DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics: Games Engineering

Vision-Based Continual Reinforcement Learning for Robotic Manipulation Tasks

Marcel Bruckner

DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics: Games Engineering

Vision-Based Continual Reinforcement Learning for Robotic Manipulation Tasks

Visionsbasiertes Kontinuierliches Verstärkendes Lernen für Robotermanipulation

Author:	Marcel Bruckner
Supervisor:	Alois Knoll, Prof. Dr.-Ing.
Advisor:	Josip Josifovski, M.Sc.
	Mohammadhossein Malmir, M.Sc
	Zhenshan Bing, Dr. rer.nat
Submission Date:	15.02.2022

I confirm that this master's thesis in informatics: games engineering is my own work and I have documented all sources and material used.

Munich, 15.02.2022 Marcel Bruckner

Abstract

The abstract is a brief summary of your thesis proposal • Length: 100 - 150 words (in certain circumstances up to 300 words) • Present a brief introduction to the issue • Outline the key tasks of your thesis

Contents

Abstract **iii**

1 Introduction **1**
1.1 Continual Learning and Catastrophic Forgetting 1
1.2 Continual Reinforcement Learning 2
1.3 Vision-Based Continual Reinforcement Learning for Robotic Manipulation Tasks 3
1.4 Research Questions 4
1.5 Experiments 4

2 Background **5**
2.1 Robotic Manipulation 5
2.2 Reinforcement Learning 5
2.2.1 Model-free Reinforcement Learning 5
2.2.2 Model-free Reinforcement Learning 6
2.3 Continual Learning Approaches 6
2.3.1 Regularization 7
2.3.2 Dynamic Architectures 7
2.3.3 Memory Systems 8
2.3.4 Meta-Learning 9
2.4 Un-/Self- Supervised Learning 9
2.5 State Representation Learning 9
2.6 Vision-Based End-to-End Learning 10
2.7 Vision-Based Continual Reinforcement Learning for Robotic Manipulation Tasks 10
2.8 Hypernetworks 11

3 Approach **12**
3.1 Overview 12
3.2 Deep Reinforcement Learning on Numeric Observations 12
3.2.1 PPO 13

4 Evaluation **14**
4.1 RL tasks to solve . . . 16
4.1.1 Reaching . . . 16
4.1.2 Pushing . . . 16
4.2 Experiments Design . . . 16
4.2.1 General Design (n-Runs, Epochs, Dataset, etc. . .) . . . 16
4.2.2 Train PPO on Numeric . . . 16
4.2.3 Train PPO on CNN . . . 16
4.2.4 Train PPO on Single-SRL . . . 16
4.2.5 Train PPO on Hypernetwork-SRL . . . 16
4.3 RL Performance Evaluation . . . 16
4.3.1 Cumulative Reward Numeric vs. CNN vs. Single-SRL. vs Hypernetwork . . . 16
4.3.2 Task-Solved Numeric vs. CNN vs. Single-SRL. vs Hypernetwork 16
4.3.3 Why on Numeric is Max . . . 16
4.3.4 Why CNN is max on visual . . . 16
4.3.5 Why SRL is worse than CNN . . . 16
4.3.6 Is Single-SRL better than Hypernetwork-SRL . . . 16
4.4 SRL Performance Evaluation . . . 16
4.4.1 Reconstruction of Ground Truth State . . . 16
4.4.2 Reward Prediction . . . 16
4.4.3 Ablation Study on Model Components . . . 16
4.4.4 Eigenvectors on Latent Space over runs . . . 16
4.4.5 k-NN on Latent Space . . . 16
4.5 CL Performance Evaluation . . . 16
4.5.1 Single-SRL vs. Hypernetwork-SRL on sequential test dataset . . 16
4.5.2 RL on Single-SRL for Continual Tasks . . . 16
4.5.3 RL on Hypernetwork-SRL for Continual Tasks . . . 16
4.6 Time and Compute Comparison . . . 16

List of Figures **17**

List of Tables **18**

Bibliography **19**

Appendix **22**

1 Introduction

Set the context for your project and captures the reader's attention • Explain the background of your study, starting from a broad picture narrowing in on your research question • Cite pertinent references

Describes the problem statement, illustrates why this is a problem and describes the contribution the thesis makes in solving this problem.

