Project Plan: Reinforcement Learning in Latent Space

We, the authors

October 10, 2018

1 Context and Motivation

In **Reinforcement Learning** (RL), machine learning problems are modelled as a sequence of actions taken by an agent in some environment to maximize a total reward. Instead of learning from a dataset, the agent builds knowledge about the environment by exploring the effect of its behaviour. Since such interaction can be cost-intensive in real-world (or physical) applications, it is desirable to pre-train agents on a simulated task and afterwards generalize the obtained knowledge to the real task.

For this reason, recent research on reinforcement learning involves a lot of work on **transfer learning** (TL). In TL, an agent learns to do a *source task* and uses its knowledge in a before unseen *target task* to perform to a reasonable level with minimal additional training. In its more extreme forms, TL is known as one- or zero-shot-learning, where only a single training step is allowed in the target task - or none at all (Goodfellow, Bengio, and Courville, 2016). In RL, one of the key challenges of applying TL is inter-task alignment of states and actions. While some work tackles this issue with hand-crafted solutions (e.g. Taylor and Stone, 2007), it is desirable to develop methods to automatically map the tasks. Another approach is to learn in a common state (and possibly action) space, into which all tasks can be translated.

Deep Learning (DL) has been successfully applied to a variety of problems in machine learning research and got increasing attention over the last two decades (Goodfellow, Bengio, and Courville, 2016). While DL is applicable to classical problems such as regression and classification - and may henceforth be used as a policy learner in RL as well - it is particularly useful for learning representations in a latent space. For instance, convolutional neural networks can be used to break down visual input into features modelling higher-level information. Sequences of variable length, such as natural language, can be embedded using recurrent neural networks (Goldberg and Hirst, 2017). Another architecture for representation learning is the so called *autoencoder* (Hinton and Salakhutdinov, 2006b). These neural networks learn to first reduce the dimensionality of their inputs. They then reconstruct the original sample from the low dimensional representation. The low-dimensional representation in the middle layer often proofs to be useful as a representation for different applications as it encodes the relevant information of the input space into a latent space. Using DL, it may be possible to find state and action representations for multiple tasks that allow joint learning and generalization to unseen tasks.

Motivation This is useful for several reasons. As stated previously, it facilitates learning in costly tasks. Take, for example, the training of a robot playing soccer. If the agent needs to learn its behaviour entirely from physical play, the training process could only be run in real time. A simulation modelling the environment closely could be run much faster and, if done correctly, the learning is generalizable to the original task. Tasks that seem infeasible to train become solvable. Going further, knowledge gathered from playing soccer should be useful in other ball sports or even physical activities in general. If we aim to develop agents capable of performing any task in a given domain[1], we would expect it to use experience from one task in other tasks if they are at least partially applicable. For that, the different state spaces (and their representations) need to be abstracted into some latent space, as can be done with DL. In this latent space, all tasks may benefit from the experience in other tasks and only need to adjust to the specific requirements of the new situation. For instance, soccer and handball share the objective of bringing a ball into a goal as well as basic movement patterns and cooperation strategies. Though, they differ in the way the ball is handled.

Objective In this project, we aim to design a learning framework in which multiple RL tasks can be trained in the same latent space. The resulting knowledge should be generalizable to unseen tasks, whose training gets kick-started or at least sped up. An agent capable of playing multiple and partly unseen Atari games will be developed. It is hypothesized that Variational Autoencoders (VAE) can serve as a technique to convert the different states of different games into a latent space. The following research questions are posed:

- **A first Question:** Some blablabla and dideldum.
- **A second Question:** More blablabla and dideldum.
- **A third Question:** the most important blablabla and dideldum.

Hence, our contributions to the field are the following

- We
- Are
- Awesome

---

2 Social Impact

While reinforcement learning proved its effectiveness in diverse areas ranging from chemistry (Zhou, Li, and Zare, 2017) to games (Silver et al., 2017), solving complex problems typically require costly computational resources. This could be a factor blocking individuals or smaller organizations from utilizing such techniques. However, if knowledge learnt on one task can be reused on other related ones, this entrance barrier could be greatly lowered. Since this research project focuses on transferring knowledge across tasks, the proposed method, if proven successful, could make reinforcement learning techniques more accessible and usable.

