NTNU
Norwegian University of
Science and Technology

Department of Computer Science

IT3920 - Master Thesis for MSIT

Policy Adaptation over Environmental Configurations in Survivor Type Games

Markus Johannes Pedersen - markusjp@stud.ntnu.no - 504902

June, 2023

Abstract

This thesis explores the applicability and effectiveness of direct policy transfer in reinforcement learning (RL), with a focus on environmental configurations within the open-source game openSURVIVORS. We investigate how policy transfer can decrease the time to threshold and enhance jumpstart and asymptotic performance across various environmental configurations. Our experimental results highlight the strong influence of the game's unique environmental dynamics on policy effectiveness and transferability, with different RL agents exhibiting varied performance based on their respective training environments. While the asymptotic performance of agents trained via policy transfer remains inconclusive due to non-convergence within the study's scope, our findings demonstrate that policy transfer outperforms training from scratch, indicating its potential advantages in learning new environments. This research advances the understanding of policy transfer in RL, offering insights into training agents more effectively across diverse environments.

Table of Contents

List of Figures

List of Tables

1 Introduction

Reinforcement learning has achieved great success in playing games at a high level such as Go, Starcraft 2 and Dota 2.[1][2][3] However, the computational costs and demands associated with the successful reinforcement learning results pose a major barrier to widespread adoption of reinforcement learning techniques in practical application. Therefore, incorporating reinforcement learning techniques into practical applications requires decreases in costs which can be primarily attributed to the training time of agents.

To approach the task of decreasing the training time of agents this thesis introduces the concept of environmental configurations. An environmental configuration can be understood as a partition of the broader environment. In essence, each environmental configuration presents a unique yet similar problem for the reinforcement learning agent to solve. For instance, in the context of autonomous driving, various car models can be viewed as different environmental configurations. Each model presents unique driving characteristics due to differences in their engineering, design, and operating conditions. Despite these differences, the overall task of driving remains the same.

If the knowledge acquired in one environmental configuration could be effectively transferred to another, then the learning time could be decreased in the training of agents in subsequent environmental configurations. This is especially promising considering that these configurations are usually known to be inherently similar, indicating the greater potential for successful knowledge transfer.

Transfer learning is an established method used in machine learning and has seen success especially in computer vision.[4]. Transfer learning is also a growing field within reinforcement learning as seen in a survey from 2009[5] and more recently in 2022.[6] Many different techniques have been developed to transfer knowledge both between domains and within domains. This thesis focuses on direct policy transfer between environmental configurations.

openSURVIVORS, which is an environment inspired by Survivor games, presents an environment that can be partitioned cleanly into multiple environmental configurations. Additionally openSURVIVROS strikes a balance between trivially solvable toy problems and complex real-world problems. The proportion of the dynamics that are different between the different environmental configurations is also neither negligible nor unreasonable. These factors make openSURVIVORS an environment well suited for transfer learning between environmental configurations.

1.1 Research Goal

The research goal of this thesis is to explore the impact of pre-training and transfer learning in reinforcement learning across environmental configurations in Survivors games. Transfer learning across environmental configurations has the potential to to decrease training times in games and real-world scenarios such autonomous driving.

Before presenting the research questions a few important terms need to be defined.

Direct policy transfer refers to using the learned policies from one environmental configuration as a starting point for training in other environmental configurations. Time to threshold denotes the amount of training time it takes for a reinforcement learning agent to attain a predefined level of performance. Jumpstart performance corresponds to the initial performance improvement observed as a result of transferred knowledge which in this case is the direct policy transfer, while asymptotic performance represents the final performance level of the reinforcement learning agent once it has fully adapted to the new environment. These three terms are common metrics used in transfer learning literature as seen in [5] and [6]

1.2 Research Questions

In order to investigate the impact of pretraining and transfer learning in survivor type games, we consider two research questions:

RQ1 To what extent does direct policy transfer decrease the time to threshold over environmental configurations?

RQ2 To what extent does direct policy transfer increase jumpstart and asymptotic performance across environmental configurations?

