* **Background research report (*1-2000 words*)**
* Your literature survey:
* justifies that your project is worth doing
* sets your work in context, critically evaluating past and current research
* provides a starting point for future work
* What inspires your work?
* This will most likely be a survey of similar projects, research articles, applications relevant to your project
* Engage critically with existing work.
* What are the boundaries, limitations, contradictions, developing areas, and dead ends of other work?
* Go beyond mere description by offering opinions to what is written
* How have other projects evaluated their work?
* Use this section to demonstrate an awareness of how your project fits within the context of the field(s) you are studying
* Provide consistently formatted references for all related work you discuss
* **Overall essentials**
* Implicitly, a good project proposal should:
* Introduce the subject area
* Highlight current research
* Identify a gap in current research
* Show how your work fills that gap
* Identify potential risks and solutions with the project

Who is it for – google algorithms and common use cases – who is it for

**Enhancing Generalisation in Reinforcement Learning Agents Through Procedural Content Generation and Adaptive Training Techniques**

**Introduction**

In the field of reinforcement learning (RL), significant challenges remain in developing agents capable of handling real-world scenarios, where environments are often diverse, dynamic, and unpredictable. One of the major obstacles in achieving this goal is achieving effective generalisation, where an agent is defined as one that can generalise well to novel, unseen situations.

This project focuses on improving an agent's generalisation capability, primarily through the use of **procedural content generation (PCG)**, which algorithmically generates diverse training environments, or ‘levels’ . By training an agent on a wide range of procedurally generated environments, we aim to improve its ability to adapt to new, unseen scenarios, thus mitigating overfitting to the training environment.   
Techniques used for training will also include the addition of CNN layers, regularisation, environmental stochasticity, and simplified curriculum learning by way of modifying the PCG process to allow for difficulty scaling.

The project uses a Dynamic Hierarchical Reinforcement Learning (dHRL) algorithm, which decomposes decision-making into subtasks executed in single timesteps. Experiments will take place in a grid-based, dungeon-crawler-like game with a discrete action and state space, providing a simple yet effective testing platform.  
To analyse the impact of generalisation techniques, multiple agents will be trained upon increasing diversity of PCG levels, and we will compare the performance of agents trained with generalisation methods against a baseline agent trained without them. Performance will be measured using metrics such as average time steps taken, percentage of levels solved, and final score.  
This evaluation aims to determine the effectiveness of these techniques in enabling agents to generalise and perform well in new and varied environments.

**1. Background**

Machine learning, and the subfield of Reinforcement Learning, exist under the umbrella of Artificial Intelligence, and has been capturing minds since the 1950’s.   
The field has been through booms and ‘winters’ over the years, and in the 1990’s with the development of more powerful computers came a resurgence of machine learning.   
  
In 1997, Tom Mitchell defined the concept of Machine Learning with “A computer program is said to learn from experience E with respect to some task T and some performance measure P , if its performance on T , as measured by P , improves with experience E .” [5]  
This granular definition shows that the essence of machine learning is in improving upon a particular task, through experience and evaluation of performance.  
Looking back at Tom Mitchell’s definition, a typical machine learning application might be to classify a class from an image or predict a numerical value, and receive a value of error (P) for their prediction (E) to then improve upon the task (T).

***1.1 Reinforcement Learning (RL)***  
Reinforcement learning, a subfield of Machine Learning, It has renewed popularity in the last decade due to the increased accessibility of deep learning.  
RL is both adaptive, and deals with long-term consequence, making it of interest for solving complex real world problems. [7]  
A reinforcement learning (RL) agent interacts with an environment by making observations, taking actions, and receiving rewards or penalties. Its goal is to learn behaviours that maximise expected rewards and minimise penalties over time. [6]

An RL system consists of four main components: a policy to determine actions, a reward signal, a value function, and optionally, a model. [7]

Roughly speaking, a policy is a mapping from perceived states of the environment, to actions to be taken within these states. This mapping defines the behaviour of our agent.  
The reward signal can be thought of as the goal within a problem. At each timestep as our agent acts within its environment it will receive back from the environment the reward for it’s action, positive, negative, or null – and our agents seek to maximise reward.  
A value function describes a function that assesses experiences the agents garner through acting within the environment, and calculates the perceived value of a state, based on an estimation of future reward to come.  
Finally, a model of the environment describes something akin to a simulation of the environment, or a way to infer behaviour of the environment. It allows an agent a degree of planning within its environment rather than relying solely on trial and error.  
Methods for reinforcement learning using models are known as model-based methods, while those without them are known as model-free. [7]  
We will be focusing on a model-free method for the purposes of this project.