A concern about the the topic of this project, however, could be that knowledge transferring is one of the first steps towards "general" artificial intelligence. Several prominent scholars and industry leaders warned about potential of reinforcement-learnt agents that can manipulate or control their reward signals (Russell, Dewey, and Tegmark, 2016). However, since existing research on transfer learning in reinforcement learning were generally validated on simple simulated physical games or Atari games, we do not consider this risk sufficiently realistic to be relevant.

[1] Such as sports or any class of activities that rely on a common ground of skills (e.g. motor skills).

3 Concepts and approach

Deep learning (DL) is a specific subfield of machine learning that aim to learn representations of the world from sets of high level features built by the composition of lower level ones. The main aspect of deep learning is that these set of features, called layers, are learned from raw data through general-purpose procedures and without the intervention of humans.

The algorithms that allow this kind of learning are the **artificial neural networks** which are a mathematical function that maps a set of inputs values to outputs values (Goodfellow, Bengio, and Courville, 2016). Such function is a composition of many simpler functions that provides several levels of abstractions.

Convolutional neural networks (CNNs) are a category of artificial neural networks designed to process "grid-like topology data" (Goodfellow, Bengio, and Courville, 2016) like images or sounds signals. Its architecture, that take advantage of the topology of the input, arrange the neurons in three dimensions: width, height and depth.

The name of this kind of neural networks comes from the operation in which they are based, the convolution, which is a special kind of linear operation. CNNs architecture is formed by four main layers:

- Convolutional layer.
- ReLu layer.
- Pooling layer.
- Classification layer.

The convolution layer is the main part of the CNNs, being in charge of detecting the features in the data by the use of filters. To improve the speed of the training, convolution layers are followed by the rectified linear units (ReLu) layer, that acts as a rectifier function.

The functionality of the pooling layer is reduce the spatial size of the representation with the objective of decrease the parameters in the network and the computation required. Finally, the last part of the network is the fully connected layer, which performs the classification task.

An **autoencoder** is neural networks trained to copy its input to the output and which is formed by two parts: the encoder, that converts the input to a dense and smaller representation and the decoder, that rebuild the input from the representation. The main application for autoencoders is the dimensionality reduction (Hinton and Salakhutdinov, 2006a), where the encoder learns to preserve the meaningful attributes of the input and generates a lower dimensional representation that is saved in the **latent space**.

One of the limitations of generic autoencoders is that the generated latent space is based in features of the data and may not be continuous. This lack of continuity works well for the replication of the input but produce unrealistic outputs in generative tasks or with inputs that has not been previously observed.

In order to create a continuous latent space, **Variational autoencoders (VAEs)** use probability distributions to describe each attribute, allowing interpolation and random sampling in the data. Moreover, VAEs are trained with **gradient based methods** which gives a better control over the latent space representation (Goodfellow, Bengio, and Courville, 2016) and generates representations with disentangled factors (Higgins et al., 2016).

Latent Space (LS) is the space in which information generated by the encoder lies. It contains the compressed representation of the input that can be used by the decoder to regenerate the data.

How will you be tackling/solving this?
Concepts:

- Existing Technologies
- Methods
- Terminology

Approach:

- What will you be doing with these technologies?
- How will you apply them?

4 Deliverables

What we want to deliver is an agent that we trained using reinforcement learning. The training environment that we will use is the OpenAI Gym (Brockman et al., 2016). This enables us to implement different approaches with varying difficulty. A relative easy approach will be to train the agent on sinusoidal tasks such as a cartpole task, a mountain car task or a pendulum task. For this, we will use the internal state representation at first, later, if possible, transforming these problems to a visual state representation.

A second deliverable will be an agent that is able to play Atari games using pixels as input. Although the OpenAI Gym framework also supports this, it will be harder to implement because we will need a lot more time to train an agent. This renders debugging very time consuming. Furthermore, none of us has experience with pixel based input data, which will make constructing the needed neural network difficult.