Optionally, it can give a short description (1-3 sentences each) of the remaining chapters.

Good introductions are concise, typically no longer than 4 pages.

The introduction reveals the full (but summarized) results of your work.

This appears counter-intuitive: does this not break the tension, like revealing the name of the murderer on the first page of a thriller?

Yes, it does. That is the whole point. A thesis, and thus its architecture, aims primarily to inform, not entertain.

1.1 Continual Learning and Catastrophic Forgetting

The human brain learns over the lifetime of a human continually as it approaches new and often distinct tasks. It learns from birth to death without forgetting what it has learned when previous. It even transfers old knowledge onto new tasks and improves performance on old ones by new experiences.
Machine Learning (ML) approaches assume the data they are using to be independent and identically distributed. As we fit a model to a dataset, we estimate the distribution of the data during learning to predict new instances in production. When the task changes, due to changes in the dataset or by changes in the training, the estimated distribution of the model changes accordingly [Les+19].
A neural network (NN), as a specific ML model, is a universal function approximator that estimates the data distributions. It consists of layers of neurons that carry a specific weight that is learned by gradient descent during training. As soon as the weights do not change anymore the NN has fit the data and can explain the data distribution.

In Continual Learning (CL) the aim is to train NNs on a sequential stream of tasks. The gradient descent method used to train NNs thus sequentially adapts the network

weights. This procedure incrementally overrides the learned weights, which leads to **catastrophic forgetting** [Kir+17] of previous tasks.
To prevent forgetting old tasks NNs need to exhibit a form of stability to retain the knowledge. On the other hand it has to exhibit enough plasticity to acquire new knowledge on previously unseen tasks. This is known as the **stability-plasticity dilemma** [Kir+17] of previous tasks.

Balancing the stability-plasticity dilemma while preventing catastrophic forgetting is the main goal of CL. Additionally, desired abilities of a CL system are to perform forward and backward transfer. In forward transfer, the model can reuse knowledge from previous tasks to improve learning and performance of following tasks. In backwards transfer, the model can incorporate new experiences to improve its performance on already learned tasks.

1.2 Continual Reinforcement Learning

In the supervised-learning setting all data is available to the model at once and independent and identically distributed (iid). As a RL learning agent interacts with its environment, it receives a constant stream of data. This data stream evolves over time, and the environment changes as the agent moves, manipulates and observes. The data the agent receives is non-independent, as future states depend on the current state, and not identically distributed, as the probability to observe certain states is high in easy to reach states and low in complex parts of the environment. This makes the data stream continuous and non-stationary.

CL addresses these continual and non-stationary data streams. A CL agent aims to maximize a function defining its performance when it is trained on a sequence of tasks. In Continual Reinforcement Learning (CRL) we have multiple sources and degrees of non-stationarity [Khe+20].

The environment the agent interacts with can be stationary, so it is independent of how the agent interacts with it and also does not evolve. More common in the RL setting is that the environment evolves passively, so what the CRL agent does has no influence on the environment. It rather evolves over time by influences like weather, lighting or other agents that the CRL agent has no active influence on. Additionally, the environment might actively change as a result of the agents interactions. When a CRL agent moves through the environment, the states it can observe might change and as the agent places, pushes, pulls or stacks objects it changes the environment. Clearly, active and passive changes can also occur at the same time in hybrid environments.

The underlying MDP defines the environment in the RL setting. The degree of the non-stationarity is defined by which parts of the MDP change over time. In a first degree CRL setting either of the observation function, the actions the agent can take, the state transition probabilities or the reward function are non-stationary and evolve over time. In a second degree setting any two of the above change, and so on for third and fourth degree non-stationarities.