2 Background

2.1 Reinforcement Learning

Reinforcement learning is a method in machine learning that aims to solve the problem of what actions an agent should perform in an environment to maximise its reward. In this type of learning, an agent performs actions in its environment and receives positive or negative rewards and an updated state. The reinforcement learning agents then learn to modify their behavior through trial and error to maximise their rewards.

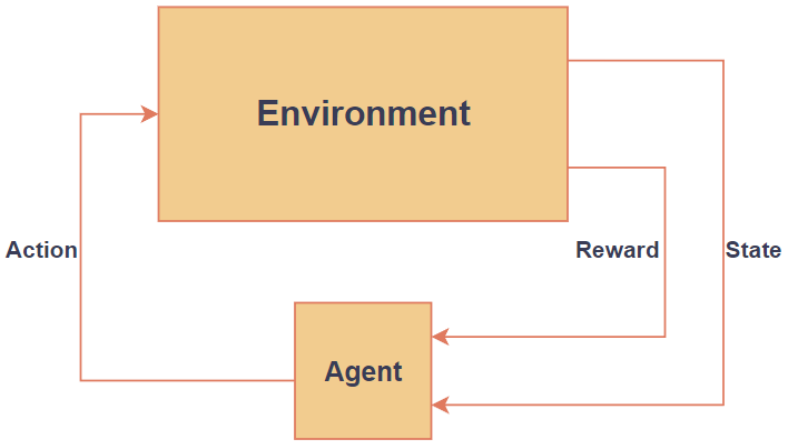

Figure 1: The Reinforcement Learning Loop

2.1.1 Markov Decision Processes

The environments that an agent acts in are often formalized as a Markov Decision Processes. A Markov Decision process is a $6 - tuple(S, A, p, r, s_0, s_f)$. S is the set of possible states. A is the set of all actions the agent can perform. $p(s'|s, a)$ is the transition function which gives the probability of transitioning into s' from s given action a. $r(s, a, s')$ is the reward function which defines the reward that is received for taking action a and transitioning into state s'. s_0 and s_f are the possible initial states and terminal states. An essential property of Markov decision processes is the Markov property. The Markov property means that future states are independent of past states given the current state. Effectively this means that memory mechanisms are not required in the agents and they can act on purely on information from the current state and still perform optimally.

2.1.2 Policies

The way an agent acts in its environment is through its policy π. The policy is a function that maps states to actions. The goal of a reinforcement learning algorithm

is to generate this policy so that the action mapped to from state s is the action that maximises the agents reward.

2.1.3 Value Functions

The value function is a function that maps a state to the reward received from that state and all future visited states. The future visited states are defined by a policy. Given an optimal policy then the respective value function represents the maximum reward that can be received from each state.

2.1.4 Exploration versus Exploitation

The problem of exploration versus exploitation is often described in the context of the multi armed bandit problem. Consider multiple levers that can be pulled. Each lever has a reward associated with it that the agent receives when the lever is pulled. The exploration versus exploitation problem is then when should the agent pull the lever that gives the most reward based on its current knowledge versus when it should try to pull a new lever to gain more information. This problem becomes drastically more difficult to solve effectively when the number of possible levers, which can be generalized to the number of unique state action pairs, grows very large.

2.1.5 Q-Learning

Q-Learning is a model-free, off-policy reinforcement learning algorithm. Model free, means that the the algorithm is not given a model of how the environment transitions from state to state given an action. The model of the environment is instead implicit in the agent's learned policy. Off-policy means that the policy that explores that environment and generates learning examples is not the policy that is being optimized. The algorithm learns by iteratively updating the Q-function.

The Q-function is very closely related to the value function where the Q-value represents the expected cumulative reward an agent will receive by taking a particular action in a specific state. The Q-value is updated using a variation of the Bellman equation.

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha[r_{t+1} + \gamma max_a(Q_(S_{t+1}, a) - Q(s_t, a_t)]$$

This states that the Q-value of the state and action pair (s, a) is equal to the current Q-value plus an error term. This error term represents the difference between the maximum Q-value over actions that can be taken in the next state minus the current Q-value.

In problems with small and discrete state spaces the Q-function be represented in a table which is called the Q-table. This is known as tabular reinforcement learning.