Since we want to use a latent space for transfer learning we will need to deliver a representation learner. This learner will be based on an autoencoder model, more precisely on a variational autoencoder. The states that were used for training the VAE should be encoded in the learned representation, which in turn could be reused for future work.

Most importantly, we will deliver a method for transfer learning and an implementation of it. We will show that it is possible to learn from previous tasks and use knowledge from those to achieve a better performance in another related task. We will show this using sinusoidal tasks as well as Atari games. These trained models will be available for demonstration purposes.

Furthermore, the results will be compared using different metrics, showing their different kind of strengths such as jumpstart, convergence and overall performance. Additionally, a qualitative analysis of the learned representation will be given. This analysis will investigate what kind of features are still present in the latent space and how an agent can learn from these representations.

5 Time Management

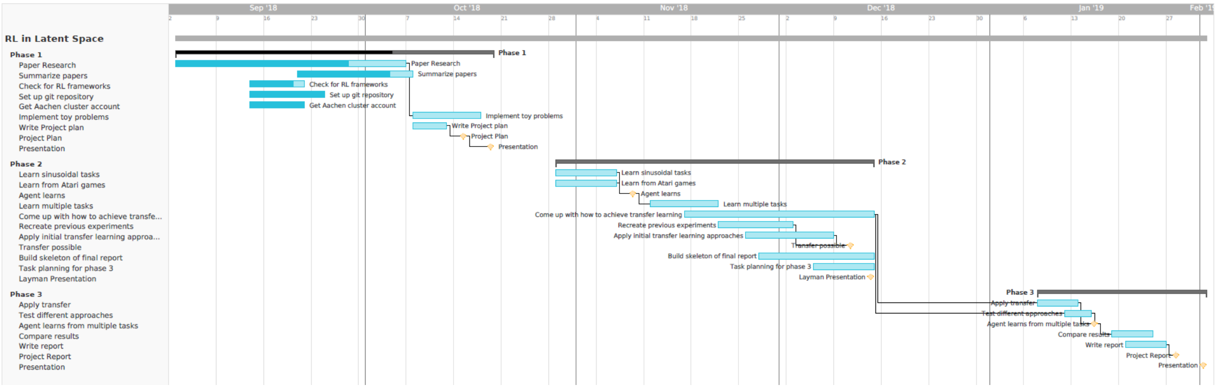
In order to tackle the project efficiently we dedicate a lot of time of the first phase into researching previous papers that already looked into transferring learning between different tasks. Furthermore, we will look for useful reinforcement learning frameworks which can be used to test our method. To verify that sufficient knowledge of these frameworks is present we will have to implement some toy problems which should be done by the end of phase 1.

In phase 2 we will split our group up into two groups to focus on different tasks that the agent will be trained on. One of these approaches is an easier implementation of learning as a fall-back if the other one does not work out. The main focus in this phase lies on coming up with a method to transfer learning between different tasks. Towards the end of this phase we plan to recreate existing transfer learning experiments and maybe even implementing our own first attempts. Furthermore, a skeleton for our final report that we have to hand in in phase 3 is drafted.

Phase 3 will then focus on applying and implementing our ideas to transfer learning. After achieving this, we will compare our results with already existing ones and start writing our final report.

5.1 Gantt Chart

To illustrate, we created a Gantt chart to show our task planning.



6 Risk Analysis

We distinguish between technical and organizational risks, where the former involve those related to the approach chosen to tackle the research problem, whereas the latter refer to those arising from the project team itself or related parties. The two types of risks are listed in table 6.1 and 6.2 respectively. In addition to the descriptions and contingency plans, justifications for the costs and likelihoods of the risks are provided in table 6.3.