The multiple combinations of types and degrees of non-stationarity imply that RL is a natural fit to apply the approaches that are concerned with CL.

1.3 Vision-Based Continual Reinforcement Learning for Robotic Manipulation Tasks

In vision-based learning, the sole input to the learning system is raw RGB camera data. The input images are very high-dimensional and thus are hard to process by RL algorithms [Ven18]. In State Representation Learning (SRL) we compress the data into a lower-dimensional latent space. The SRL is included into the learning pipeline and requires no human feature engineering.

The SRL modules often consist of Variational Auto Encoder (VAE) networks [KW13] or its derivatives [Hig+17; GMO18]. By learning the intermediate representations the system can learn faster and more robust [GNZ17], as higher level modules can rely on the extracted and compressed information of lower levels. SRL extracts a unified representation over the state space and decouples RL models from numerical features that are measured by the agent [Raf+19]. This is especially useful as numerical measurements are often prone to errors and sensor drift.

With the growing need of robotic aid in production lines, we aim to implement a vision-based CRL model for robotic manipulation tasks. Training in a CRL fashion has the goal that robots continually learn distinct tasks without catastrophic forgetting. This is especially useful in fast changing production lines with a growing need of robots capable to solve multiple tasks. The vision-based approach we chose enables a fast setup of the system and does not require lengthy sensor calibration processes.

In this thesis we aim to train a robotic manipulation arm (Kuka LBR iiwa [Kuk21]) using the described techniques of vision-based CRL. Our aim is to train a CRL agent to control the robotic arm on a sequence of tasks. We start the training on a simulator and then transfer the models to the real robot.

As input to the CRL system we only use visual observations taken by a camera mounted atop of the robotic arm or a virtual camera in a simulator. The goal of our approach is to learn the two object manipulation tasks of moving towards an object and pushing an object.

1.4 Research Questions

In this thesis we explore techniques to train RL agents on visual data to enable them to learn without access to their numeric state. To enable the agent to learn we explore how to use SRL and its potential benefits for the training of the agent. We want to enable the agent to solve two distinct tasks sequentially without access to data from the previous tasks. To tackle this problem of vision-based continual reinforcement learning for robotic manipulation tasks we answer the following research questions:

- How well does an RL agent trained on image features perform compared to an agent trained on numeric features? Can the vision-based agent solve the given tasks as well as the numeric agent?
- How well does an agent trained on raw image features perform compared to an agent trained upon the latent state of the SRL module? Is there a benefit in using the latent state from SRL over the raw image features?
- How well can the CSRL module encode the latent state when trained on sequential tasks compared to the SRL module trained on the tasks separately? How well can the CSRL module avoid catastrophic forgetting?
- How well does the CRL agent trained sequentially on either numeric or latent states perform compared to the RL agent trained separately? How well can the CRL agent avoid catastrophic forgetting?

1.5 Experiments

Describe what and how we use it to test what we have done.

2 Background

Defines the fundamental concepts your thesis builds on.

Your thesis implements a new type of parser generator and uses the term non-terminal symbol a lot?

Here is where you define what you mean by it.

The key to this chapter is to keep it very, very short.

Whenever you can, dont reinvent a description for an established concept, but reference a text book or paper instead.

If a reader cares enough about your topic to read your thesis, he probably knows the terms anyway.

The chapter thus mostly serves to disambiguate overloaded terms and as a collection of pointers to more detailed sources for novice readers.

2.1 Robotic Manipulation

2.2 Reinforcement Learning

2.2.1 Model-free Reinforcement Learning

In Reinforcement Learning (RL) an ML agent learns how to solve tasks via sequences of decisions, opposed to supervised learning where an agent learns an input-output mapping from labelled data. The RL agent learns to solve its tasks while acting in an environment that is uncertain and potentially complex. The underlying assumption in RL is that the environment behaves according to a Markov Decision Process (MDP) [Bel57].