Two policies can be derived from the Q-function one that is responsible for generating values in the table and one that defines the current most optimal policy. The first is called the ϵ-Greedy policy and the other is the pure Greedy policy. The ϵ-Greedy policy picks the action that has the highest Q-value for a given state with probability 1-ϵ and a random action with probability ϵ. This allows the policy to achieve a balance between exploration and exploitation based on the value of ϵ. When after training the Q-table for multiple iterations until optimally the Greedy policy, which purely takes the action that has the highest Q-value, is now equal to the optimal policy.

2.1.6 Function Approximators

For most environments it is infeasible to have direct representations of the value functions and Q-functions such as in a Q-table. Therefore they are often approximated using function approximators. Artificial neural networks, initially inspired by the biological brain, act as universal function approximators[7] and are generally used for this purpose in reinforcement learning.

2.2 Proximal Policy Optimization

Proximal Policy Optimization(PPO) is a state-of-the-art(SOTA) reinforcement learning algorithm that belongs to the policy gradient family[8]. Policy gradient algorithms are trained by optimizing an objective function through gradient ascent. In contrast to Q-learning, PPO is an on-policy reinforcement learning algorithm which means that the policy that is being updated is the same policy that is acting in and exploring the environment. In the case of PPO, the objective function is the clipped surrogate objective function, which is designed to minimize the divergence between the next policy and the current policy.

$$Loss(\theta) = min(r_t(\theta)\hat{A}_t, clip(r_t(\theta)), 1-\epsilon, 1+\epsilon)\hat{A}_t$$

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta old}(a_t|s_t)}$$

Where $r_t(\theta)$ is the ratio of the probability of action a being taken at state s at time t for the updated policy versus the current policy. The advantage can be any advantage estimation, but in the case of the StableBaselines3 implementation the advantage is calculated using the Generalized Advantage Estimator [9]. The advantage is based on the value function and Q-function to estimate the relative advantage of each action. Formally the advantage is defined as $A(s,a) = Q(s,a) - V(s)$.

PPO is a SOTA algorithm within reinforcement learning and has become the default algorithm used by OpenAI[10] and was therefore chosen as the learning algorithm for this thesis. The robustness and simplicity of PPO made it a desirable choice compared to the alternatives such as Deep Q-Networks, Advantage Actor Critic, and Deep Deterministic Policy Gradient.

Figure 2: A screenshot of the early game of vampire survivors with a wave of bats approaching the player

2.3 Vampire Survivors

Vampire Survivors is an example of a Survivor type game. The game puts both the player's strategy and reflexes to the test. With the objective of surviving for 30 minutes, the player is given the opportunity to choose from a variety of weapons, items, and upgrades to help them along the way. These items are selected and used automatically based on their individual cooldowns. Experience points can be collected from defeating enemies and accumulating enough experience points will result in a level up. When the player levels up they can choose from a pool of three options sampled from total remaining weapons, items, and upgrades. Many of the items have synergistic effects, allowing for the player to create unique strategies for tackling the oncoming waves of enemies. Only by combining the player's movement with the selection of synergistic items can the player survive the 30 minute challenge.

The waves of enemies in the game become increasingly more difficult as the game progresses, with the properties of the wave changing every minute. These properties include the type of enemies, the number of each type, additional wave events, and bosses. Because the sequence of these waves is predetermined, players can exploit the sequence to maximize their chances of surviving until the 30-minutes are up. Particularly challenging waves can also serve as bottlenecks for a player's progression, as they must learn to successfully overcome them. A classic example is the wave that appears at the 11-minute mark, in which the player is surrounded by enemies from all sides. The only way to survive is to have a powerful enough loadout to create a path through the horde. This can serve as both a bottleneck and an opportunity, as a player with a powerful enough loadout can use the wave as a chance to become even more powerful by killing a large number of enemies which will result in more potential experience that can be used to level up.

(a) Item choices on level up

(b) The current loadout with their level showing as filled dots

The game can be decomposed into two main challenges. The first challenge involves moving the character around the map while avoiding enemies and positioning oneself to get the most out of current loadout of weapons and upgrades. The second challenge is selecting the items and upgrades that will allow the player to most easily survive the oncoming onslaught. The corresponding action space is selecting either the first, second or third option that is presented to the player when leveling up.

