r-kVsf (visited on 09/10/2024).

[77] D. Silver, T. Hubert, J. Schrittwieser, *et al.*, "Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm," *arXiv*, Dec. 2017. DOI: 10.48550/arXiv.1712.01815.

[78] D. Silver, J. Schrittwieser, K. Simonyan, *et al.*, "Mastering the Game of Go without Human Knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, Oct. 2017, ISSN: 1476-4687. DOI: 10.1038/nature24270.

[79] V. Mnih, K. Kavukcuoglu, D. Silver, *et al.*, "Human-Level Control Through Deep Reinforcement Learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015, ISSN: 1476-4687. DOI: 10.1038/nature14236.

[80] D. Silver, G. Lever, N. Heess, *et al.*, "Deterministic Policy Gradient Algorithms," in *Proceedings of the 31st International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 32, PMLR, Jun. 2014, pp. 387–395. [Online]. Available: https://proceedings.mlr.press/v32/silver14.html (visited on 09/10/2024).

[81] T. P. Lillicrap, J. J. Hunt, A. Pritzel, *et al.*, *Continuous Control with Deep Reinforcement Learning*, Jul. 2019. DOI: 10.48550/arXiv.1509.02971.

[82] S. Fujimoto, H. van Hoof, and D. Meger, "Addressing Function Approximation Error in Actor-Critic Methods," *arXiv*, Oct. 2018. DOI: 10.48550/arXiv.1802.09477.

[83] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor," *arXiv*, Aug. 2018. DOI: 10.48550/arXiv.1801.01290.

[84] M. Andrychowicz, F. Wolski, A. Ray, *et al.*, "Hindsight Experience Replay," in *Advances in Neural Information Processing Systems*, vol. 30, Curran Associates, Inc., Jul. 2017. DOI: 10.48550/arXiv.1707.01495.

[85] J. Bujalance and F. Moutarde, "Reward Relabelling for Combined Reinforcement and Imitation Learning on Sparse-Reward Tasks," in *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, Feb. 2023, pp. 2565–2567. DOI: 10.48550/arXiv.2201.03834.

[86] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized Experience Replay," *arXiv*, Feb. 2016. DOI: 10.48550/arXiv.1511.05952.

[87] J. Oh, Y. Guo, S. Singh, and H. Lee, "Self-Imitation Learning," *arXiv*, Jun. 2018. DOI: 10.48550/arXiv.1806.05635.

[88] C. Wang and K. Ross, "Boosting Soft Actor-Critic: Emphasizing Recent Experience without Forgetting the Past," *arXiv*, Jun. 2019. DOI: 10.48550/arXiv.1906.04009.

[89] C. Florensa, D. Held, M. Wulfmeier, M. Zhang, and P. Abbeel, "Reverse Curriculum Generation for Reinforcement Learning," *arXiv*, Jul. 2018. DOI: 10.48550/arXiv.1707.05300.

[90] B. Ivanovic, J. Harrison, A. Sharma, M. Chen, and M. Pavone, "BaRC: Backward Reachability Curriculum for Robotic Reinforcement Learning," in *2019 International Conference on Robotics and Automation (ICRA)*, May 2019, pp. 15–21. DOI: 10.1109/ICRA.2019.8794206.

[91] T. Salimans and R. Chen, "Learning Montezuma's Revenge from a Single Demonstration," *arXiv*, Dec. 2018. DOI: 10.48550/arXiv.1812.03381.

[92] S. Sukhbaatar, Z. Lin, I. Kostrikov, *et al.*, "Intrinsic Motivation and Automatic Curricula via Asymmetric Self-Play," *arXiv*, Apr. 2018. DOI: 10.48550/arXiv.1703.05407.

[93] C. Florensa, D. Held, X. Geng, and P. Abbeel, "Automatic Goal Generation for Reinforcement Learning Agents," *arXiv*, Jul. 2018. DOI: 10.48550/arXiv.1705.06366.

[94] V. H. Pong, M. Dalal, S. Lin, *et al.*, "Skew-Fit: State-Covering Self-Supervised Reinforcement Learning," *arXiv*, Aug. 2020. DOI: 10.48550/arXiv.1903.03698.

[95] S. Racaniere, A. K. Lampinen, A. Santoro, *et al.*, "Automated Curricula Through Setter-Solver Interactions," *arXiv*, Jan. 2020. DOI: 10.48550/arXiv.1909.12892.

[96] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum Learning," in *Proceedings of the 26th Annual International Conference on Machine Learning (ICML)*, Association for Computing Machinery, Jun. 2009, pp. 41–48, ISBN: 9781605585161. DOI: 10.1145/1553374.1553380.

[97] R. Portelas, C. Colas, L. Weng, K. Hofmann, and P.-Y. Oudeyer, "Automatic Curriculum Learning For Deep RL: A Short Survey," *arXiv*, May 2020. DOI: 10.48550/arXiv.2003.04664.

[98] X. Wang, Y. Chen, and W. Zhu, "A Survey on Curriculum Learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 9, pp. 4555–4576, Sep. 2022, ISSN: 1939-3539. DOI: 10.1109/TPAMI.2021.3069908.

[99] A. Yahya, A. Li, M. Kalakrishnan, Y. Chebotar, and S. Levine, "Collective Robot Reinforcement Learning with Distributed Asynchronous Guided Policy Search," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2017, pp. 79–86. DOI: 10.1109/IROS.2017.8202141.

[100] H. Zhu, A. Gupta, A. Rajeswaran, S. Levine, and V. Kumar, "Dexterous Manipulation with Deep Reinforcement Learning: Efficient, General, and Low-Cost," in *2019 International Conference on Robotics and Automation (ICRA)*, May 2019, pp. 3651–3657. DOI: 10.1109/ICRA.2019.8794102.

[101] A. Nagabandi, K. Konolige, S. Levine, and V. Kumar, "Deep Dynamics Models for Learning Dexterous Manipulation," in *Proceedings of the Conference on Robot Learning*, PMLR, May 2020, pp. 1101–1112. [Online]. Available: https://proceedings.mlr.press/v100/nagabandi20a.html (visited on 09/10/2024).