An MDP is a tuple

$$\text{MDP} = \langle S, A, P, R, \gamma \rangle . \tag{2.1}$$

It consists of the state of possible spaces the environment can take on. Often the agent cannot observe the full state of environment, as it might be to complex or the sensor of the agent do not have enough range. Thus, the agent observes the state of the environment via its observation function

$$o_s = observe(s \in S), \tag{2.2}$$

whereas the set of observations O is a subset of the original states $O \subseteq S$.
The RL agent interacts with the environment via a set of actions A. These actions define the possibilities the agent has to influence the environment and to change its own and the environments state.
The state transition matrix P defines how the environment evolves as the agent interacts. Especially in RL, the state transition probability matrix

$$P^a_{ss'} = \mathbb{P}[S_{t+1} = s' | S_t = s, A_t = a] \tag{2.3}$$

defines the transition probabilities from all states $s \in S$ to all successor states $s' \in S$ as the agent takes the action $a \in A$.
The reward function R gives feedback to the agent based on how good the current state it observes is for the task to solve and the actions it takes. The reward function

$$R^a_s = \mathbb{E}[R_{t+1} | S_t = s, A_t = a] \tag{2.4}$$

is a function of the current state $s \in S$ and action $a \in A$ the agent takes. R^a_s rewards the agent by reaching certain states beneficial to its task based on the actions it takes.
The goal of the agent is to maximize the cumulative reward

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} = \gamma^k R_{t+k+1} \tag{2.5}$$

over all future time steps.
The discount factor $\gamma \in [0, 1]$ gives the future rewards a weighting towards the cumulative reward. It discounts rewards that are distant in the future as the agent cannot tell certain how the rewards will evolve.

2.2.2 Model-free Reinforcement Learning

1. Explain the learning of the value and policy function

2.3 Continual Learning Approaches

Gradient descent methods override the NN weights whenever it sees new data during training. If the new data is not from the same distribution, the NN will forget about the distribution of the previously seen data. To overcome the catastrophic forgetting this evokes, several approaches have been proposed.

2.3.1 Regularization

NNs are trained using gradient descent to update the learnable weights, whereas the gradient of different tasks often point into different directions. CL approaches based on regularization alter the gradients to prevent forgetting. Regularization forces the gradients of the currently learned task to align with gradients of the previously learned tasks. [Yu+20] proposed a gradient surgery approach that projects a task's gradient onto the normal plane of the gradient of any other task that has a conflicting gradient. This eliminates potential interference, as it drives the model towards similar minima for different tasks.

Another way is to regularize the loss on new tasks to minimize forgetting of previous tasks. This approach estimates the importance of each model parameter for the previous tasks and penalizes changes to the parameters based on their importance [Had+20]. [Kir+17] introduced the Elastic Weight Consolidation (EWC) model, where they reduced the plasticity of neurons that are important to previous tasks. They regularize using the difference between the parameters of the old and the new tasks and incorporation the fisher information matrix to estimate the importance of each neuron. [ZPG17] introduced the Synaptic Intelligence (SI) model, where each neuron locally estimates its importance in solving tasks the NN has been trained on.

[HVD15] proposed to transfer knowledge of a NN into another using knowledge distillation. When a NN approaches a new task, a copy of the NN is stored as the teacher model [Mir+19] to prevent overriding the embedded knowledge. In the Learning without Forgetting (LwF) approach by [LH16], during the ongoing training of the student model on a new task it is regularized using the copied teacher model. This forces the student model to infer the same predictions as the teacher on similar data, while learning about new data simultaneously. After training, the student model has embedded the knowledge about the previous tasks learned from the teacher together with the new task specific knowledge.

2.3.2 Dynamic Architectures

In the dynamic architectures approach the network architecture is changed as new tasks are solved, opposed to the regularization based approaches where gradients are changed.