* 1. ***Markov Decision Process (MDP)***

A Markov decision process or MDP is a mathematical framework used to model a decision making problem in which the Markov property is upheld, and outcomes are partly random, partly controlled. The Markov property is present if future state can be predicted based on the observation of the current state, more specifically no knowledge of previous state is needed. [7]  
MDP’s feature a set of states, actions, rewards dependent on state and action, and transition probabilities for states and actions. [24]

* 1. ***Q-Learning***

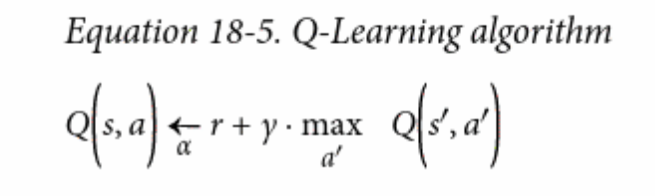
Q-Learning is an important algorithm within reinforcement learning. It provides a method for converging on an optimal policy within a system defined by an MDP.   
Using the Q-Value Iteration Algorithm defined by Bellman in the 1950’s [5], one can estimate the optimal state-action values, otherwise known as Quality Values, or Q-Values, which is the sum of the discounted future rewards on average after an agent reaches state *s* and takes action *α,* before it has seen the outcome of the action and assuming it acts optimally afterwards. [5]  
In the following equations, *γ* is equal to a discount factor, *Q being* the Q-Value for a state-action, *R* for the set of rewards, *s* for states, *α* for actions, and *T* being the transition function for states and actions.

A math equation with black text

Description automatically generated

*Figure taken from A. Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. [5]*

Knowing the Q-Values, once can easily find the optimal policy, simply choose the action with the highest Q-Value.  
  
Q-Learning uses a variation of this algorithm, based on the assumption that Q-Values are initially unknown, and seeks to update them by trial and error as an agent gains experience within the environment. Once it has trained accurate enough Q-Value estimates, the optimal policy is a greedy one, choosing the action with the maximum Q-Value. [5]



*Figure taken from A. Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. [5]*

Q-Learning will converge, but given it does not know transition probabilities or rewards, it can take much hyperparameter tuning, and many iterations to find optimal Q-Values.  
Q-Learning is known as an *off-policy* algorithm, as the policy being trained does not need to be the one being executed – as opposed to a policy gradient algorithm, which is *on-policy*, altering the same policy that is in execution.

* 1. ***Exploration vs Exploitation***

We can make use of the off-policy nature of Q-Learning to aid the training process by using an exploratory policy in execution, rather than one that is simply random. [5]  
An example of this would be the Boltzmann (or softmax) exploration algorithm, in which actions with high expected rewards have a higher chance to be selected.  
The figure below describes the exploration strategy, finding the probability of taking an action.

A black and white math equation

Description automatically generated

*Figure taken from R. Niel and M. A. Wiering, ‘Hierarchical Reinforcement Learning for Playing a Dynamic Dungeon Crawler Game’ [20]*

*T* refers to the temperature, in which higher temperature reduces the difference in chance, while lower temperature favours expected reward. *P* refers to the probability of taking an action in that state, and *A* is the set of actions. [20]

* 1. ***Deep Q-Network (DQN)***

When the possible space of states and actions becomes too large, Q-Learning starts to falter. Consider the relatively simple game of Ms Pacman, with a number of ghosts, the player, and over 150 pellets to eat.   
The number of possible states based on what pellets have been eaten is 2150 already, and when adding possible positions of the ghosts and the player it becomes greater than the number of atoms on the planet. Keeping estimates of Q-Values for every possible state is not currently feasible. [5]

We can adapt the approach to instead approximate Q-Values for state-actions, rather than rely on exact values.  
One way of doing this is through the use of neural networks to find a function that approximates these values using a number of parameters (given by the parameter vector θ). [5] This can be extended to deeper networks, and there has been notable success with the added use of convolutional layers within a network, presumably helpful in determining and extracting features for use within these networks. [18]

A black text on a white background

Description automatically generated

*Figure taken from A. Géron, Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. [5]*

***1.6 HRL and d(HRL)***

HRL stands for Hierarchical Reinforcement Learning. This is a term for algorithms that seek to decompose a Markov Decision Process into a hierarchy of smaller MDP’s.  
It is primarily a divide and conquer strategy, where the bottom of the hierarchy contains simple actions, while the parents contain higher level abstractions of behaviour.