[102] H. Zhu, J. Yu, A. Gupta, *et al.*, "The Ingredients of Real-World Robotic Reinforcement Learning," *arXiv*, Apr. 2020. DOI: 10.48550/arXiv.2004.12570.

[103] R. Szeliski, *Computer Vision: Algorithms and Applications* (Texts in Computer Science). Springer International Publishing, 2022, ISBN: 9783030343712. DOI: 10.1007/978-3-030-34372-9.

[104] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, 2004, ISBN: 9780521540513.

[105] A. Ouaknine, *Review of Deep Learning Algorithms for Object Detection*, Medium, Feb. 2018. [Online]. Available: https://medium.com/zylapp/review-of-deep-learning-algorithms-for-object-detection-c1f3d437b852 (visited on 09/10/2024).

[106] Y. LeCun, B. Boser, J. S. Denker, *et al.*, "Backpropagation Applied to Handwritten Zip Code Recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, Dec. 1989, ISSN: 0899-7667. DOI: 10.1162/neco.1989.1.4.541.

[107] O. Russakovsky, J. Deng, H. Su, *et al.*, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, Dec. 2015, ISSN: 1573-1405. DOI: 10.1007/s11263-015-0816-y.

[108] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. The MIT Press, 2016, ISBN: 9780262035613.

[109] Zhou and Chellappa, "Computation of Optical Flow Using a Neural Network," in *IEEE 1988 International Conference on Neural Networks*, Jul. 1988, 71–78 vol.2. DOI: 10.1109/ICNN.1988.23914.

[110] Y.-L. Boureau, J. Ponce, and Y. LeCun, "A Theoretical Analysis of Feature Pooling in Visual Recognition," in *Proceedings of the 27th International Conference on International Conference on Machine Learning (ICML)*, Omnipress, Jun. 2010, pp. 111–118, ISBN: 9781605589077. [Online]. Available: https://dl.acm.org/doi/10.5555/3104322.3104338 (visited on 09/10/2024).

[111] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv*, Apr. 2015. DOI: 10.48550/arXiv.1409.1556.

[112] J. Jordan, *Common Architectures in Convolutional Neural Networks*. Apr. 2018. [Online]. Available: https://www.jeremyjordan.me/convnet-architectures/ (visited on 09/24/2024).

[113] A. Dosovitskiy, L. Beyer, A. Kolesnikov, *et al.*, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," *arXiv*, Jun. 2021. DOI: 10.48550/arXiv.2010.11929.

[114] Y. Liu, Y. Zhang, Y. Wang, *et al.*, "A Survey of Visual Transformers," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 35, no. 6, pp. 7478–7498, Jun. 2024, ISSN: 2162-2388. DOI: 10.1109/TNNLS.2022.3227717.

[115] H. Rezatofighi, N. Tsoi, J. Gwak, *et al.*, "Generalized Intersection Over Union: A Metric and a Loss for Bounding Box Regression," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2019, pp. 658–666, ISBN: 9781728132938. DOI: 10.1109/CVPR.2019.00075.

[116] Biggerj1, *PR Curve with Optimal F-score*, Wikipedia, 2023. [Online]. Available: https://en.m.wikipedia.org/wiki/File:PR_curve_with_optimal_fscore.png (visited on 09/10/2024).

[117] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2014, pp. 580–587. DOI: 10.1109/CVPR.2014.81.

[118] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition," in *the 13th European Conference on Computer Vision (ECCV)*, ser. Lecture Notes in Computer Science, Springer International Publishing, 2014, pp. 346–361, ISBN: 9783319105789. DOI: 10.1007/978-3-319-10578-9_23.

[119] R. Girshick, "Fast R-CNN," in *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec. 2015, pp. 1440–1448. DOI: 10.1109/ICCV.2015.169.

[120] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017, ISSN: 0162-8828, 2160-9292. DOI: 10.1109/TPAMI.2016.2577031.

[121] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2016, pp. 779–788, ISBN: 9781467388511. DOI: 10.1109/CVPR.2016.91.

[122] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jul. 2017, pp. 6517–6525, ISBN: 9781538604571. DOI: 10.1109/CVPR.2017.690.

[123] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," *arXiv*, Apr. 2018. DOI: 10.48550/arXiv.1804.02767.

[124] W. Liu, D. Anguelov, D. Erhan, *et al.*, "SSD: Single Shot MultiBox Detector," in *the 14th European Conference on Computer Vision (ECCV)*, Springer International Publishing, 2016, pp. 21–37, ISBN: 9783319464480. DOI: 10.1007/978-3-319-46448-0_2.

[125] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal Loss for Dense Object Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 42, no. 2, pp. 318–327, Feb. 2020, ISSN: 1939-3539. DOI: 10.1109/TPAMI.2018.2858826.

[126] Ultralytics, *Documentation of YOLOv5*. [Online]. Available: https://docs.ultralytics.com/yolov5 (visited on 09/10/2024).

[127] C. Li, L. Li, H. Jiang, *et al.*, "YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications," *arXiv*, Sep. 2022. DOI: 10.48550/arXiv.2209.02976.

[128] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors," *arXiv*, Jul. 2022. DOI: 10.48550/arXiv.2207.02696.

[129] Ultralytics, *Documentation of YOLOv8*. [Online]. Available: https://docs.ultralytics.com/models/yolov8 (visited on 09/10/2024).

[130] C.-Y. Wang, I.-H. Yeh, and H.-Y. M. Liao, "YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information," *arXiv*, Feb. 2024. DOI: 10.48550/arXiv.2402.13616.

[131] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90.

[132] Z. Zou, Z. Shi, Y. Guo, and J. Ye, "Object Detection in 20 Years: A Survey," *arXiv*, May 2019. DOI: 10.48550/arXiv.1905.05055.

[133] F. Zhuang, Z. Qi, K. Duan, *et al.*, "A Comprehensive Survey on Transfer Learning," *Proceedings of the IEEE*, vol. 109, no. 1, pp. 43–76, Jan. 2021, ISSN: 1558-2256. DOI: 10.1109/JPROC.2020.3004555.

[134] O. Day and T. M. Khoshgoftaar, "A Survey on Heterogeneous Transfer Learning," *Journal of Big Data*, vol. 4, no. 1, p. 29, Sep. 2017, ISSN: 2196-1115. DOI: 10.1186/s40537-017-0089-0.