2.3.1 Game Trajectory

The game starts with the player having a default weapon based on their chosen character. In this trajectory the starting weapon will be the whip. A screenshot of the game can bee seen in 2 with a wave of bats approaching the player. The player can move their player character through the game world with the arrow keys. The whip weapon will then be automatically be used and damage enemies that are caught within its area of effect. When enemies have received enough damage they will die and have a chance to drop experience crystals. Throughout the report these experience crystals will be referred to as experience. Experience will automatically be picked up by the player when they are within a small distance from each other. When enough experience has been gathered the player levels up and is given a choice. An example of the choices given is shown if 3a. Here the player is provided with the choice of three weapons, Santa Water, Pentagram, and Ebony Wings. The selection could also include items which have passive effects such as increasing damage, or decreasing weapon cooldowns. The player will be able to choose fill their load out with six unique weapons and six unique items. Once an item has been added to their loadout the pool of choices may include upgrading a weapon or item from their loadout to increase their power. The player will take damage from the enemies if they get too close. If the player takes too much damage they will die and lose the game. If the player manages to survive for 30 minutes they win the game.

2.3.2 Environmental Configurations

The openSURVIVORS environment which is an environment inspired by Vampire Survivors removes the choosing of weapons and items as a part of the game loop. The player is instead granted one weapon which is leveled up automatically when

enough experience is gained. Each loadout now consists of only a single weapon and can be seen of as individual environmental configurations.

3 Related Work

3.1 Transfer Learning

Transfer Learning explores the idea that the encoded knowledge required to solve one task can be utilized to solve other tasks. Transfer learning has achieved success in other machine learning fields such as computer vision[4] and has also seen a lot of focus within reinforcement learning. There are two important questions that are important to consider in the context of transfer learning. What are the potential differences in tasks and what kind of knowledge can be transferred across these tasks. This requires a formalization over what a task is in reinforcement learning and how they can differ from one another. All of the properties that make up an MDP set the stage for the potential variations in environments and the tasks that are to be learned within those environments.

3.1.1 Differences in Environments

Cartpole, which was used in one of the earliest transfer learning works[11] in reinforcement learning, will be used as an example environment for potential differences between environments. The Cartpole task is a simple classic control problem that involves balancing a pole vertically on a cart through applying forces to it. The goal is to keep the pole balanced for as long as possible.

An environment can be split up into three main categories the environment's transition dynamics, states, and actions. Environments can differ along these categories to arbitrary degrees. For example chess and a robotics control are different in transition dynamics, states, and actions available. However, large differences such as these are outside the scope of this thesis. While the degree of similarity or difference between two environments is not formally defined, some informal estimations of similarity act as a good starting point.

3.1.2 Transition Dynamics

Within Cartpole the transition dynamics are defined by a physics model. This physics model is dependent on different parameters such as the mass of the cart and the pole, the length of the pole, friction coefficients for the cart and the pole, and the frequency of physics updates. These variables can be modified, removed, or new variables can be added all together to change the transition dynamics of the environment. In [11] the mass and length of the pole where changed as the difference between the source and the target task.

3.1.3 State Spaces

The environment's state is the information that is presented to the agent acting in the environment. In Cartpole the state is usually represented as four variables. Cart

position (x): The horizontal position of the cart on the track. Cart velocity (dx/dt): The rate of change of the cart's position. Pole angle (θ): The angle between the pole and the vertical axis. Pole angular velocity (dθ/dt): The rate of change of the pole's angle. Information can be added or removed, or modified to change the state representation. However it is important to keep in mind that for an agent to be able to learn to act in an environment it the information present in the state has to be sufficient for the desired behavior.

3.1.4 Action Space

The actions in an environment define what an agent can do within the environment. In Cartpole the agent can take one of two discrete actions to control the cart every time step. Move left: Apply a force to the left to push the cart. Move right: Apply a force to the right to push the cart. Similarly to the states, actions can be added, removed or modified. In the case of modifying the actions, since the transition dynamics are dependent on action, changes in the actions can also be thought of as modifying the transition dynamics of the environment.