We will be implementing dHRL, proposed by R. Niel and M. A. Wiering [20], where similar to other HRL systems, the problem is decomposed into smaller MDP’s. It consists of a root behaviour, which can then choose from other sub-behaviours. The root behaviour and sub behaviours have their own reward functions, independent of other behaviours, and can either execute primitive actions or delegate to other sub-behaviours until an action is chosen. Commonly in HRL systems, subtasks can contain arbitrarily long sequences of actions, longer than a normal timestep. A key difference with dHRL is that each primitive action in sub behaviours takes only a single timestep. This allows for dynamic reaction to the environment and less getting stuck in one task, additionally allowing increased focus on a single goal within training.

**2. Overview of Generalisation**

When we talk about the ability of an agent to generalise, we mean the ability to generalise learning from previously experienced states [7] to ones that are novel and unseen. [2]

Reinforcement learning agents are prone to overfitting their environments, [13] and this leads to an inability to deal with new situations. This is especially important to tackle, as the real world is dynamic and complex, and we need agents able to deal with diverse situations and not fail catastrophically in the face of the unseen.

The scope of this this review will be that of modern techniques used to aid in the improvement and evaluation of generalisation capability of a software agent.  
These techniques vary from the augmentation of data and regularisation of models, to training across varied and diverse procedurally generated environments in an attempt to train underlying behaviours rather than overfit one environment.

Generalisation is similar in concept to transfer learning and domain adaptation, which is a specific type of transfer learning.  
In the case of transfer learning, the goal is to leverage knowledge from a previously learned task to learn a new target task quicker than if there was no transfer at all. [14] In this case, we are taking knowledge from one task to a different task, for example from one game to another.  
Domain adaptation is a specific type of transfer learning, where the task may remain relatively the same, but we are dealing with domain shifts, such as from a simulation to the real world, and aiming to achieve meaningful generalisation to this other setting. [15]

In the case of generalisation, we are staying within the same setting, and looking for a robust performance on unseen instances of the same task.

This task is not an easy one, and there is a great chance that a model will still overfit to a considerable, even when using techniques to combat this, as in the case of the CoinRun benchmark (Cobbe et al. 2019) [16].

**3. Key techniques of improving generalisation in RL**

When looking to teach our an agent to generalise, we want to teach the underlying shapes of the problem, the behaviour that we’d like to see across a range of environments, and to that end there are a few popular techniques seen in modern research.

***3.1 Data Augmentation***

Data augmentation is process of augmenting the input data, so that there is an artificial variance in data. This is commonly used with CNN’s, and augmenting visual data for reducing overfitting.

In the case of passing image data , you may try randomizing colours and textures [15]. This level of variation can help the agent to avoid overfitting when detecting visual features in new environments – both in settings within the same task and when adapting to new domains.

The same idea can be and is applied to good effect within environments working with physics simulations, by varying the dynamics of an task, we can better bridge the reality gap and train policies capable of adapting to unfamiliar dynamics when applied to real world situations. [19]

Ultimately the concept is “if the variability in simulation is significant enough, models trained in simulation will generalize to the real world with no additional training.” [15]

***3.2 PCG (Procedurally Generated Content)***

We can also randomise the setting within a domain by way of **PCG** (Procedurally Generated Content. In doing so we aim do avoid overfitting to any one environment, by providing randomised and varied layouts of ‘levels’ for an agent to train on – say in the case of a game.  
This is shown to correlate with positive effects on the agent’s capability to generalise to new settings within the same domain [16].  
  
Going one step further than this, there are instances of taking the **PCG** technique further, with one example being **PPCG** (Progressive Procedurally Generated Content) [17], wherein a ‘level’ is not just procedurally generated, but the difficulty of the of the level is tuned to the performance of the agent. A “level generator will initially create easy levels and progressively increase the difficulty as the agent learns.”, [17] the results of experiments implement this showed that “dynamically adapting the level difficulty during training allows the agent to solve more complex levels than training it on the most difficult levels directly.” [17]  
  
In a similar vein, there are algorithms designed to feature both a ‘teacher’ and ‘student’, such as the ACCEL algorithm [3], which is inspired on the POET algorithm [21]. In these the environments are similarly paired with an student agent, and environments are dynamically edited to present increasing challenges, helping to teach complex behaviours. In the case of ACCEL, Kruskal’s algorithm was used to create mazes of increasing complexity. [3]

Notably, it has been posited that PCG, while a popular technique to help generalise an agent within a domain, it is recommended that it should not be solely relied upon, and environments should also include other factors of controllable variation. [1]

***3.3 Environmental Stochasticity***

Adding stochasticity into the agent’s actions within an environment can be helpful in general when training an agent – using probability distribution to sometimes not take a preferred action and instead explore other possible options can lead to significantly improved performance down the line [20], both in performance of models trained in one environment or reducing generalisation error, with a notable example being the epsilon-greedy algorithm. [16]

***3.4 Batch Normalisation and Regularisation***

When working with neural networks and deep reinforcement learning, there are additional techniques employed that take inspiration from the realm of supervised machine learning. Regularisation in the form of Dropout and L2 regularisation saw some success within the CoinRun benchmark experiments (Cobbe et all 2019) [16].