[135] M. Sugiyama, T. Suzuki, S. Nakajima, *et al.*, "Direct Importance Estimation for Covariate Shift Adaptation," *Annals of the Institute of Statistical Mathematics*, vol. 60, no. 4, pp. 699–746, Dec. 2008, ISSN: 1572-9052. DOI: 10.1007/s10463-008-0197-x.

[136] Z. Wang, Z. Dai, B. Poczos, and J. Carbonell, "Characterizing and Avoiding Negative Transfer," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2019, pp. 11 285–11 294, ISBN: 9781728132938. DOI: 10.1109/CVPR.2019.01155.

[137] B. O. Community, *Blender - a 3D Modelling and Rendering Package*, Blender Foundation, 2018. [Online]. Available: http://www.blender.org (visited on 09/10/2024).

[138] D. Kingma and M. Welling, "Auto-Encoding Variational Bayes," in *the 2nd International Conference on Learning Representations (ICLR)*, May 2014. DOI: 10.48550/arXiv.1312.6114.

[139] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, *et al.*, "Generative Adversarial Networks," *arXiv*, Jun. 2014. DOI: 10.48550/arXiv.1406.2661.

[140] N. Jakobi, "Evolutionary Robotics and the Radical Envelope-of-Noise Hypothesis," *Adaptive Behavior*, vol. 6, no. 2, pp. 325–368, Sep. 1997, ISSN: 1059-7123. DOI: 10.1177/105971239700600205.

[141] T. Schaul, D. Horgan, K. Gregor, and D. Silver, "Universal Value Function Approximators," in *Proceedings of the 32nd International Conference on Machine Learning*, vol. 37, PMLR, Jul. 2015, pp. 1312–1320. [Online]. Available: https://proceedings.mlr.press/v37/schaul15.html (visited on 09/10/2024).

[142] J. Ramírez, W. Yu, and A. Perrusquía, "Model-Free Reinforcement Learning from Expert Demonstrations: A Survey," *Artificial Intelligence Review*, vol. 55, no. 4, pp. 3213–3241, Apr. 2022, ISSN: 1573-7462. DOI: 10.1007/s10462-021-10085-1.

[143] R. E. Bellman, *Dynamic Programming*. Princeton University Press, 1957, ISBN: 9780691079516.

[144] R. S. Sutton, "Integrated Architectures for Learning, Planning, and Reacting Based on Approximating Dynamic Programming," in *Machine Learning Proceedings 1990*, Morgan Kaufmann, Jan. 1990, pp. 216–224, ISBN: 9781558601413. DOI: 10.1016/B978-1-55860-141-3.50030-4.

[145] C. Watkins, "Learning From Delayed Rewards," *Ph.D. thesis, University of Cambridge.*, Jan. 1989.

[146] G. Rummery and M. Niranjan, "On-Line Q-Learning Using Connectionist Systems," *Technical Report CUED/F-INFENG/TR 166*, Nov. 1994.

[147] R. J. Williams, "Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning," *Machine Learning*, vol. 8, no. 3, pp. 229–256, May 1992, ISSN: 1573-0565. DOI: 10.1007/BF00992696.

[148] V. Mnih, A. P. Badia, M. Mirza, *et al.*, "Asynchronous Methods for Deep Reinforcement Learning," in *Proceedings of The 33rd International Conference on Machine Learning*, PMLR, Jun. 2016, pp. 1928–1937. [Online]. Available: https://proceedings.mlr.press/v48/mniha16.html (visited on 09/10/2024).

[149] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," *arXiv*, Aug. 2017. DOI: 10.48550/arXiv.1707.06347.

[150] P. Auer, "Using Confidence Bounds for Exploitation-Exploration Trade-offs," *Journal of Machine Learning Research*, vol. 3, no. Nov, pp. 397–422, Nov. 2002, ISSN: 1533-7928. [Online]. Available: https://www.jmlr.org/papers/v3/auer02a.html (visited on 09/10/2024).

[151] R. S. Sutton, "Generalization in Reinforcement Learning: Successful Examples Using Sparse Coarse Coding," in *Advances in Neural Information Processing Systems*, vol. 8, MIT Press, 1995. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/1995/hash/8f1d43620bc6bb580df6e80b0dc05c48-Abstract.html (visited on 09/10/2024).

[152] Z. I. Botev, D. P. Kroese, R. Y. Rubinstein, and P. L'Ecuyer, *Chapter 3 - The Cross-Entropy Method for Optimization* (Handbook of Statistics). Elsevier, Jan. 2013, vol. 31, pp. 35–59. DOI: 10.1016/B978-0-444-53859-8.00003-5.

[153] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, "Trust Region Policy Optimization," *arXiv*, Apr. 2017. DOI: 10.48550/arXiv.1502.05477.

[154] Y. Wu, E. Mansimov, S. Liao, R. Grosse, and J. Ba, "Scalable Trust-Region Method for Deep Reinforcement Learning Using Kronecker-Factored Approximation," *arXiv*, Aug. 2017. DOI: 10.48550/arXiv.1708.05144.

[155] S. C. Y. Chan, S. Fishman, A. Korattikara, J. Canny, and S. Guadarrama, "Measuring the Reliability of Reinforcement Learning Algorithms," in *the 8th International Conference on Learning Representations (ICLR)*, Apr. 2020. [Online]. Available: https://iclr.cc/virtual_2020/poster_SJlpYJBKvH.html (visited on 09/10/2024).

[156] C. Colas, O. Sigaud, and P.-Y. Oudeyer, "How Many Random Seeds? Statistical Power Analysis in Deep Reinforcement Learning Experiments," *arXiv*, Jul. 2018. DOI: 10.48550/arXiv.1806.08295.

[157] P. Henderson, R. Islam, P. Bachman, *et al.*, "Deep Reinforcement Learning That Matters," 1, vol. 32, Apr. 2018, pp. 3207–3214. DOI: 10.1609/aaai.v32i1.11694.