3.1.5 Practical Consideratios

Common reinforcement learning algorithms such as DQN and PPO are fixed in their inputs and outputs which relate to the state space and action space respectively. In the standard Cartpole case this is 4 variables describing the state and 2 possible actions. This means that if the size of the state space or action space changes from the source task to the target task then directly reusing the policy is not possible. If it is still desirable to reuse the policy directly an additional transfer function is necessary to map source states to target states and source actions to target actions.

3.1.6 Transfer Learning Metrics

Some metrics have been established within transfer learning[5][6] 1. Jumpstart performance: The initial performance of an agent after transfer before any additional training takes place compared a default initialization of the agent. 2. Asymptotic Performance: The the final performance the transfer learning agent compared to learning without transfer. 3. Total Reward: The total reward accumulated by the transfer learner compared to without transfer. 4. Transfer Ratio: The ratio of the total reward accumulated by the transfer learner and the total reward accumulated by the non-transfer learner. 5. Time to Threshold: The learning time needed by the agent to achieve a pre-specified performance level may be reduced via knowledge transfer.

3.1.7 Negative Transfer

In some cases transfer learning can lead to worse performance on downstream tasks.[12] This survey does not cover reinforcement learning, but raises the point of transfer

learning can be detrimental when the source task and the target task are too dissimilar. It highlights 4 major contributors to negative transfer. Differences in the domains, Efficacy of the transfer methodology, Quality of the data that was used in the source domain, and Quality of the data used in the Target domain. Additional research is also being done in estimating the similarity between domains such that a quantitative analysis on domain divergence and negative transfer can be established. Other targets of research have been to improve the efficacy of the transfer methodologies.

3.1.8 Behavior Cloning

Behavior Cloning is a successful method related to transfer learning. The Atari Grand Challenge Dataset[13] presents a large amount of human demonstrations of Atari game play which is used to train policies directly to mimic their behavior. The AlphaGO policy, was initially trained on human demonstrations of GO games, before being trained further with Monte Carlo Tree Search.[14]

A combination of pretraining on videos, reward shaping, and finetuning has shown promising results in Minecraft[15].The agent was tasked to to try to recreate the behavior from the videos as accurately as possible. By the pretrained policy as a base for reinforcement learning with an evolving reward function performance that was previously unobtainable was achieved.

These works in behavior cloning are similar to the work in this thesis as that they leverage pre-trained policies as a starting point for further training. The pre-trained policies are, however, trained on human performance, which is generally very expensive to produce. However the success of these both AlphaGO and the Minecraft agents show promise for policy transfer over environmental configurations.

3.1.9 Knowledge Distillation

Knowledge Distillation uses one or more teacher policies to inform a student policy. The knowledge can be distilled to the student policy through supervised learning by minimizing the divergence between he output of the teacher policy and the student policy.[16] In the case where there are multiple teacher policies the contribution of each teacher agent can be weighted

3.2 Curriculum Learning

Curriculum learning, a concept within machine learning, aims to enhance learning by presenting training examples or learning tasks in a specific order that facilitates the learning process. Notably, arranging training examples in ascending order of difficulty has been shown to expedite convergence and improve overall performance. Curriculum learning can be thought of as a generalization of transfer learning.

In reinforcement learning, the sequencing of tasks or experiences into a curriculum

has primarily been achieved through manual generation for each problem.[17] When transitioning from one task to another within a curriculum, RL agents employ transfer learning techniques to transfer knowledge. Since the tasks within a curriculum can differ in terms of state and action spaces, transition functions, and reward functions, effective transfer and integration of relevant information across tasks become crucial. The foundation of curriculum learning heavily relies on prior research in transfer learning, as knowledge transfer from one stage to another within the curriculum is essential.

4 Method

4.1 Environment

The environment used in this thesis to explore transfer learning is the openSURVIVORS environment. openSURVIVORS is inspired by the game Vampire Survivors which has spawned a genre of Survivor games. This environment was initially created in the prestudy to this thesis and has been further refined to accommodate the research goals of this thesis. Survivor games are uniquely suited to explore the branch of transfer learning that this thesis is concerned with. Each weapon that the player character utilizes can be thought of as separate environmental configurations, where a subset of the transition dynamics are changed, but the state spaces and action spaces stay the same.