However one of the findings within this paper was that while these techniques some saw success, they worked best when used together, and may still be addressing the underlying causes of poor generalisation.  
Batch normalisation is known to have a substantial regularising effect within the ream of supervised learning [12], and was also employed to great effect within the CoinRun environment, normalising between CNN layers. [16]

**3.5 Convolutional Neural Network (CNN)**

Convolutional Neural Networks have been very successful in practical application, extracting meaningful features from data that exists in a grid-like format, notably visual input. [23]  
They have seen success in game-based RL agent applications base on these strengths [23][16], and understandably so.

This is of note for our intentions – looking at the training of a DQN with a simple CNN. Our aim to generalise across environments, where the layouts of the levels feature paths, shapes, that can be extracted as features, passing such features into training rather than an entire grid of cells.

These features possibly pertaining to the shape of available paths, distance to nearby walls or enemies could allow our DQN to train a policy that is able to better abstract the shape of the level and avoid overfitting to specific layouts.

**4. Contribution**

There has been, and continues to be a lot of research in the field of RL, tackling some of the major obstacles on the route to real world applicable agents, including that of generalisation. One paper of particular interest to us is R. Niel’s and M. A. Wiering’s, ‘Hierarchical Reinforcement Learning for Playing a Dynamic Dungeon Crawler Game’, detailing their novel architecture for a dHRL algorithm, which saw improved performance over an agent implemented with hierarchical MaxQ algorithm.  
In their paper they note that it would be of interest to see further work on observing if the generalisation capability of the agent could be improved through use of multiple levels to train on, chosen randomly each epoch.  
This is where the focus of this project will lie. We will attempt to reimplement the dHRL algorithm and train multiple agents in a PCG environment with increasing diversity of PCG levels, as well as utilising various techniques , such as batch normalisation, regularisation, simplified curriculum learning, and the addition of CNN’s.

We will design an a simplified dungeon crawler environment within python, utilising the wealth of utility and resources for ML available, namely TensorFlow, Keras, and gymnasium (formerly OpenAI gym) as they are well supported and industry standard.

Using a modified Kruskal’s algorithm, is it possible to create mazes, both perfect [3] and non-perfect [25]. Further modification will be used to alter production of levels and creating more sparse environments, and increase difficulty artificially when developing for a simplified curriculum learning approach.

By doing this, we hope to first follow on from the work of R. Niel and M. A. Wiering and provide some level of insight into the effect training upon diverse environments can have with a dHRL agent. Secondly we will further this by gauging the effect of combining multiple common generalisation techniques.

IDENTIFYING A GAP – Which algorithms and techniques?  
What has been done:  
List papers:

1. coinrun
2. PCG / PPCG : we show that this can lead to overfitting on a higher level, such as the distribution of generated levels presented during training.
3. ACCEL
4. Scan ZSG
5. Stochasticity exploration boltzmann

What are the limitations faced?

1. Coinrun – large scale timesteps, still overfitting despite lots of stuff
2. Difficulty in finding the right reward schemes, hyper parameters
3. PCG can lead to other types of overfitting

Dead ends? Boundaries? PCG is not a cure-all solution alone (ZSG) paper Developing areas? ACCEL, SCAN ZSG, CRL

What is my opinion of what has been written

How were those projects evaluated

dHRL look at total score, coin run looks at levels solved

generalisation error

timesteps avg

**What is the gap I will focus on:**  
dHRL recreation with generalisation, compared with techniques on a simple DQN

**How does this fit with the current field?**

dHRL is a novel algorithm

What is MaxQ

Why dHRL

dHRL has not been tested with generalisation

comparison to a DQN and other methods of ZSG

**Identify potential risks and pitfalls with my work**

I have no clue what im doing

Computational limits perhaps

My brains computational limits also

Overfitting despite generalisation techniques

Reproducibility of a novel algorithm  
  
REFERENCES:

**[1][2]** R. Kirk, A. Zhang, E. Grefenstette, and T. Rocktäschel, ‘A Survey of Zero-shot Generalisation in Deep Reinforcement Learning’, *jair*, vol. 76, pp. 201–264, Jan. 2023, doi: [10.1613/jair.1.14174](https://doi.org/10.1613/jair.1.14174).  
**[3]** A. A. J. P.-H. O. U. M. J. UCL, M. A. M. D. U. B. M. S. UCL, M. A. J. F. O. U. E. G. UCL, M. A. T. R. UCL, M. A. \*Equal contribution P. March 3, and 2022 ARXIV Full paper, ‘Evolving Curricula’. Accessed: Nov. 08, 2024. [Online]. Available:<https://accelagent.github.io>