[158] R. Agarwal, M. Schwarzer, P. S. Castro, A. C. Courville, and M. Bellemare, "Deep Reinforcement Learning at the Edge of the Statistical Precipice," in *Advances in Neural Information Processing Systems*, vol. 34, Curran Associates, Inc., 2021, pp. 29 304–29 320. [Online]. Available: https://proceedings.neurips.cc/paper/2021/hash/f514cec81cb148559cf475e7426eed5e-Abstract.html (visited on 09/10/2024).

[159] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The Arcade Learning Environment: An Evaluation Platform for General Agents," *Journal of Artificial Intelligence Research*, vol. 47, pp. 253–279, Jun. 2013, ISSN: 1076-9757. DOI: 10.1613/jair.3912.

[160] O. G. Selfridge, R. S. Sutton, and A. G. Barto, "Training and Tracking in Robotics," in *Proceedings of the 9th international joint conference on Artificial intelligence*, ser. IJCAI'85, Morgan Kaufmann Publishers Inc., Aug. 1985, pp. 670–672, ISBN: 9780934613026. [Online]. Available: https://dl.acm.org/doi/10.5555/1625135.1625265 (visited on 09/10/2024).

[161] E. L. Allgower and K. Georg, *Numerical Continuation Methods: An Introduction*. Springer Science and Business Media, Dec. 2012, ISBN: 9783642612572.

[162] T. Gong, Q. Zhao, D. Meng, and Z. Xu, "Why Curriculum Learning and Self-Paced Learning Work in Big/Noisy Data: A Theoretical Perspective," *Big Data and Information Analytics*, vol. 1, no. 1, pp. 111–127, Sep. 2015, ISSN: 2380-6966. DOI: 10.3934/bdia.2016.1.111.

[163] Y. Bengio, "Evolving Culture Versus Local Minima," in *Growing Adaptive Machines: Combining Development and Learning in Artificial Neural Networks*, Springer, 2014, pp. 109–138, ISBN: 9783642553370. DOI: 10.1007/978-3-642-55337-0_3.

[164] Y. Fan, F. Tian, T. Qin, X.-Y. Li, and T.-Y. Liu, "Learning to Teach," *arXiv*, May 2018. DOI: 10.48550/arXiv.1805.03643.

[165] W. Wang, I. Caswell, and C. Chelba, "Dynamically Composing Domain-Data Selection with Clean-Data Selection by "Co-Curricular Learning" for Neural Machine Translation," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, Jul. 2019, pp. 1282–1292. DOI: 10.18653/v1/P19-1123.

[166] S. Jin, A. RoyChowdhury, H. Jiang, *et al.*, "Unsupervised Hard Example Mining from Videos for Improved Object Detection," in *the 15th European Conference on Computer Vision (ECCV)*, Springer-Verlag, Sep. 2018, pp. 316–333, ISBN: 9783030012601. DOI: 10.1007/978-3-030-01261-8_19.

[167] A. Shrivastava, A. Gupta, and R. Girshick, "Training Region-Based Object Detectors with Online Hard Example Mining," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, Jun. 2016, pp. 761–769, ISBN: 9781467388511. DOI: 10.1109/CVPR.2016.89.

[168] G. Hacohen and D. Weinshall, "On The Power of Curriculum Learning in Training Deep Networks," in *Proceedings of the 36th International Conference on Machine Learning*, PMLR, May 2019, pp. 2535–2544. [Online]. Available: https://proceedings.mlr.press/v97/hacohen19a.html (visited on 09/10/2024).

[169] E. A. Platanios, O. Stretcu, G. Neubig, B. Poczos, and T. Mitchell, "Competence-Based Curriculum Learning for Neural Machine Translation," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Association for Computational Linguistics, Jun. 2019, pp. 1162–1172. DOI: 10.18653/v1/N19-1119.

[170] V. I. Spitkovsky, H. Alshawi, and D. Jurafsky, "From Baby Steps to Leapfrog: How "Less is More" in Unsupervised Dependency Parsing," in *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, Association for Computational Linguistics, Jun. 2010, pp. 751–759. [Online]. Available: https://aclanthology.org/N10-1116 (visited on 09/10/2024).

[171] Y. Tay, S. Wang, A. T. Luu, *et al.*, "Simple and Effective Curriculum Pointer-Generator Networks for Reading Comprehension over Long Narratives," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, Jul. 2019, pp. 4922–4931. DOI: 10.18653/v1/P19-1486.

[172] M. Kumar, B. Packer, and D. Koller, "Self-Paced Learning for Latent Variable Models," in *Advances in Neural Information Processing Systems*, vol. 23, Curran Associates, Inc., 2010. [Online]. Available: https://papers.nips.cc/paper_files/paper/2010/hash/e57c6b956a6521b28495f2886ca0977a-Abstract.html (visited on 09/10/2024).

[173] C. Li, F. Wei, J. Yan, *et al.*, "A Self-Paced Regularization Framework for Multilabel Learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 6, pp. 2660–2666, Jun. 2018, ISSN: 2162-2388. DOI: 10.1109/TNNLS.2017.2697767.

[174] D. Weinshall, G. Cohen, and D. Amir, "Curriculum Learning by Transfer Learning: Theory and Experiments with Deep Networks," in *Proceedings of the 35th International Conference on Machine Learning*, PMLR, Jul. 2018, pp. 5238–5246. [Online]. Available: https://proceedings.mlr.press/v80/weinshall18a.html (visited on 09/10/2024).

[175] T. Matiisen, A. Oliver, T. Cohen, and J. Schulman, "Teacher–Student Curriculum Learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 9, pp. 3732–3740, Sep. 2020, ISSN: 2162-2388. DOI: 10.1109/TNNLS.2019.2934906.

[176] Y. Tsvetkov, M. Faruqui, W. Ling, B. MacWhinney, and C. Dyer, "Learning the Curriculum with Bayesian Optimization for Task-Specific Word Representation Learning," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, Aug. 2016, pp. 130–139. DOI: 10.18653/v1/P16-1013.

[177] D. Zhang, H. Tian, and J. Han, "Few-Cost Salient Object Detection with Adversarial-Paced Learning," in *Advances in Neural Information Processing Systems*, vol. 33, Curran Associates, Inc., 2020, pp. 12 236–12 247. [Online]. Available: https://proceedings.neurips.cc/paper/2020/hash/8fc687aa152e8199fe9e73304d407bca-Abstract.html (visited on 09/10/2024).