Explicit architecture modifications can be done by adding, cloning or saving parts of the model parameters to avoid forgetting. In the Progressive Neural Networks (PNN) model by [Rus+16] they add a new network whenever there is a new task. They transfer knowledge of old tasks to the new instances via lateral connections between the networks layers. This approach specifically prevents catastrophic forgetting as they only train the newest instance and fix the old networks weights. No backwards transfer can be performed by PNNs as older network instances have fixed weights. In the LwF approach [LH16] the authors add output layers per task and thus reuse lower layer knowledge across tasks. The dynamically expanding network (DEN) models of [Yoo+17] increase their number of trainable weights according to the estimated needed capacity for new tasks.

Implicit architecture modifications inactivate neurons based on their importance for learned tasks or change the forward pass path according to the task to solve. These approaches do not change the architecture of the network, but rather introduce an ensemble of networks within the network. They achieve the implicit modification by masking parameters per task as it is done in the PackNet and Piggyback models by [ML17; MDL18]. The Hard Attention to the Task (HAT) model by [Ser+18] uses an attention mechanism that prevents forgetting of previous task information while preserving plasticity of the network to learn new tasks. For training the PathNet model by [Fer+17] the authors used a genetic algorithm to learn the optimal paths for gradient flow through the network per task.

Dual and multi architecture approaches split the network architecture into two or more models. A plastic and fast adapting model learns the currently observed task, while one or multiple models store memory about past experiences. In the FearNet model by [KK17] three models resemble the dual memory as found in the human brain.

2.3.3 Memory Systems

Inspired by the human brain there is also the approach of using memory systems to encode, store and recall knowledge or experience.

Replay, also called rehearsal, based approaches store a set of observations from previously seen tasks. During training of the NN the memory system replays samples of the stored observations and interleaves them with currently seen observations. The NN thus sees data from different tasks simultaneously and learns as in a multi-task setting. To minimize memory consumption we only store observations that have a high impact for learning, such as observations that exceeded a large loss or resulted in strong gradients. In the iCaRL model by [Reb+16] the authors carefully sort the seen observations and only store a coreset consisting of only the most representative samples. Active Long Term Memory (A-LTM) networks introduced by [Fur+16] store a coreset that is used for regularization during knowledge distillation. Gradient Episodic Memory (GEM) and Averaged Gradient Episodic Memory (A-GEM) networks by [LR17; Cha+19] regularize the gradients based on the stored episodes.

Pseudo-rehearsal approaches do not store explicit knowledge, but rather encode knowledge into a generative model. The generative models are usually autoencoder

(AE) networks [KW13] or generative adversarial (GAN) networks [Goo+14]. During training, the pseudo-rehearsal system generates sample experiences from previous tasks and interleaves them with current experience. The model learns to solve the current and the replay task jointly and prevents catastrophic forgetting of old tasks. The FearNet model by [KK17] uses pseudo-rehearsal to resemble the long term memory of the human brain. In the Reinforcement-Pseudo-Rehearsal (RePR) model by [Atk+20] the authors use a combination of knowledge distillation and generated observations to transfer knowledge from a teacher to a student model.

2.3.4 Meta-Learning

Meta-learning, often also called learning to learn, is a data driven approach for improving an agent's learning efficiency [Khe+20]. This approach is less concerned with forgetting old knowledge, but rather how to quickly adapt to new environments and tasks. Agents in this regime learn to alter their own optimization process, whereas the modifications to the learning process will generalize into the future.

In the meta-training and meta-testing setting a NN attempts to learn to alter its own optimization process. The learning consists of an inner loop that is responsible for fast time-scale learning and quick adaption to new tasks. Another outer loop is a slower process of learning about the learning of the inner loop. The meta-training phases consist of meta-updates where the inner loop optimizes on a meta-train training dataset. Based on the performance of the inner loop based on a meta-train testing dataset the outer loop performs a learning step. During a final meta-testing phase the inner loop trains on a hold out meta-test training dataset. The performance and the ability to quickly adapt to the meta-testing environment is measured on a meta-test testing dataset. The Model Agnostic Meta-Learning (MAML) framework by [FAL17] performs standard gradient descent within the inner loop, whereas the outer loop computes a gradient to improve the performance of the inner loop learning process.