**[4]** ‘Machine learning, explained | MIT Sloan’. Accessed: Nov. 13, 2024. [Online]. Available: <https://mitsloan.mit.edu/ideas-made-to-matter/machine-learning-explained>

**[5][6]** A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. Sebastopol, UNITED STATES: O’Reilly Media, Incorporated, 2019. Accessed: Nov. 13, 2024. [Online]. Available: <http://ebookcentral.proquest.com/lib/goldsmiths/detail.action?docID=5892320>

**[7]** R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, Massachusetts London, England: The MIT Press, 2015. Accessed: Nov. 13, 2024. [Online]. Available: <https://web.stanford.edu/class/psych209/Readings/SuttonBartoIPRLBook2ndEd.pdf>

**[8]** OpenAI *et al.*, ‘Dota 2 with Large Scale Deep Reinforcement Learning’, Dec. 13, 2019, *arXiv*: arXiv:1912.06680. doi: [10.48550/arXiv.1912.06680](https://doi.org/10.48550/arXiv.1912.06680).

**[9]** D. Silver *et al.*, ‘Mastering the game of Go with deep neural networks and tree search’, *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016, doi: [10.1038/nature16961](https://doi.org/10.1038/nature16961).

**[10]** S. Wang, D. Jia, and X. Weng, ‘Deep Reinforcement Learning for Autonomous Driving’, May 19, 2019, *arXiv*: arXiv:1811.11329. doi: [10.48550/arXiv.1811.11329](https://doi.org/10.48550/arXiv.1811.11329).

**[11]** J. Levinson et al., ‘Towards fully autonomous driving: Systems and algorithms’, in 2011 IEEE Intelligent Vehicles Symposium (IV), Jun. 2011, pp. 163–168. doi: 10.1109/IVS.2011.5940562.

**[12]** S. Ioffe and C. Szegedy, ‘Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift’, Mar. 02, 2015, *arXiv*: arXiv:1502.03167. doi: [10.48550/arXiv.1502.03167](https://doi.org/10.48550/arXiv.1502.03167).

**[13]** C. Zhao, O. Sigaud, F. Stulp, and T. M. Hospedales, ‘Investigating Generalisation in Continuous Deep Reinforcement Learning’, Feb. 20, 2019, *arXiv*: arXiv:1902.07015. doi: [10.48550/arXiv.1902.07015](https://doi.org/10.48550/arXiv.1902.07015).

**[14]** M. E. Taylor and P. Stone, ‘Transfer Learning for Reinforcement Learning Domains: A Survey’, *Journal of Machine Learning Research*, vol. 10, no. 56, pp. 1633–1685, 2009.

**[15]** J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, ‘Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World’, Mar. 20, 2017, *arXiv*: arXiv:1703.06907. Accessed: Nov. 13, 2024. [Online]. Available: <http://arxiv.org/abs/1703.06907>

**[16]** K. Cobbe, O. Klimov, C. Hesse, T. Kim, and J. Schulman, ‘Quantifying Generalization in Reinforcement Learning’, Jul. 14, 2019, *arXiv*: arXiv:1812.02341. Accessed: Nov. 10, 2024. [Online]. Available: <http://arxiv.org/abs/1812.02341>

**[17]** N. Justesen, R. R. Torrado, P. Bontrager, A. Khalifa, J. Togelius, and S. Risi, ‘Illuminating Generalization in Deep Reinforcement Learning through Procedural Level Generation’, Nov. 29, 2018, *arXiv*: arXiv:1806.10729. doi: [10.48550/arXiv.1806.10729](https://doi.org/10.48550/arXiv.1806.10729).

**[18]** V. Mnih *et al.*, ‘Human-level control through deep reinforcement learning’, *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015, doi: [10.1038/nature14236](https://doi.org/10.1038/nature14236).

**[19]** X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, ‘Sim-to-Real Transfer of Robotic Control with Dynamics Randomization’, Mar. 03, 2018, *arXiv*: arXiv:1710.06537. doi: [10.48550/arXiv.1710.06537](https://doi.org/10.48550/arXiv.1710.06537).

**[20]** R. Niel and M. A. Wiering, ‘Hierarchical Reinforcement Learning for Playing a Dynamic Dungeon Crawler Game’, in *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, Bangalore, India: IEEE, Nov. 2018, pp. 1159–1166. doi: [10.1109/SSCI.2018.8628914](https://doi.org/10.1109/SSCI.2018.8628914).