[178] M. Bellemare, S. Srinivasan, G. Ostrovski, *et al.*, "Unifying Count-Based Exploration and Intrinsic Motivation," in *Advances in Neural Information Processing Systems*, vol. 29, Curran Associates, Inc., 2016. [Online]. Available: https://papers.nips.cc/paper_files/paper/2016/hash/afda332245e2af431fb7b672a68b659d-Abstract.html (visited on 09/10/2024).

[179] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-Driven Exploration by Self-Supervised Prediction," in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Jul. 2017, pp. 488–489. DOI: 10.1109/CVPRW.2017.70.

[180] S. Risi and J. Togelius, "Increasing Generality in Machine Learning through Procedural Content Generation," *arXiv*, Mar. 2020. DOI: 10.48550/arXiv.1911.13071.

[181] T. Bansal, J. Pachocki, S. Sidor, I. Sutskever, and I. Mordatch, "Emergent Complexity via Multi-Agent Competition," *arXiv*, Mar. 2018. DOI: 10.48550/arXiv.1710.03748.

[182] A. Zeng, S. Song, J. Lee, A. Rodriguez, and T. Funkhouser, "TossingBot: Learning to Throw Arbitrary Objects With Residual Physics," *IEEE Transactions on Robotics*, vol. 36, no. 4, pp. 1307–1319, Aug. 2020, ISSN: 1941-0468. DOI: 10.1109/TRO.2020.2988642.

[183] I. Alonso, L. Riazuelo, and A. C. Murillo, "MiniNet: An Efficient Semantic Segmentation ConvNet for Real-Time Robotic Applications," *IEEE Transactions on Robotics*, vol. 36, no. 4, pp. 1340–1347, Aug. 2020, ISSN: 1941-0468. DOI: 10.1109/TRO.2020.2974099.

[184] A. Pashevich, R. Strudel, I. Kalevatykh, I. Laptev, and C. Schmid, "Learning to Augment Synthetic Images for Sim2Real Policy Transfer," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov. 2019, pp. 2651–2657. DOI: 10.1109/IROS40897.2019.8967622.

[185] K. Bousmalis, A. Irpan, P. Wohlhart, *et al.*, "Using Simulation and Domain Adaptation to Improve Efficiency of Deep Robotic Grasping," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, May 2018, pp. 4243–4250. DOI: 10.1109/ICRA.2018.8460875.

[186] J. Tobin, R. Fong, A. Ray, *et al.*, "Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2017, pp. 23–30. DOI: 10.1109/IROS.2017.8202133.

[187] A. Dehban, J. Borrego, R. Figueiredo, *et al.*, "The Impact of Domain Randomization on Object Detection: A Case Study on Parametric Shapes and Synthetic Textures," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov. 2019, pp. 2593–2600. DOI: 10.1109/IROS40897.2019.8968139.

[188] E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A Physics Engine for Model-Based Control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2012, pp. 5026–5033. DOI: 10.1109/IROS.2012.6386109.

[189] J. Borrego, R. Figueiredo, A. Dehban, *et al.*, "A Generic Visual Perception Domain Randomisation Framework for Gazebo," in *2018 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, Apr. 2018, pp. 237–242, ISBN: 9781538652213. DOI: 10.1109/ICARSC.2018.8374189.

[190] N. Koenig and A. Howard, "Design and Use Paradigms for Gazebo, an Open-Source Multi-Robot Simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, Sep. 2004, 2149–2154 vol.3. DOI: 10.1109/IROS.2004.1389727.

[191] K. Perlin, "Improving Noise," *ACM Transactions on Graphics*, vol. 21, no. 3, pp. 681–682, Jul. 2002, ISSN: 0730-0301. DOI: 10.1145/566654.566636.

[192] E. Coumans and Y. Bai, *PyBullet, a Python Module for Physics Simulation for Games, Robotics and Machine Learning*. [Online]. Available: http://pybullet.org (visited on 09/10/2024).

[193] R. Coulom, "Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search," in *Computers and Games*, ser. Lecture Notes in Computer Science, Springer, 2007, pp. 72–83, ISBN: 9783540755388. DOI: 10.1007/978-3-540-75538-8_7.

[194] T. DeVries and G. W. Taylor, "Improved Regularization of Convolutional Neural Networks with Cutout," *arXiv*, Nov. 2017. DOI: 10.48550/arXiv.1708.04552.

[195] S. James, A. J. Davison, and E. Johns, "Transferring End-to-End Visuomotor Control from Simulation to Real World for a Multi-Stage Task," *arXiv*, Oct. 2017. DOI: 10.48550/arXiv.1707.02267.

[196] A. Devo, G. Mezzetti, G. Costante, M. L. Fravolini, and P. Valigi, "Towards Generalization in Target-Driven Visual Navigation by Using Deep Reinforcement Learning," *IEEE Transactions on Robotics*, vol. 36, no. 5, pp. 1546–1561, Oct. 2020, ISSN: 1941-0468. DOI: 10.1109/TRO.2020.2994002.

[197] Epic Games, *Unreal engine*, version 4.22.1. [Online]. Available: https://www.unrealengine.com (visited on 09/10/2024).

[198] Y. Chen, W. Li, C. Sakaridis, D. Dai, and L. Van Gool, "Domain Adaptive Faster R-CNN for Object Detection in the Wild," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2018, pp. 3339–3348. DOI: 10.1109/CVPR.2018.00352.

[199] M. Johnson-Roberson, C. Barto, R. Mehta, *et al.*, "Driving in the Matrix: Can Virtual Worlds Replace Human-Generated Annotations for Real World Tasks?" In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 746–753, ISBN: 9781509046331. DOI: 10.1109/ICRA.2017.7989092.

[200] C. Sakaridis, D. Dai, and L. Van Gool, "Semantic Foggy Scene Understanding with Synthetic Data," *International Journal of Computer Vision*, vol. 126, no. 9, pp. 973–992, Sep. 2018, ISSN: 1573-1405. DOI: 10.1007/s11263-018-1072-8.

[201] S. Sankaranarayanan, Y. Balaji, A. Jain, S. N. Lim, and R. Chellappa, "Learning from Synthetic Data: Addressing Domain Shift for Semantic Segmentation," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2018, pp. 3752–3761. DOI: 10.1109/CVPR.2018.00395.