4.1.1 Environment Description

openSURIVORS is a top down game where the player controls a character has one of four weapons Garlic, Knife, Whip, and Bible. Figure 4 shows the player character with the Garlic weapon. The The objective of the environment is to survive for five minutes while waves of enemies approach the player. The only actions available to the player is its movement which allows it to move both orthogonally and diagonally for a total of 8 directions. The player has to move to dodge the incoming enemies and utilize their weapon which attacks automatically to facilitate their survival. Every 30 seconds the difficulty increases with both a more difficult and a greater number of enemies spawning to assault the player.

4.1.2 Implementation details

openSURVIVORS is meant to be a learning environment for reinforcement learning and as a result does not contain all the content and intricacies of Vampire Survivors. Vampire Survivors contains over 32 weapons[18] and 198 enemies[19]. openSURVIVORS is limited to four weapons and two enemies.

The chosen weapons were chosen to exhibit the particular interesting dynamics present in Survivors games. Each weapon is reliant on avoiding enemies, but requires a distinct movement styles to be utilized effectively. The difficulty of the environment progresses with time and it can be useful to classify this progression of difficulty into two stages. A power gathering phase and a survival phase. In the power gathering phase the enemies that spawn are simpler to defeat allowing the player to collect experience crystals to level up and become more powerful. The survival phase begins when the difficulty has progressed and more powerful enemies spawn and require the player to focus more on avoiding enemies than collecting experience.

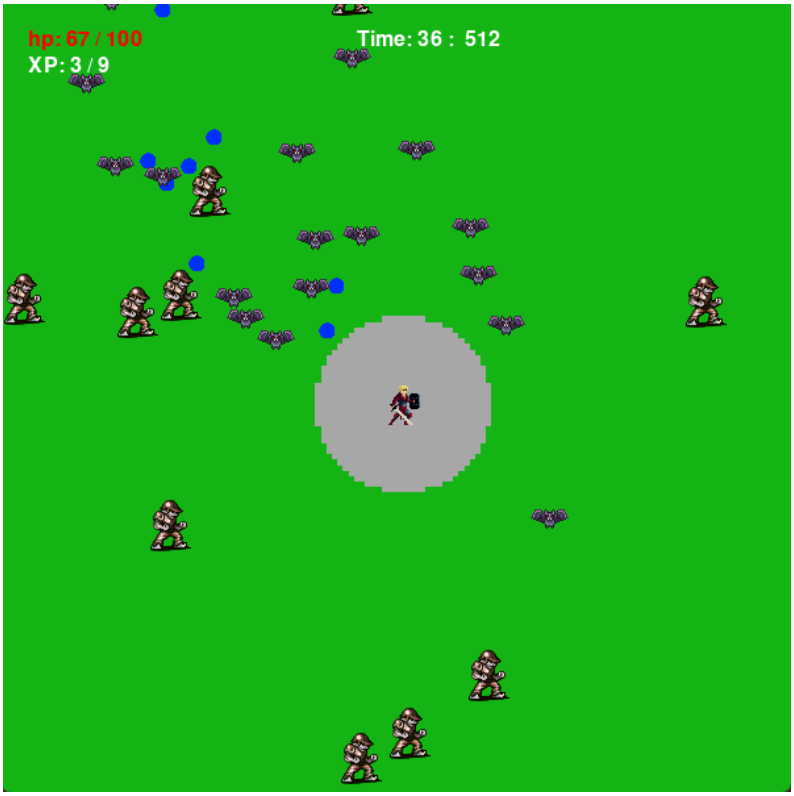

Figure 4: A screenshot of openSURVIVORS with a wave of bats and goblins approaching the player with the Garlic weapon

4.1.3 Weapons in openSURVORS

The 4 weapons that were implemented can also fit into two categories where the required movement is qualitatively similar. Both Garlic and Bible damage enemies in a ring surrounding them requiring the player to stay relatively close to enemies to damage them, but where the direction between the player and the enemies is arbitrary. The knife and whip weapons are both dependent on the direction that the player is facing where the direction of the attack is equal to the direction of the the players movement[1]. This means that the position of the player relative to the enemies is more important for the Knife and Whip weapons and the Garlic and Bible weapons.