**[21]** R. Wang, J. Lehman, J. Clune, and K. O. Stanley, ‘Paired Open-Ended Trailblazer (POET): Endlessly Generating Increasingly Complex and Diverse Learning Environments and Their Solutions’, Feb. 21, 2019, *arXiv*: arXiv:1901.01753. doi: [10.48550/arXiv.1901.01753](https://doi.org/10.48550/arXiv.1901.01753).

**[22]** V. Mnih *et al.*, ‘Human-level control through deep reinforcement learning’, *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015, doi: [10.1038/nature14236](https://doi.org/10.1038/nature14236).

**[23]** Ian Goodfellow and Yoshua Bengio and Aaron Courville, *Deep Learning*. MIT Press, 2016. Accessed: Nov. 14, 2024. [Online]. Available: <http://www.deeplearningbook.org>

**[24]** M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, 1st ed. USA: John Wiley & Sons, Inc., 1994.

**[25]** M. Ihsan, D. Suhaimi, M. Ramli, S. M. Yuni, and I. Maulidi, ‘Non-perfect maze generation using Kruskal algorithm’, *J. Nat.*, vol. 21, no. 1, Feb. 2021, doi: [10.24815/jn.v21i1.18840](https://doi.org/10.24815/jn.v21i1.18840).

**Introduction**

What you wanna do,

Motivation

Project is about this

Main contribution is to create this

**Background**

What is rl

Q learning

dHRL

Has been done

**Introduction**

Very brief intro

What we gonna focus on

Main contribution is to create this

**Background**

What is ml / rl

Q learning

Deep Q N

dHRL

CNN

Generalisation

**Generalisation deep dive**

**What is missing /**

Has been done

LIT REVIEW PAPER NOTES

Prime candidate, a novel dHRL system.

Paper = Hierarchical Reinforcement Learning for Playing a Dynamic Dungeon Crawler Game

<https://ieeexplore.ieee.org/document/8628914>

**Paper looking at the performance of a dHRL algorithm vs MaxQ-Q on a simple dungeon crawler setting.  
Mentions future work ideas of generalisation.**

Algorithm: dHRL, maxQ-Q

Training techniques: Boltzmann exploration, MLPs

“During training both start with a temperature of 4 for the Boltzmann exploration algorithm which is multiplied by 0.98 after each game, to a minimum of 0.1. Their neural network learning rates are initialised slightly differently, for MaxQ-Q it starts at 0.001 while for dHRL it starts at 0.0005. Both have a decay after each game of 0.995.”  
“We found that when a greedy policy is used during testing, both AIs sometimes got stuck in small loops of states. When the Boltzmann exploration algorithm is used with a low temperature, it significantly improves performance for both systems. Hence both use the Boltzmann algorithm with a temperature of 0.1 during performance tests.”Evaluation: Total score from task, standard error Score / Epochs

Avg Score / Mean error/ Win rate

Paper 2 =Quantifying Generalization in Reinforcement Learning

<https://arxiv.org/pdf/1812.02341>

**OpenAI paper looking at procedural level generation and the generalisation error in a 2d platformer Coinrun.**

**It talks about the surprising level of generalisation error, techniques to measure this level of overfitting in generalisation, and ways to reduce it through convolutional architectures and regularisation.**

Algorithm: Nature-CNN, IMPALA-CNN, PPO

Training techniques: PCG, Dropout, L2 Regularisation, Data Augmentation, Stochasticity (environmental through epsilon greedy action selection, Batch Normalization

Notes: “In previous work, epsilon-greedy action selection has been used both as a means to encourage exploration and as a theoretical safeguard against overfitting (Bellemare et al., 2012; Mnih et al., 2013).”

“We find that combining data augmentation, batch normalization, and L2 regularization yields slightly better test time performance than using any one of them individually.”

“We have observed the surprising extent to which agents can overfit to a fixed training set. Using the procedurally generated CoinRun environment, we can precisely quantify such overfitting. With this metric, we can better evaluate key architectural and algorithmic decisions. We believe that the lessons learned from this environment will apply in more complex settings, and we hope to use this benchmark, and others like it, to iterate towards more generalizable agents.”

ON PCG “Mazes are generated using Kruskal’s algorithm (Kruskal, 1956).”

ON L2 and DO “They turn to supervised learning for inspiration, finding that both L2 regularization and dropout can help agents learn more generalizable features.”