2.4 Un-/Self- Supervised Learning

2.5 State Representation Learning

Overview: https://arxiv.org/abs/1802.04181

2.6 Vision-Based End-to-End Learning

In End-to-End (E2E) learning, we train complex learning systems holistically by gradient descent on all parts of the system. All modules that form an E2E systems are designed to be differentiable and thus learnable. In the E2E approach, not only a central NN performing a specific subtask on a problem is trained, but also all peripheral modules like State Representation Learning (SRL) or memory formation. By merging complex data preprocessing components into the ML pipeline, E2E learning blurs the classical boundaries between Data Engineering and ML. E2E learning results in a unified learning scheme for all parts of the pipeline and uses available data to optimize every module. Opposed to decomposing problems by human engineering, as in the classical design of complex learning systems, training in an E2E manner ignores this explicitly decomposition. During training, the structural preconditioning in the data is sufficiently strong to train the E2E system. [GNZ17]

In vision-based E2E learning, the sole input to the learning system is raw RGB camera data. The input images are very high-dimensional and thus are hard to process by RL algorithms. To process the high-dimensional image data no explicit feature extractors are designed, instead SRL is used to compress the data into a lower-dimensional latent space. In vision-based E2E learning the SRL modules often consist of AE networks [KW13] or GAN networks [Goo+14]. By learning the intermediate representations the system can learn faster and more robust [GNZ17] as higher level modules can rely on the extracted and compressed information of lower levels. The SRL modules extract a unified representation over the state space and decouple [Raf+19] RL models from numerical features that are measured by the agent and often prone to errors and sensor drift.

2.7 Vision-Based Continual Reinforcement Learning for Robotic Manipulation Tasks

The aim of this thesis is to train a robotic manipulation arm (Kuka LBR iiwa [Kuk21]) in a Vision-Based Continual Reinforcement Learning fashion. Our aim is to train a CRL agent to control the robotic arm on a sequence of tasks using the CRL approach described in **??**. As input to the E2E CRL system, we only use visual observations taken by a camera mounted atop of the robotic arm. This enables us to train the system in an E2E manner.

In our CRL setup we observe active and passive non-stationarity (section 1.2) as our robotic manipulator arm actively influences the objects in the environment. In fact, the aim of this thesis is to learn different object manipulation tasks like moving, pushing, throwing and stacking of objects. Additionally, when we transfer the learned agent from the simulation environment a passive non-stationarity is introduced. These non-stationarities are of second degree, as the reward function changes during learning defined by a change in the tasks to solve, and as the state transition matrix changes as the model dynamics change when switching from simulation to reality.

2.8 Hypernetworks

3 Approach

Outlines the main thing your thesis does.
Your thesis describes a novel algorith for X?
Your main contribution is a case study that replicates Y?
Describe it here.

3.1 Overview

In this thesis we explore vision-based continual reinforcement learning for robotic manipulation tasks. We apply concepts and algorithms from reinforcement learning, continual learning, state representation learning and computer vision.

We use the KUKA LBR iiwa robot [Kuk21] displayed in Figure 3.1. It is an industrial grade assembly robot that is lightweight and designed for cooperation with humans. The robot consists of seven rotating joints that can move between 240 and 359 degrees. The seven degrees of freedom allow the robot to span a range up to 1.6 meters around its base.

3.2 Deep Reinforcement Learning on Numeric Observations

We train a neural-network-based model to control the robot, such that it learns to solve given tasks. We use deep reinforcement learning (DRL) to train the model.

