CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

A Chalmers University of Technology Master's thesis template for LaTeX

A Subtitle that can be Very Much Longer if Necessary

Master's thesis in Computer science and engineering

NAME FAMILYNAME

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2022

Master's thesis 2022

A Chalmers University of Technology Master's thesis template for LaTeX

A Subtitle that can be Very Much Longer if Necessary

NAME FAMILYNAME

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2022

A Chalmers University of Technology Master's thesis template for LaTeX
A Subtitle that can be Very Much Longer if Necessary
NAME FAMILYNAME

Supervisor: Name, Department
Advisor: Name, Company or Institute (if applicable)
Examiner: Name, Department

Master's Thesis 2022
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Description of the picture on the cover page (if applicable)

Typeset in LaTeX
Gothenburg, Sweden 2022

A Chalmers University of Technology Master's thesis template for LaTeX
A Subtitle that can be Very Much Longer if Necessary
NAME FAMILYNAME
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Abstract text about your project in Computer Science and Engineering.

Keywords: Computer, science, computer science, engineering, project, thesis.

Acknowledgements

Here, you can say thank you to your supervisor(s), company advisors and other people that supported you during your project.

Name Familyname, Gothenburg, May 2022

Contents

List of Figures

List of Tables

1
Introduction

Before 1980s Reinforcement learning(RL) has two main threads.One thread associated with learning by trail and error in animal psychology and the second thread concerned with optimal control and its solution in value function and dynamic programming[1].A third less common type of reinforcement learning is temporal difference method which is used in tic-tac-toe.After 1980s by combining these three threads modern reinforcement learning has emerged.

Reinforcement learning is learning what to do—how to map situations to actions—so as to maximize a numerical reward signal[1].It is a discipline that encompass learning problem,a class of solutions related to learning and field of study of problems in learning and solution methods. It is most of time confused with unsupervised learning because it does not rely on examples of correct behavior or labeled data. unlike unsupervised learning reinforcement learning is trying to maximize a reward signal instead of learning to find hidden structure.

All reinforcement learning agents have explicit goals, can sense aspects of their environments, and can choose actions to influence their environments.One of the two successful approaches to solve sequential decision making commonly known as Markov Decision Process(MDP) optimisation problem is reinforcement learning[7].Markov decision processes are intended to include just three aspects sensation,action and goal in their simplest possible forms without trivializing any of them[1].we seek actions that brings about state of highest value not highest reward,because these actions obtains the greatest amount of reward for us over the long run.

Existing Reinforcement learning algorithms could be categorised as being either model based or model-free.Currently, the best performing algorithms are of the model-free type, they do not explicitly model the dynamics of the environment.one of the benefits of using model-free algorithms is its generality and applicability to a wide range of problems.In addition to not modeling the dynamics of the environment,the lack of interpretability of model -free algorithm limits there application in safety-critical control systems.

Many recent success in advancing reinforcement learning through deep learning was kick-stared by Deep Q-Networks algorithm [2][3].Its is a combination of convulational neural network trained with a variant of Q-learning from raw pixels has improved agents to play many atari games at human-level performance.One of major problem of this algorithm is overestimation of action values under certain conditions,

to address this double Q-learning algorithm(DDQN) [4] was introduced ,which was initially proposed in tabluar setting [5].By decoupling selection and evaluation of the bootstrap action the overestimation bias of DQN were resolved.later the dueling network architecture [6] helps to generalize across actions by separately representing state values and action advantage.Since then, many extensions have been proposed that enhance the speed or stability of DQN.

Model based reinforcement learning can be defined as any MDP approach that uses a model(known or learned) and uses learning to approximate a global value or policy function[7].model-based approaches [6] offer an avenue to address the two most common issues related to model-free algorithms modeling the dynamics of the environment and the lack of interpretability. Model-based reinforcement learning converge faster than their model-free algorithms.