[202] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez, "The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 3234–3243. DOI: 10.1109/CVPR.2016.352.

[203] S. R. Richter, V. Vineet, S. Roth, and V. Koltun, "Playing for Data: Ground Truth from Computer Games," in *the 14th European Conference on Computer Vision (ECCV)*, ser. Lecture Notes in Computer Science, Springer International Publishing, 2016, pp. 102–118, ISBN: 9783319464756. DOI: 10.1007/978-3-319-46475-6_7.

[204] J. Tremblay, A. Prakash, D. Acuna, *et al.*, "Training Deep Networks with Synthetic Data: Bridging the Reality Gap by Domain Randomization," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Jun. 2018, pp. 1082–10 828. DOI: 10.1109/CVPRW.2018.00143.

[205] A. Geiger, P. Lenz, and R. Urtasun, "Are We Ready for Autonomous Driving? The KITTI Vision Benchmark Suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2012, pp. 3354–3361. DOI: 10.1109/CVPR.2012.6248074.

[206] J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: Object Detection via Region-Based Fully Convolutional Networks," in *Advances in Neural Information Processing Systems*, vol. 29, Curran Associates, Inc., 2016. [Online]. Available: https://papers.nips.cc/paper_files/paper/2016/hash/577ef1154f3240ad5b9b413aa7346a1e-Abstract.html (visited on 09/10/2024).

[207] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig, "VirtualWorlds as Proxy for Multi-Object Tracking Analysis," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 4340–4349. DOI: 10.1109/CVPR.2016.470.

[208] *Poly Haven*. [Online]. Available: https://polyhaven.com/ (visited on 09/10/2024).

[209] S. Hinterstoisser, V. Lepetit, P. Wohlhart, and K. Konolige, "On Pre-Trained Image Features and Synthetic Images for Deep Learning," *arXiv*, Nov. 2017. DOI: 10.48550/arXiv.1710.10710.

[210] B. T. Phong, "Illumination for Computer Generated Pictures," *Communications of the ACM*, vol. 18, no. 6, pp. 311–317, Jun. 1975, ISSN: 0001-0782. DOI: 10.1145/360825.360839.

[211] F. Zhang, J. Leitner, Z. Ge, M. Milford, and P. Corke, "Adversarial Discriminative Sim-to-Real Transfer of Visuo-Motor Policies," *The International Journal of Robotics Research*, vol. 38, no. 10-11, pp. 1229–1245, Sep. 2019, ISSN: 0278-3649. DOI: 10.1177/0278364919870227.

[212] H. M. Clever, P. Grady, G. Turk, and C. C. Kemp, "BodyPressure – Inferring Body Pose and Contact Pressure from a Depth Image," *arXiv*, May 2021. DOI: 10.48550/arXiv.2105.09936.

[213] H. M. Clever, Z. Erickson, A. Kapusta, *et al.*, "Bodies at Rest: 3D Human Pose and Shape Estimation From a Pressure Image Using Synthetic Data," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2020, pp. 6214–6223. DOI: 10.1109/CVPR42600.2020.00625.

[214] S. Liu, X. Huang, N. Fu, *et al.*, "Simultaneously-Collected Multimodal Lying Pose Dataset: Towards In-Bed Human Pose Monitoring under Adverse Vision Conditions," *arXiv*, Aug. 2020. DOI: 10.48550/arXiv.2008.08735.

[215] D. F. Gomes, P. Paoletti, and S. Luo, "Generation of GelSight Tactile Images for Sim2Real Learning," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 4177–4184, Apr. 2021, ISSN: 2377-3766. DOI: 10.1109/LRA.2021.3063925.

[216] E. Rohmer, S. P. N. Singh, and M. Freese, "V-REP: A Versatile and Scalable Robot Simulation Framework," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nov. 2013, pp. 1321–1326. DOI: 10.1109/IROS.2013.6696520.

[217] *OpenGL*. [Online]. Available: https://www.opengl.org/ (visited on 09/10/2024).

[218] J. Lee, M. Grey, S. Ha, *et al.*, "DART: Dynamic Animation and Robotics Toolkit," *The Journal of Open Source Software*, vol. 3, p. 500, Feb. 2018. DOI: 10.21105/joss.00500.

[219] M. Macklin, M. Müller, N. Chentanez, and T.-Y. Kim, "Unified Particle Physics for Real-Time Applications," *ACM Transactions on Graphics*, vol. 33, no. 4, 153:1–153:12, Jul. 2014, ISSN: 0730-0301. DOI: 10.1145/2601097.2601152.

[220] M. Matl, *Pyrender*. [Online]. Available: https://github.com/mmatl/pyrender (visited on 09/10/2024).

[221] B. Calli, A. Singh, A. Walsman, *et al.*, "The YCB Object and Model Set: Towards Common Benchmarks for Manipulation Research," in *2015 International Conference on Advanced Robotics (ICAR)*, Jul. 2015, pp. 510–517. DOI: 10.1109/ICAR.2015.7251504.

[222] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, and A. Vedaldi, "Describing Textures in the Wild," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2014, pp. 3606–3613. DOI: 10.1109/CVPR.2014.461.

[223] *Free Stock Textures*. [Online]. Available: https://freestocktextures.com/ (visited on 09/10/2024).

[224] *Texture Ninja*. [Online]. Available: http://textureninja.com (visited on 09/10/2024).

[225] E. Angel and D. Shreiner, *Interactive Computer Graphics: A Top-Down Approach with WebGL*, 7th ed. Pearson, Feb. 2014, ISBN: 9780133574845.

[226] *Multiclass Confusion Matrix for Object Detection*, May 2023. [Online]. Available: https://www.tenyks.ai/blog/multiclass-confusion-matrix-for-object-detection (visited on 09/22/2024).

[227] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, *Detectron2*, 2019. [Online]. Available: https://github.com/facebookresearch/detectron2 (visited on 09/10/2024).

[228] A. Paszke, S. Gross, F. Massa, *et al.*, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Advances in Neural Information Processing Systems*, vol. 32, Curran Associates, Inc., 2019. [Online]. Available: https://papers.nips.cc/paper_files/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html (visited on 09/10/2024).