State

Observation

Action Space

Transition Function

Reward

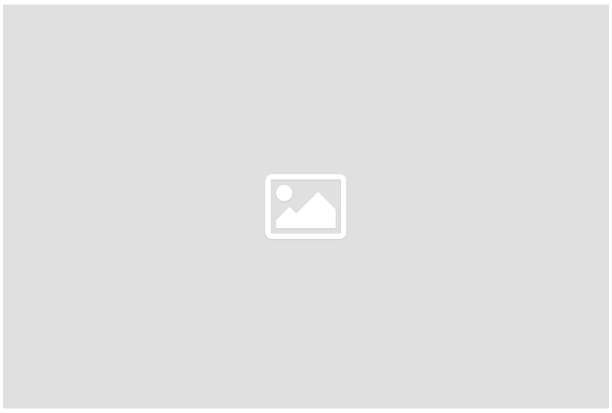

Figure 3.1: The Kuka IIWA [Kuk21] robot we use in this thesis.

3.2.1 PPO

We use the Proximal Policy Optimization algorithm

4 Evaluation

Describes why your approach really solves the problem it claims to solve. You implemented a novel algorithm for X? This chapter describes how you ran it on a dataset and reports the results you measured. You replicated a study? This chapter gives the results and your interpretations.

4.1 RL tasks to solve

4.1.1 Reaching

4.1.2 Pushing

4.2 Experiments Design

4.2.1 General Design (n-Runs, Epochs, Dataset, etc. . .)

4.2.2 Train PPO on Numeric

4.2.3 Train PPO on CNN

4.2.4 Train PPO on Single-SRL

4.2.5 Train PPO on Hypernetwork-SRL

4.3 RL Performance Evaluation

4.3.1 Cumulative Reward Numeric vs. CNN vs. Single-SRL. vs Hypernetwork

4.3.2 Task-Solved Numeric vs. CNN vs. Single-SRL. vs Hypernetwork

4.3.3 Why on Numeric is Max

4.3.4 Why CNN is max on visual

4.3.5 Why SRL is worse than CNN

4.3.6 Is Single-SRL better than Hypernetwork-SRL

4.4 SRL Performance Evaluation

4.4.1 Reconstruction of Ground Truth State

4.4.2 Reward Prediction

4.4.3 Ablation Study on Model Components

4.4.4 Eigenvectors on Latent Space over runs

4.4.5 k-NN on Latent Space

4.5 CL Performance Evaluation

4.5.1 Single-SRL vs. Hypernetwork-SRL on sequential test dataset

4.5.2 RL on Single-SRL for Continual Tasks

4.5.3 RL on Hypernetwork-SRL for Continual Tasks

4.6 Time and Compute Comparison

List of Figures

3.1 The Kuka IIWA [Kuk21] robot we use in this thesis. 13

List of Tables

Bibliography

[Atk+20] C. Atkinson, B. McCane, L. Szymanski, and A. Robins. “Pseudo-Rehearsal: Achieving Deep Reinforcement Learning without Catastrophic Forgetting.” In: (Dec. 2020). DOI: `10.1016/j.neucom.2020.11.050`.

[Bel57] R. Bellman. *Dynamic Programming*. Dover Publications, 1957. ISBN: 9780486428093.

[Cha+19] A. Chaudhry, M. Ranzato, M. Rohrbach, and M. Elhoseiny. “Efficient Lifelong Learning with A-GEM.” In: *7th International Conference on Learning Representations, ICLR 2019* (Dec. 2019).

[FAL17] C. Finn, P. Abbeel, and S. Levine. “Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks.” In: (Mar. 2017).

[Fer+17] C. Fernando, D. Banarse, C. Blundell, Y. Zwols, D. Ha, A. A. Rusu, A. Pritzel, and D. Wierstra. “PathNet: Evolution Channels Gradient Descent in Super Neural Networks.” In: (Jan. 2017).

[Fur+16] T. Furlanello, J. Zhao, A. M. Saxe, L. Itti, and B. S. Tjan. “Active Long Term Memory Networks.” In: (June 2016).

[GMO18] A. Graves, J. Menick, and A. van den Oord. “Associative Compression Networks for Representation Learning.” In: (Apr. 2018).

[GNZ17] T. Glasmachers, Y.-K. Noh, and M.-L. Zhang. *Limits of End-to-End Learning*. 2017, pp. 17–32.