Evaluation: Generalisation error through Levels Solved% of train vs test on number of training levels, levels solved% of certain levels solved based on number of timesteps

Paper3= NETWORK RANDOMIZATION:

A SIMPLE TECHNIQUE FOR GENERALIZATION

IN DEEP REINFORCEMENT LEARNING <https://arxiv.org/pdf/1910.05396>

**This paper talks about improving generalisation capabilities through the addition of a CNN to randomise the rendering inputs to a DRL to help with generalisation, and sees success.**

Published as a conference paper at ICLR 2020

Algorithm: PPO, IMPALA-CNN

Training Techniques: CNN Network Randomisation, Dropout, L2 Regularisation, Data augmentation

Notes: **REFERENCES** “However, it has been evidenced in recent years that deep RL agents of-

ten struggle to generalize to new environments, even when semantically similar to trained agents

(Farebrother et al., 2018; Zhang et al., 2018b; Gamrian & Goldberg, 2019; Cobbe et al., 2019). For

example, RL agents that learned a near-optimal policy for training levels in a video game fail to

perform accurately in unseen levels (Cobbe et al., 2019), while a human can seamlessly generalize

across similar tasks. Namely, RL agents often overfit to training environments, thus the lack of gen-

eralization ability makes them unreliable in several applications, such as health care (Chakraborty

& Murphy, 2014) and finance (Deng et al., 2016)”

**Failure of agents to generalise on small visual changes** : “We also found that RL agents completely fail

due to small visual changes1 because it is challenging to learn generalizable representations from

high-dimensional input observations, such as images.”  
**Characterisation of Generalisation**: “The generalization of RL agents can be characterized by visual changes (Cobbe et al., 2019; Gam-

rian & Goldberg, 2019), different dynamics (Packer et al., 2018), and various structures (Beattie

et al., 2016; Wang et al., 2016).”

**On methods of improving generalisation:** “To improve generalization, several strategies, such as regularization (Farebrother et al., 2018; Zhang

et al., 2018b; Cobbe et al., 2019) and data augmentation (Tobin et al., 2017; Ren et al., 2019),

have been proposed in the literature (see Section 2 for further details). In particular, Tobin et al.

(2017) showed that training RL agents in various environments generated by randomizing rendering

in a simulator improves the generalization performance, leading to a better performance in real

environments.”

**CNN Network Randomisation:** “Our main idea is to

utilize random (convolutional) networks to generate randomized inputs (see Figure 1(a)), and train

RL agents (or their policy) by feeding them into the networks. Specifically, by re-initializing the

parameters of random networks at every iteration, the agents are encouraged to be trained under a

broad range of perturbed low-level features, e.g., various textures, colors, or shapes.”

Paper 4= **This paper looks at generalisation and overfitting using procedural level generation for 4 different games.**

Illuminating Generalization in Deep Reinforcement Learning

through Procedural Level Generation

<https://rlg.mlanctot.info/papers/AAAI19-RLG-Paper22.pdf>

Algorithms: A2C

Training Techniques: **GVG AI (environment),** **Progressive PCG, RMS Optimizer**

Notes: **On architecture of their A2C** “The neural networks in this paper have the same architecture originally used from Mnih et al. (Mnih et al. 2016) with three convolutional layers and a single fully-connected layer.”

**On curricula or increasing difficult generation of content** “Addi-

tionally, it is possible to achieve better performance with less

data by manipulating the difficulty of the levels in response

to the performance of the agent.”

“Progressive PCG,

where the difficulty of levels/tasks is increased gradually to

match the agent’s performance. While similar techniques of

increasing difficulty have been used before, they have not

been combined with a PCG-based approach in which agents

are evaluated on a completely new level every time a new

episode begins.”

**ON PPCG implementation** “In the PPCG implementation in this paper, levels are initially created with the lowest difficulty of 0. If the agent wins an episode, the difficulty will be incremented such that future levels during training become harder. The difficulty is increased by α for a win and decreased by the same amount for a loss.”

Evaluation: Score/Steps/Difficulty

Max score vs random baseline score vs actual score

Paper 5: **Deep Reinforcement Learning for General Video Game AI**

<https://ar5iv.labs.arxiv.org/html/1806.02448>

Algorithms: DQN, Dueling DQN, A2C

Training Techniques: GVGAI (environment)

Notes: **On gvgai** “The General Video Game AI (GVGAI) competitions and framework were created with the express purpose of providing a versatile general AI benchmark *[*[*3*](https://ar5iv.labs.arxiv.org/html/1806.02448#bib.bib3)*,* [*4*](https://ar5iv.labs.arxiv.org/html/1806.02448#bib.bib4)*,* [*5*](https://ar5iv.labs.arxiv.org/html/1806.02448#bib.bib5)*,* [*6*](https://ar5iv.labs.arxiv.org/html/1806.02448#bib.bib6)*]*. The *planning tracks* of the competition, where agents are given a forward model allowing them to plan but no training time between games, have been very popular and seen a number of strong agents based on tree search or evolutionary planning submitted.”