[229] T.-Y. Lin, P. Dollár, R. Girshick, *et al.*, "Feature Pyramid Networks for Object Detection," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 936–944. DOI: 10.1109/CVPR.2017.106.

[230] J. Lee, B. Bagheri, and H.-A. Kao, "A Cyber-Physical Systems Architecture for Industry 4.0-Based Manufacturing Systems," *Manufacturing Letters*, vol. 3, pp. 18–23, Jan. 2015, ISSN: 2213-8463. DOI: 10.1016/j.mfglet.2014.12.001.

[231] *Robotic Operating System*. [Online]. Available: https://www.ros.org (visited on 09/10/2024).

[232] *ROS Wrapper for Intel RealSense Devices*. [Online]. Available: https://github.com/IntelRealSense/realsense-ros (visited on 09/10/2024).

[233] *Universal Robots ROS Driver*. [Online]. Available: https://github.com/UniversalRobots/Universal_Robots_ROS_Driver (visited on 09/10/2024).

[234] F. Exner, *Universal Robot*. [Online]. Available: https://github.com/fmauch/universal_robot (visited on 09/10/2024).

[235] D. Ordonez, *Robotiq 2-Finger Grippers*. [Online]. Available: https://github.com/Danfoa/robotiq_2finger_grippers (visited on 09/10/2024).

[236] Tossy, *YOLO V4 for Darknet ROS*. [Online]. Available: https://github.com/Tossy0423/yolov4-for-darknet_ros (visited on 09/10/2024).

[237] K. Kleeberger, R. Bormann, W. Kraus, and M. F. Huber, "A Survey on Learning-Based Robotic Grasping," *Current Robotics Reports*, vol. 1, no. 4, pp. 239–249, Dec. 2020, ISSN: 2662-4087. DOI: 10.1007/s43154-020-00021-6.

[238] H. Zhang, X. Lan, S. Bai, *et al.*, "ROI-Based Robotic Grasp Detection for Object Overlapping Scenes," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Nov. 2019, pp. 4768–4775. DOI: 10.1109/IROS40897.2019.8967869.

[239] M. Görner, R. Haschke, H. Ritter, and J. Zhang, "MoveIt! Task Constructor for Task-Level Motion Planning," in *2019 International Conference on Robotics and Automation (ICRA)*, May 2019, pp. 190–196. DOI: 10.1109/ICRA.2019.8793898.

[240] E. Marchand, F. Spindler, and F. Chaumette, "ViSP for Visual Servoing: A Generic Software Platform with a Wide Class of Robot Control Skills," *IEEE Robotics and Automation Magazine*, vol. 12, no. 4, p. 40, Dec. 2005, ISSN: 1070-9932. DOI: 10.1109/MRA.2005.1577023.

[241] B. Mehta, M. Diaz, F. Golemo, C. J. Pal, and L. Paull, "Active Domain Randomization," *arXiv*, Jul. 2019. DOI: 10.48550/arXiv.1904.04762.

[242] S. Luo, H. Kasaei, and L. Schomaker, "Accelerating Reinforcement Learning for Reaching Using Continuous Curriculum Learning," in *2020 International Joint Conference on Neural Networks (IJCNN)*, Jul. 2020, pp. 1–8. DOI: 10.1109/IJCNN48605.2020.9207427.

[243] J. Ferret, O. Pietquin, and M. Geist, "Self-Imitation Advantage Learning," *arXiv*, Dec. 2020. DOI: 10.48550/arXiv.2012.11989.

[244] Q. Gallouédec, N. Cazin, E. Dellandréa, and L. Chen, "Panda-Gym: Open-Source Goal-Conditioned Environments for Robotic Learning," *arXiv*, Dec. 2021. DOI: 10.48550/arXiv.2106.13687.

[245] M. Plappert, M. Andrychowicz, A. Ray, *et al.*, "Multi-Goal Reinforcement Learning: Challenging Robotics Environments and Request for Research," *arXiv*, Mar. 2018. DOI: 10.48550/arXiv.1802.09464.

[246] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine, "D4RL: Datasets for Deep Data-Driven Reinforcement Learning," *arXiv*, Feb. 2021. DOI: 10.48550/arXiv.2004.07219.

[247] R. S. Sutton, "Learning to Predict by the Methods of Temporal Differences," *Machine Learning*, vol. 3, no. 1, pp. 9–44, Aug. 1988, ISSN: 1573-0565. DOI: 10.1007/BF00115009.

[248] D. Kumaran, D. Hassabis, and J. L. McClelland, "What Learning Systems do Intelligent Agents Need? Complementary Learning Systems Theory Updated," *Trends in Cognitive Sciences*, vol. 20, no. 7, pp. 512–534, Jul. 2016, ISSN: 1364-6613. DOI: 10.1016/j.tics.2016.05.004.

[249] Q. Gallouédec, *Panda-Gym Documentation*. [Online]. Available: https://github.com/qgallouedec/panda-gym (visited on 11/10/2024).

[250] *Gymnasium-Robotics Fetch Documentation*. [Online]. Available: https://robotics.farama.org/envs/fetch/index.html (visited on 11/10/2024).

[251] *Gymnasium-Robotics PointMaze Documentation*. [Online]. Available: https://robotics.farama.org/envs/maze/point_maze.html (visited on 11/10/2024).

[252] *Spinning Up Documentation of Soft Actor-Critic*. [Online]. Available: https://spinningup.openai.com/en/latest/algorithms/sac.html (visited on 09/10/2024).

[253] R. Agarwal, D. Schuurmans, and M. Norouzi, "An Optimistic Perspective on Offline Reinforcement Learning," in *Proceedings of the 37th International Conference on Machine Learning*, PMLR, Nov. 2020, pp. 104–114. [Online]. Available: https://proceedings.mlr.press/v119/agarwal20c.html (visited on 09/10/2024).

[254] A. P. Badia, B. Piot, S. Kapturowski, *et al.*, "Agent57: Outperforming the Atari Human Benchmark," in *Proceedings of the 37th International Conference on Machine Learning*, PMLR, Nov. 2020, pp. 507–517. [Online]. Available: https://proceedings.mlr.press/v119/badia20a.html (visited on 09/10/2024).