In model-based reinforcement learning both the model and the policy are learned.In order to successfully learn the two model-based algorithm needs to balance between exploration and exploitation to find the optimal policy (like a model-free algorithm) and also balance between learning the model and learning the policy itself. Recent studies on model based reinforcement learning have shown a promise in this paradigm by obtaining a state of the art performance on offline reinforcement learning benchmarks [10][11],improving a powerful model-free approach (e.g.[12]).

The main problem with model based reinforcement learning is balancing between learning the model and policy.if the learned model is not a sufficiently good representation of the environment, or if it models only certain parts of the environment well, the learned policy will in effect over-fit on that model, resulting in poor performance. On the other hand, if a policy is not performing well during training, the agent won't be able to access new parts of the environment (for example the agent will not be able to progress to a new level in a game). Often this results in model-based algorithms converging to a poorer policy than their model-free counterparts.

In recent years we have seen the use of meta-learning in reinforcement learning.which is a form of learning how to learn. Meta-learning algorithms aim to learn models that can adapt to new scenarios or tasks with few data points.Meta-learning, in the context of RL, aims to learn a policy that adapts fast to new tasks or environments [12,13,14].Another extension of model based reinforcement learning is offline learning ,which utilizes previously collected data with out additional online data collection.In this learning the stochasticity of the environment is maintained by limiting the interaction of the agent to collect additional transition by interacting with the environment by using the behaviour policy.

In Practical application,Although the above stated have shown remarkable adavance in the field,it is still difficult to learn a precise environment dynamic models.In previous studies[2,3,4,6,16] deep Q-learning was trained using raw pixel image,which remarks one of greater advantages of using deep learning in reinforcement learning.Training reinforcement learning agents from high dimensional image representa-

tions can be very computationally expensive.By using Deep auto-encoders for reinforcement learning to compress high dimensional data such as pixelated images into small latent representation [21] were able to achieve improved performance compared to a baseline reinforcement learning agent with a convolutional feature extractor, while using less than 2W of power.While it is not a novel idea to use auto-encoders to accelerate reinforcement learning we have not found a substantial work focused on the use of compressed representation by auto-encoders to learn to model the dynamic on an Atari environment and subsequently train a policy completely in a simulation.

2
Background

Markove Decision Processes(MDPs):

Markov decision processes model decision making in discrete, stochastic, sequential environments[9].The formal definition of Markov decision process can be summarised in the tuple $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, \mathcal{P}(\mathcal{S}_l), \mathcal{T}, r\}$[8]. The goal of the model is that an agent inhabits a stochastic environment in a response to action choice made by the agent[9].

The environment consists of the the Transition function, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \to (\mathcal{PS})$.
and the reward function $(r(s_t, a_t))$, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$.In MDPs the transition probability and reward only depends on the current state and the action chosen by the agent but not the past state and action.
The agent acts in environment according to a policy $\pi : \mathcal{S} \to (\mathcal{PS})$.policy learned can be off policy where the behaviour policy is different from the policy used for action selection one common example is Q-learning or on-policy methods which attempts to evaluate or improve the policy that is used to make decision.
The former state-action value function (Q function for short)can be computes recursively with dynamic programming:

$$Q^{\pi}(s, a) = E'_s[r + \gamma E'_a \ \pi(s')[Q^{\pi}(s', a')]|s, a, \pi]$$

(2.1)

optimal q value under deterministic policy

$$a = argmax'_a \epsilon A Q^*(s, a') \tag{2.2}$$

$$Q^*(s, a) = \max_{\pi} Q^{\pi}(s, a)$$

(2.3)

if the optimal function satisfies the bellman equation:

$$Q^*(s, a) = E'_s[r + \gamma max(a')Q^*(s', a')|s, a, \pi] \tag{2.4}$$

The advantage function describes “how good” the action a is, compared to the expected return when following direct policy pi.

$$A^{\pi}(s,a) = Q^{\pi}(s,a) - V^{\pi}(s) \tag{2.5}$$

The value function V measures the how good it is to be in a particular state s. The Q function, however, measures the the value of choosing a particular action when in this state[6].