4.1.4 Enemies in openSURVIVORS

The two enemies implemented in openSURVIVORSS are designed to fill two roles. The first role is that of a weak enemy that can easily be killed to drop experience and let the player become more powerful. The second role is that of a strong enemy that is much more difficult to kill and is best avoided until the player has become more powerful.

[1]Whip only attacks horizontally while knife can attack in all 8 directions the player can face

Weapons	
Name	Description
Knife	The knife weapon fires multiple small square projectiles in the direction the player last moved, Each projectile deals 6.5 damage. New projectiles are fired every 3 seconds.
Whip	The whip weapon creates a rectangular area of damage either to the left or the right of the player depending on if the player last moved to the left or the right dealing 10 damage. The attack occurs every 1.35 seconds.
Garlic	The Garlic weapon creates a permanent circular area of damage around the player, dealing 5 damage every 1.3 seconds.
Bible	The Bible weapon spawns multiple projectiles that orbit the player for 2 seconds dealing 5 damage. The projectiles are re spawned every 4 seconds.

Table 1: openSURVIVORS Weapon Descriptions

Enemies			
Name	HP	damage	speed
Bat	10	3	15
Goblin	40	6	18

Table 2: Stats of the enemies in openSURVIVORS

The environment's action space is of the format (3, 3). This means that the game expects two separate actions to be given each frame. The first action can be either a 0, 1, or 2 which represents noop, left, or right respectively. The first action controls the movement of the character in the horizontal direction. The second action is the same but with noop, up, or down instead controlling the character in the vertical direction.

The environments observation space is originally a 600x600x3 rgb image which is the rendered screen. The environment is wrapped in a custom wrapper that resizes the observation to a 84x84 grey scale image with the information about the current level of the weapon concatenated. The size of 84x84 for the observation space is a standard used in many other environments that use image observation spaces such as the Arcade Learning Environment and Procgen. It was important the agent is also presented with information on what level the agent currently is so that the agents can know their weapons current capabilities. The Level Observation which is represents the level of the agent's weapon in an a array of length 8. A custom MultiInputPolicy is also used with the following network parameters shown in 4 following the structure shown in 5.

4.1.5 Seeding

The environment was seeded with seed value equal to 1 for all experiments in this thesis. This means that the location of enemy spawns is consistent from game to game. This does affect the generality of the produced agents across multiple different seeds. However, the training time decreases dramatically and stability of training is also improved. The environment becomes simpler to solve by seeding the environments and still allows for the central focus of this thesis to be studied which is transfer learning across environmental configurations and not across seeds.

4.2 PPO

Each agent trained on the openSURVIVORS environment was trained with PPO with the hyper parameters shown in Table 3. The hyper parameters were selected after an initial hyper parameter search. The reward function was also selected during this hyper parameter search. The rewards function was a composite reward function. A negative reward was given equal to the amount of health lost by the player to encourage the agent's to avoid enemies. And a positive rewards of 1 was given for each experience point accumulated.

While the environment is seeded the PPO agents still sample their actions from their policy which is a probability distribution over actions dependent on the current state. The randomness present in PPO is essential for the algorithms exploration. This means that there is inherent randomness in the training trajectories that occur while using PPO.

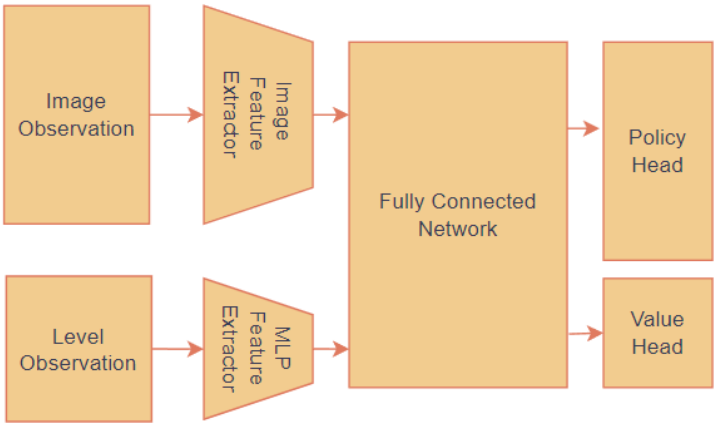

Figure 5: Network Architecture of the Policy used in the PPO algorithm

4.3 Experiments

Through this project two major experiments were conducted to answer the research questions presented in this thesis. The first experiment focuses on the the first research question and sets the stage for the following experiment by creating the pre-trained agents whose policies will be used in the policy transfer. The second experiment focuses on the second research question utilizing direct policy transfer.