[255] J. Zhang, J. Huang, S. Jin, and S. Lu, *Vision-Language Models for Vision Tasks: A Survey*, Feb. 2024. DOI: 10.48550/arXiv.2304.00685.

[256] M. Coeckelbergh, *AI Ethics*. The MIT Press, 2020, ISBN: 9780262538190.

[257] *Regulation (EU) 2024/1689 of the European Parliament and of the Council of 13 June 2024 laying down harmonised rules on artificial intelligence and amending Regulations (EC) No 300/2008, (EU) No 167/2013, (EU) No 168/2013, (EU) 2018/858, (EU) 2018/1139 and (EU) 2019/2144 and Directives 2014/90/EU, (EU) 2016/797 and (EU) 2020/1828 (Artificial Intelligence Act)*, Legislative Body: CONSIL, EP, Jun. 2024. [Online]. Available: http://data.europa.eu/eli/reg/2024/1689/oj/eng (visited on 11/06/2024).

[258] S. Russell and P. Norvig, "Artificial Intelligence: A Modern Approach," in 3rd ed. Prentice Hall, 2010, pp. 1–16, ISBN: 9780134610993.

[259] R. Bellman, *An Introduction to Artificial Intelligence: Can Computers Think?* Boyd and Fraser Publishing Company, 1978, ISBN: 9780878350667.

[260] E. Charniak and D. V. McDermott, *Introduction to Artificial Intelligence*. Addison-Wesley, 1985, ISBN: 9780201119459.

[261] R. Kurzweil, *The Age of Intelligent Machines*. MIT Press, 1990, ISBN: 9780262610797.

[262] D. Poole, A. Mackworth, and R. Goebel, *Computational Intelligence: A Logical Approach*. Oxford University Press, 1998, ISBN: 9780195102703.

[263] Z. Liu, Y. Wang, S. Vaidya, *et al.*, *KAN: Kolmogorov-Arnold Networks*, May 2024. DOI: 10.48550/arXiv.2404.19756.

[264] A. K. Kolmogorov, "On The Representation of Continuous Functions of Several Variables by Superposition of Continuous Functions of One Variable and Addition," *Doklady Akademii Nauk SSSR*, vol. 114, pp. 369–373, 1957.

[265] L. Brunke, M. Greeff, A. W. Hall, *et al.*, "Safe Learning in Robotics: From Learning-Based Control to Safe Reinforcement Learning," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 5, pp. 411–444, May 2022, ISSN: 2573-5144. DOI: 10.1146/annurev-control-042920-020211.

[266] J. B. Rawlings, D. Q. Mayne, and M. Diehl, *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publishing, 2017, ISBN: 978-0-9759377-3-0.

[267] K. Zhou, J. C. Doyle, and K. Glover, *Robust and Optimal Control*. Prentice Hall, 1996, ISBN: 9780134565675.

[268] D. Mayne, M. Seron, and S. V. Raković, "Robust Model Predictive Control of Constrained Linear System with Bounded Disturbances," *Automatica*, vol. 41, pp. 219–224, Feb. 2005. DOI: 10.1016/j.automatica.2004.08.019.

[269] E. Altman, *Constrained Markov Decision Processes*. Chapman and Hall/CRC, 1999.

[270] A. Nilim and L. El Ghaoui, "Robust Control of Markov Decision Processes with Uncertain Transition Matrices," *Operations Research*, vol. 53, no. 5, pp. 780–798, Oct. 2005, ISSN: 0030-364X. DOI: 10.1287/opre.1050.0216.

[271] W. Zhao, T. He, R. Chen, T. Wei, and C. Liu, "State-Wise Safe Reinforcement Learning: A Survey," *arXiv*, Jun. 2023. DOI: 10.48550/arXiv.2302.03122.

[272] G. Dalal, K. Dvijotham, M. Vecerik, *et al.*, "Safe Exploration in Continuous Action Spaces," *arXiv*, Jan. 2018. DOI: 10.48550/arXiv.1801.08757.

[273] K. Srinivasan, B. Eysenbach, S. Ha, J. Tan, and C. Finn, "Learning to be Safe: Deep RL with a Safety Critic," *arXiv*, Oct. 2020. DOI: 10.48550/arXiv.2010.14603.

[274] H. Bharadhwaj, A. Kumar, N. Rhinehart, *et al.*, "Conservative Safety Critics for Exploration," *arXiv*, Apr. 2021. DOI: 10.48550/arXiv.2010.14497.

[275] B. Thananjeyan, A. Balakrishna, S. Nair, *et al.*, "Recovery RL: Safe Reinforcement Learning With Learned Recovery Zones," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4915–4922, Jul. 2021, ISSN: 2377-3766. DOI: 10.1109/LRA.2021.3070252.

[276] K. P. Wabersich and M. N. Zeilinger, "Linear Model Predictive Safety Certification for Learning-Based Control," in *2018 IEEE Conference on Decision and Control (CDC)*, Dec. 2018, pp. 7130–7135. DOI: 10.1109/CDC.2018.8619829.

[277] K. P. Wabersich, L. Hewing, A. Carron, and M. N. Zeilinger, "Probabilistic Model Predictive Safety Certification for Learning-Based Control," *arXiv*, Jan. 2021. DOI: 10.48550/arXiv.1906.10417.

[278] K. P. Wabersich and M. N. Zeilinger, "A Predictive Safety Filter for Learning-Based Control of Constrained Nonlinear Dynamical Systems," *arXiv*, May 2021. DOI: 10.48550/arXiv.1812.05506.

[279] S. P. Singh and R. S. Sutton, "Reinforcement Learning with Replacing Eligibility Traces," *Machine Learning*, vol. 22, no. 1, pp. 123–158, Mar. 1996, ISSN: 1573-0565. DOI: 10.1007/BF00114726.

[280] H. van Seijen, H. van Hasselt, S. Whiteson, and M. Wiering, "A Theoretical and Empirical Analysis of Expected Sarsa," in *2009 IEEE Symposium on Adaptive Dynamic Programming and Reinforcement Learning*, May 2009, pp. 177–184. DOI: 10.1109/ADPRL.2009.4927542.