Deep Reinforcement Learning and DQN

(paraphrase)After deep learning has shown remarkable results in learning from raw pixel data in computer vision it's application has been adopted to solve many classical learning problems.The goal of Deep Reinforcement Learning is to connect reinforcement learning algorithm to deep neural network which operates directly on RGB images and efficiently process training data by using stochastic gradient updates[2].

The use of label data and the assumption the distribution of data to be identical and independent through out training process in deep learning makes it complex to use it directly into reinforcement learning algorithms which must be able to learn from sparse ,noisy and delayed reward and highly correlated data.To address this issues and successfully apply deep learning to reinforcement learning [2] used experience replay mechanism[15] which randomly samples previous transitions, and thereby smooths the training distribution over many past behaviors.Convolutional neural network was used to learn successful control policies from raw video data in complex reinforcement learning environment.The network is trained with a variant of Q-learning algorithm,with stochastic gradient descent to update the weights.

This architecture was based on Tesauro's TD-Gammon architecture [17]which updates the parameters of the network that estimates the value function directly from on-policy samples of experience. Similar to this approach the online network in DQN utilize a technique known as experience replay [18] where the agent's experiences at each time-step, $\rceil_t = (s_t, a_t, r_t, s_t + 1)$ is stored in a data set $\mathcal{D} = e1, ..., e_N$ pooled over many episodes into a replay memory.By drawing random samples from this pool of stored experiences the Q-learning is updated.After performing experience replay, the agent selects and executes an action according to an -greedy policy.The use of experience replay and target networks enables relatively stable learning of Q values, and led to super human performance on several Atari games.

The advantage of using Deep Q-learning over Q-learning includes allowing to have greater sample efficiency,reduced variance by randomising the sample bias and avoiding being stuck in local minimum.Drawback DQNs are it only handle discrete,low-dimensional action space.

2.0.1 Extension of DQN

Several Studies were performed to increase the performance of DQNs(Paraphrase).

2.0.1.1 Double Deep Q-networks: DDQN

DQN suffer from overestimation bias due to due to the maximization step in optimisation function in Q-learning.Q-learning can overestimate actions that have been tried often and the estimations can be higher than any realistic optimistic estimate .Double Q-learning [19], addresses this overestimation by decoupling, in the maximization performed for the bootstrap target, the selection of the action from its evaluation.Double Q-learning stores two Q-functions,The average of the two Q values for each action and then performed ϵ-greedy exploration with the resulting average Q values.It was successfully combined with DQN to reduce overestimations.
TODOS: DDQN target equation

2.0.1.2 Prioritized replay

The main use of replay buffer is to sample transitions with maximum probability.Both DQN and DDQN samples experiences uniformly.Prioritized replay [20] samples transitions using the maximum priority ,providing a bias towards recent transitions and stochastic transitions even when there is little left to learn about them.

2.0.1.3 Dueling Network

Dueling Network was designed for value based learning,this architecture separates the representation of sate-value and state-dependent action advantages without supervision[6].its consists of two streams that represents the value and advantage functions,while sharing a common convolutional feature learning module.This network has a single Q-learning network with two streams that replace DQN architecture[3].

$$Q(s,a;\theta,\alpha,\beta) = V(s,\theta,\beta) + A(s,a;\theta,\alpha) \tag{2.6}$$

if needed TODO ADD EQUATION:factorization of action values

2.0.1.4 Multi-step learning

Previously stated extension of DQN have indicated that the use deep learning has enhanced the learning capability of Q-learning.The performance of Q-learning is still limited by greedy action selection after accumulating a single reward.An alternative approach was multi-step targets:

$$y_{j,t} = \sum_{t'=t}^{t+N-1} \gamma^{t-t'} r_{j,t'} + \gamma^N \max_{a_{j,t+N}} Q_{\phi'}(s_{j,t+N}, a_{j,t+N}) \tag{2.7}$$

A multi-step variant of DQN is then defined by minimizing the alternative loss[16],

$$R_t^{(n)} + \gamma_t^{(n)} \max_{a'} q_{\bar{\theta}}^{-}(S_{t+n}, a') - q_\theta(S_t, A_t) \tag{2.8}$$

2.0.1.5 Noisy Nets

The one limitation of ϵ-greedy policy is many actions must be executed to collect the first reward.Noisy Nets proposed a noisy linear layer that combines a deterministic and noisy stream.Depending on the learning rate the network ignores to learn the noisy stream.