6.1 Technical Risks

ID	Risk Description	Contingency Plan	Cost	Likelihood
1	The team cannot reproduce the results reported in selected papers on time.	Provide explanations, and substantiate with experiment results.	+	+++
2	The team cannot implement a working RL-based Atari-playing agent on time.	Focus on transfers across sinusoidal tasks (inverted pendulum, mountain car, cart pole) and explore a larger variety transfer methods.	+++	++
3	The proposed method does not bring improved performance (no effect or “negative transfer”) on the interested tasks pairs / groups.	Investigate the reasons of the failure to transfer. Compare with scenarios where transfer is successful.	++	+++

6.2 Organizational Risks

ID	Risk Description	Contingency Plan	Cost	Likelihood
4	One team member has to be absent from the team for long due to uncontrollable factors like severe illness or family circumstances.	Reduce the scope of the project. When in the phase of implementation, reduce the number of alternative models. When in the phase of experiments, reduce the number of experiments and prioritize those with the highest chance of yielding insightful results.	+++	+
5	Downtime in the Aachen computing cluster makes it impossible to run experiments as scheduled.	Rent GPU-enabled virtual machines on Microsoft Azure or Amazon Web Services (AWS) with the free student credits.	++	+

6.3 Cost and Likelihood Explanation

ID	Cost Explanation	Likelihood Explanation
1	+: Reproduced experiments results, regardless whether the same as in the original publications, can still be used as a comparison with our experiment results.	+++: It is likely that not all hyper-parameters or initialization conditions are explained in the publication, making the experiment results difficult to reproduce.
2	+++: With a large number of experiment scenarios (Atari games) ruled out, the experiments will be limited to sinusoidal tasks.	++: Although no team member had experience with game control based on raw pixels, there are existing tutorials and Git repositories that accomplished this task.
3	++: Negative results without well-interpreted results would render this research unsuccessful.	+++: It may be challenging to apply suitable techniques to achieve transfer across our experiment tasks, as transfer learning in reinforcement learning is still an area with open issues.
4	+++: The absence would cause a 20% drop in manpower.	+: Factors causing the long-term absence of a teammate are rare.
5	++: Experiments results may be delayed, impacting project progress.	+: Long-lasting downtime of the Aachen cluster is unlikely.

References

Brockman, Greg et al. (2016). *OpenAI Gym*. eprint: `arXiv:1606.01540`.

Goldberg, Yoav and Graeme Hirst (2017). *Neural Network Methods in Natural Language Processing*. Morgan & Claypool Publishers. ISBN: 1627052984, 9781627052986.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press.

Higgins, I. et al. (June 2016). "Early Visual Concept Learning with Unsupervised Deep Learning". In: *ArXiv e-prints*. arXiv: `1606.05579 [stat.ML]`.

Hinton, G. E. and R. R. Salakhutdinov (2006a). "Reducing the Dimensionality of Data with Neural Networks". In: *Science* 313.5786, pp. 504–507. ISSN: 0036-8075. DOI: `10.1126/science.1127647`. eprint: `http://science.sciencemag.org/content/313/5786/504.full.pdf`. URL: `http://science.sciencemag.org/content/313/5786/504`.

Hinton, Geoffrey E and Ruslan R Salakhutdinov (2006b). "Reducing the dimensionality of data with neural networks". In: *science* 313.5786, pp. 504–507.

Russell, Stuart J., Daniel Dewey, and Max Tegmark (2016). “Research Priorities for Robust and Beneficial Artificial Intelligence”. In: *CoRR* abs/1602.03506. arXiv: 1602.03506. URL: http://arxiv.org/abs/1602.03506.

Silver, David et al. (Oct. 2017). “Mastering the game of Go without human knowledge”. In: *Nature* 550.7676, p. 354. ISSN: 1476-4687. DOI: 10.1038/nature24270. URL: http:https://doi.org/10.1038/nature24270.

Taylor, Matthew E. and Peter Stone (2007). “Cross-domain Transfer for Reinforcement Learning”. In: *Proceedings of the 24th International Conference on Machine Learning*. ICML ’07. Corvalis, Oregon, USA: ACM, pp. 879–886. ISBN: 978-1-59593-793-3. DOI: 10.1145/1273496.1273607. URL: http://doi.acm.org/10.1145/1273496.1273607.

Zhou, Zhenpeng, Xiaocheng Li, and Richard N. Zare (2017). “Optimizing Chemical Reactions with Deep Reinforcement Learning”. In: *ACS Central Science* 3.12. PMID: 29296675, pp. 1337–1344. DOI: 10.1021/acscentsci.7b00492. eprint: https://doi.org/10.1021/acscentsci.7b00492. URL: https://doi.org/10.1021/acscentsci.7b00492.