[Goo+14] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. “Generative Adversarial Networks.” In: (June 2014).

[Had+20] R. Hadsell, D. Rao, A. A. Rusu, and R. Pascanu. *Embracing Change: Continual Learning in Deep Neural Networks*. Dec. 2020, pp. 1028–1040. DOI: `10.1016/j.tics.2020.09.004`.

[Hig+17] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, A. Lerchner, and G. Deepmind. *Beta-VAE: LEARNING BASIC VISUAL CONCEPTS WITH A CONSTRAINED VARIATIONAL FRAMEWORK*. 2017.

[HVD15] G. Hinton, O. Vinyals, and J. Dean. "Distilling the Knowledge in a Neural Network." In: (Mar. 2015).

[Khe+20] K. Khetarpal, M. Riemer, I. Rish, and D. Precup. "Towards Continual Reinforcement Learning: A Review and Perspectives." In: (Dec. 2020).

[Kir+17] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell. *Overcoming catastrophic forgetting in neural networks*. 2017.

[KK17] R. Kemker and C. Kanan. "FearNet: Brain-Inspired Model for Incremental Learning." In: *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings* (Nov. 2017).

[Kuk21] Kuka. "Sensitive robotics LBR iiwa." 2021.

[KW13] D. P. Kingma and M. Welling. "Auto-Encoding Variational Bayes." In: (Dec. 2013).

[Les+19] T. Lesort, V. Lomonaco, A. Stoian, D. Maltoni, D. Filliat, and N. Diaz-Rodriguez. "Continual Learning for Robotics: Definition, Framework, Learning Strategies, Opportunities and Challenges." In: (June 2019).

[LH16] Z. Li and D. Hoiem. "Learning without Forgetting." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40 (12 June 2016), pp. 2935–2947.

[LR17] D. Lopez-Paz and M. Ranzato. "Gradient Episodic Memory for Continual Learning." In: *Advances in Neural Information Processing Systems* 2017-December (June 2017), pp. 6468–6477.

[MDL18] A. Mallya, D. Davis, and S. Lazebnik. "Piggyback: Adapting a Single Network to Multiple Tasks by Learning to Mask Weights." In: (Jan. 2018).

[Mir+19] S.-I. Mirzadeh, M. Farajtabar, A. Li, N. Levine, A. Matsukawa, and H. Ghasemzadeh. "Improved Knowledge Distillation via Teacher Assistant." In: (Feb. 2019).

[ML17] A. Mallya and S. Lazebnik. "PackNet: Adding Multiple Tasks to a Single Network by Iterative Pruning." In: (Nov. 2017).

[Raf+19] A. Raffin, A. Hill, R. Traore, T. Lesort, N. Diaz-Rodriguez, and D. Filliat. "Decoupling feature extraction from policy learning: assessing benefits of state representation learning in goal based robotics." In: (Jan. 2019).

[Reb+16] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert. “iCaRL: Incremental Classifier and Representation Learning.” In: *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017* 2017-January (Nov. 2016), pp. 5533–5542.

[Rus+16] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell. “Progressive Neural Networks.” In: (June 2016).

[Ser+18] J. Serra, D. Suris, M. Miron, and A. Karatzoglou. *Overcoming Catastrophic Forgetting with Hard Attention to the Task*. 2018.

[Ven18] N. Venkat. *The Curse of Dimensionality: Inside Out*. 2018.

[Yoo+17] J. Yoon, E. Yang, J. Lee, and S. J. Hwang. “Lifelong Learning with Dynamically Expandable Networks.” In: *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings* (Aug. 2017).

[Yu+20] T. Yu, S. Kumar, A. Gupta, S. Levine, K. Hausman, and C. Finn. “Gradient Surgery for Multi-Task Learning.” In: (Jan. 2020).

[ZPG17] F. Zenke, B. Poole, and S. Ganguli. *Continual Learning Through Synaptic Intelligence*. July 2017, pp. 3987–3995.

Appendix