2.0.1.6 Integrated Agent:Rainbow

In the Rainbow architecture [16] tries combine the above six method stated above all together.Distributional loss (3) was replaced with a multi-step variant.The target distribution was constructed by contracting the value distribution in St+n according to the cumulative discount, and shifting it by the truncated n-step discounted return. multi-step distributional loss with double Q-learning by using the greedy action in St+n selected according to the online network as the bootstrap action at+n, and evaluating such action using the target network.
TODO ADD EQUATION:target distribution

3
Methods

In this section, we will explain the architecture of our auto-encoder and reinforcement learning algorithm. This includes a description of the environment and preprocessing in Section 3.0.1., the collection of training data for the auto-encoder and model architecture in section 3.0.2. and the training of the RL agent in Section 3.0.3.

3.0.1 Enviroment and Preprocessing

[!ADD MORE HERE AFTER NAP]
We perform a comprehensive evaluation of our proposed method on the Arcade Learning Environment (Bellemare et al., 2013), which is composed of 57 Atari games. The challenge is to deploy a single algorithm and architecture, with a fixed set of hyper-parameters, to learn to play all the games given embedded latent space representation of the environment from auto encoder and game rewards. This environment is very demanding because it is both comprised of a large number of highly diverse games and the observations are high-dimensional.

Working with raw Atari frames, which are 210 x 160 pixel pictures with a 128 color palette, is computationally expensive, therefore we do a basic preprocessing step to reduce the input dimensionality. The raw frames are preprocessed by down sampling to a 110 x 84 picture and transforming their RGB representation to gray-scale.Cropping an 84 x 84 rectangle of the image that nearly captures the playing area yields the final input representation to encoder part of the auto encoder.

3.0.2 Deep Auto-encoder and Model Architecture

[!ADD MORE HERE AFTER NAP]
The first step in the process is to collect data to train the auto-encoder. we run a data collection module to generate the 100000 frame for each stated under the result section.The raw images are transformed to tensors and then trained a variational autoencoder with the objective of re-constructing the original image fed to the network. The auto-encoder was trained for maximum 100 epochs.When reconstructing an image with a network bottleneck, the encoder is forced to compress the original image to a smaller dimensional vector in the latent space.
[By compressing the raw pixels environment to smaller dimensional vector in the latent space we aim to improve the training time it takes for the integrated Rl agent developed by [16] and the shift in the latent space representation and its impact

on the RL agent learning performance.We show this in more detail in the following sections.]

3.0.3 Training the RL Agent

[!ADD MORE HERE AFTER NAP]
In this paper we integrate auto encoder with integrated agent called Rainbow[12],the selection of this architecture was based it's ability to out perform all the previous architecture.our main focus was to experiment with the latent space representation of the environment.To address this we set up two experiment architectures one two step training and parallel training.

Two step Training: First,we train the auto encoder with the prepossessed data and the weights of the encoder are saved in file for training the agent.In this set up the auto encoder is not updated such that we have a static representation of the environment.
second,we train the integrated agent with the results from the auto encoder.

Parallel Training: In this set up the agent is trained using the dynamic state representation from the encoder as we train both the auto encoder and the integrated agent in parallel.This introduces stochasticity of the environment to the training and in real life control system we believe that this will set a new paradigm on the use RL.

4
Results

state the Experiments

4.0.1 Two Step Training

4.0.2 Parallel Training

5
Conclusion

You may consider to instead divide this chapter into discussion of the results and a summary.