**EVALUATION:** Mean episode reward / timesteps taken

**On evaluation** “This is done by reporting the sum of the incremental rewards for the episode at a given time step. Since this data is noisy due to episode restarts, the 20 results are averaged to smooth the graph and better show a trend. A2C allows running in parallel, we were able to run 12 networks in parallel at once. To keep the comparisons fair, A2C is still only allowed one million GVGAI calls and therefore each of the 12 networks is given one-twelfth of a million calls each. This results in the training graph seen in Figure [4](https://ar5iv.labs.arxiv.org/html/1806.02448#S4.F4). To compare this with the linear algorithms, each time step of A2C is associated with 12 time-steps of the DQN algorithms in Figure [3](https://ar5iv.labs.arxiv.org/html/1806.02448#S4.F3). The value for each time step of A2C is the average of all 12 rewards.”

PAPER 6 =[1] Evolving Curricula with Regret-Based Environment Design

A. A. J. P.-H. O. U. M. J. UCL, M. A. M. D. U. B. M. S. UCL, M. A. J. F. O. U. E. G. UCL, M. A. T. R. UCL, M. A. \*Equal contribution P. March 3, and 2022 ARXIV Full paper, ‘Evolving Curricula’. Accessed: Nov. 08, 2024. [Online]. Available: <https://accelagent.github.io>

**Paper looks at training a generalised model using curricula, teacher and a student models with good results**

**Also mentions Prioritized Level Replay**

Algorithms: ACCEL, PAIRED, POET, PLR

Training Techniques:UED

Notes: **On efficacy of DR** “However, DR (domain randomisation) is often not enough to train robust agents in domains where the agent struggles to make progress on many challenging levels.”

**On the realism of an MDP framework** “Despite the generality of the MDP framework, it is often an unrealistic model for real-world environments. First, it assumes full observability of the state, which is often impossible in practice. This is addressed in partially-observable MDPs, or POMDPs, which include an observation function \mathcal{I}: S \rightarrow O which maps the true state (which is unknown to the agent) to a potentially noisy set of observations O. Secondly, the traditional MDP framework assumes a single reward and transition function, which are fixed throughout learning. Instead, in the real world, agents may experience variations not seen during training, which makes it crucial that policies are capable of robust transfer.”

**On Dr**: “Domain Randomization (DR) can be viewed as the most basic form of UED.”

**On PCG frameworks:** “Our work also relates to the field of procedural content generation (PCG), which seeks to algorithmically generate diverse levels. Popular PCG environments used in RL include the Procgen Benchmark, MiniGrid, Obstacle Tower, GVGAI, and the NetHack Learning Environment. This work uses the recently proposed MiniHack environment, which provides a flexible framework for creating diverse environments.”

**On regret based algorithms:** “Dennis et al, 2020 first formalized UED and introduced the PAIRED algorithm, a minimax regret UED algorithm whereby an environment adversary learns to present levels that maximize regret, approximated as the difference in performance between the main student agent and a second agent.”

Evaluation: Test Return (score from level)

**This paper talks about the difficulty of avoiding memorisation of data in a RL setting, particularly with a continuous domain.**

**It also offers some techniques to avoid it.**

A Dissection of Overfitting and Generalization in

Continuous Reinforcement Learning

<https://arxiv.org/pdf/1806.07937>

**This paper surveys recent research of zero shot generalisation in DRL, aiming to improve algorithms capable of adapting to real world settings.  
They also aim to present a framework classifying ZSG as a class of problems, and state that generalisation suffers from the no free lunch theorem, and thus is not one singular problem.**

**Doing this they categorise types of generalisation.**

This paper surveys recent research on zero-shot generalization (ZSG) in deep reinforcement learning (RL), aiming to create algorithms capable of robust, adaptable performance in real-world settings. The authors present a unifying framework that formalizes ZSG as a class of problems, rather than a single one, to help researchers better understand and address its complexity. They categorize types of generalization, such as combinatorial and interpolation versus extrapolation, and clarify the importance of context awareness in training. The paper reviews existing benchmarks, suggests underexplored problem areas, and provides guidance for future research to foster progress toward more universally generalizable RL methods.

“We recommend that future environments should use a combination of PCG and controllable factors of variation.”

A Survey of Zero-shot Generalisation in Deep Reinforcement Learning

<https://jair.org/index.php/jair/article/view/14174/26890>

**ON difficulties in evaluating generalisation** “This means using it as theonly metric for improved performance will likely not lead to robust progress in ZSG. Further,given how broad the current set of assumptions is, it is unlikely there is a single generalmeasure of progress towards tackling ZSG: across such a broad problem class, objectivesmay even be conflicting (Wolpert & Macready, 1997).”