5.1 Discussion

5.2 Conclusion

Bibliography

[1] [1] Sutton, R. S., Barto, A. G. (2018). Reinforcement learning: An introduction. MIT press.

[2] Mnih, Volodymyr and Kavukcuoglu, Koray and Silver, David and Graves, Alex and Antonoglou, Ioannis and Wierstra, Daan and Riedmiller, Martin.(2013). Playing Atari with Deep Reinforcement Learning.

[3] Mnih, V., Kavukcuoglu, K., Silver, D. et al.(2015). Human-level control through deep reinforcement learning. Nature 518, 529–533.

[4] Hado van Hasselt and Arthur Guez and David Silver.(2015).Deep Reinforcement Learning with Double Q-learning.

[5] van Hasselt.(2010).Double Q-learning. Advances in Neural Information Processing Systems, 23:2613–2621.

[6] Wang, Ziyu and Schaul, Tom and Hessel, Matteo and van Hasselt, Hado and Lanctot, Marc and de Freitas, Nando.2015.Dueling Network Architectures for Deep Reinforcement Learning.

[7] Moerland, Thomas M. and Broekens, Joost and Plaat, Aske and Jonker, Catholijn M.Model-based Reinforcement Learning: A Survey.(2020).

[8] Puterman, M. L. (2014). Markov Decision Processes.: Discrete Stochastic Dynamic Pro- gramming. John Wiley Sons.

[9] M.L. Littman, in International Encyclopedia of the Social Behavioral Sciences, 2001.

[10]Kidambi et al., 2020
[11] Yu et al., 2021
[12]Ignasi Clavera and Jonas Rothfuss and John Schulman and Yasuhiro Fujita and Tamim Asfour and Pieter Abbeel, Model-Based Reinforcement Learning via Meta-Policy Optimization,2018
[13]Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel. RLˆ2: Fast Reinforcement Learning via Slow Reinforcement Learning. 11 2016.

[14]. Sung, L. Zhang, T. Xiang, T. M. Hospedales, and Y. Yang. Learning to learn: Meta-critic networks for sample efficient learning. CoRR, abs/1706.09529, 2017.
[15]
[16]Hessel, Matteo and Modayil, Joseph and van Hasselt, Hado and Schaul, Tom and Ostrovski, Georg and Dabney, Will and Horgan, Dan and Piot, Bilal and Azar, Mohammad and Silver, David.2017.Rainbow: Combining Improvements in Deep Reinforcement Learning.
[17]Gerald Tesauro. Temporal difference learning and td-gammon. Communications of the ACM, 38(3):58–68, 1995.
[18]Long-Ji Lin. Reinforcement learning for robots using neural networks. Technical report, DTIC Document, 1993.
[19]Double DQN
[20]Prioritized replay [21]On the use of Deep Auto-encoders for Efficient Embedded Reinforcement Learning

A

Appendix 1

(UNFINISED)
//UPDATE THE FORMAT LATER

1. Agent: It is an assumed entity which performs actions in an environment to gain some reward.
2. Environment (e): A scenario that an agent has to face.anything the agent cannot change arbitrarily is considered to be part of the environment.
3. Reward (R): An immediate return given to an agent when he or she performs specific action or task.
4. State (s): State refers to the current situation returned by the environment.
5. Policy (): It is a strategy which applies by the agent to decide the next action based on the current state.
6. Value (V): It is expected long-term return with discount, as compared to the short-term reward.
7. Value Function: It specifies the value of a state that is the total amount of reward. It is an agent which should be expected beginning from that state.
8. Model of the environment: This mimics the behavior of the environment. It helps you to make inferences to be made and also determine how the environment will behave.
9. Model based methods: It is a method for solving reinforcement learning problems which use model-based methods.
10. Q value or action value (Q): Q value is quite similar to value. The only difference between the two is that it takes an additional parameter as a current action.