4.3.1 Experiment 1: Pre-training

To be able to say anything about the jump start performance of agents across environmental configurations, agents that are trained on those environmental configurations

Hyper Parameters	
Learning Rate	0.0003
N Steps	4096
Batch Size	64
N Epochs	10
Gamma	0.99
GAE Lambda	0.95
Clip Range	0.2
Entropy Coefficient	0
Value Function Coefficient	0.5
Max Gradient Norm	0.5

Table 3: Hyperparameters for PPO.

have to be created. An agent is trained on each of the four environmental configurations present in the openSURVIVORS environment.

Each agent was trained for 2 million time steps. The trained agents are not expected to reach optimal performance, but training for 2 million time steps guaranteed that the agents learned to perform significantly better than random. The four agents produced from this experiment will be referred to as Pre-Train-Garlic, Pre-Train-Knife, Pre-Train-Whip, and Pre-Train-Bible.

After training was completed the performance of the trained agents was evaluated on each environmental configurations to show the jump start performance achieved. To account for the variability in performance due to the random nature of PPO policies each agent was evaluated on each environmental configuration 30 times

4.3.2 Experiment 2: Direct Policy Transfer

The second experiment was designed to gather information about how the agents learned from a random starting point versus a pre-trained policy. Various random factors contribute to variation in training reinforcement learning agents including the initialization of weights in the neural networks and random sampling of actions from the agent's policy. To account for this variation multiple agents were trained for each source and target environmental configuration. To be able to perform the training runs within the allotted time for this thesis the agents were trained for 200k time steps.

To establish a baseline to compare the pre-trained policies too, 10 agents for each environmental configuration were trained from scratch for 200k time steps.

Pre-Train-Garlic will be used as an example. The Pre-Train-Garlic agent is selected as a source which means that the weights that define the agents policy are loaded as

Image Observation Extractor					
Layer	In Features	Out Features	Kernel Size	Stride	Slope
Conv2d	1	32	5, 5	1, 1	NA
MaxPool2d	NA	NA	2	2	NA
LeakyReLU	NA	NA	NA	NA	0.01
Conv2d	32	64	3, 3	1, 1	NA
MaxPool2d	NA	NA	2	2	NA
LeakyReLU	NA	NA	NA	NA	0.01
Flatten	NA	NA	NA	NA	NA

Level Observation Extractor					
Linear	8	16	NA	NA	NA

Fully Connected Network					
Concatenate	NA	23120	NA	NA	NA
Linear	23120	128	NA	NA	NA
LeakyReLU	NA	NA	NA	NA	0.01
Linear	128	128	NA	NA	NA
LeakyReLU	NA	NA	NA	NA	0.01

Policy Head					
Linear	128	6	NA	NA	NA

Value Head					
Linear	128	1	NA	NA	NA

Table 4: Network Parameters for the PPO policy

a starting point for further training.[2] For each other environmental configuration, in this case Knife, Whip, and Bible, 10 agents are trained from that starting point for 200k time steps.

4.3.3 Agent Analysis

The policies that the agent's learn define the movement of the player character in the openSURVIVORS environment. Qualitative descriptions of the different agent's movement can serve as a useful tool for analysing the learned behavior and how that relates to the transferal of knowledge from one environmental configuration to another. However being able to back up the qualitative descriptions with quantitative data on the movement of the agents within the environment would be greatly beneficial. Therefore during the evaluation of the agents produced in the previous experiments, the x and y positions of the agents were recorded as well as the number of kills.

[2]Since the weights are loaded directly from the pre-trained agents policy there is no variation in weight initialization across these 30 agents.

5 Results

5.1 Experiment 1

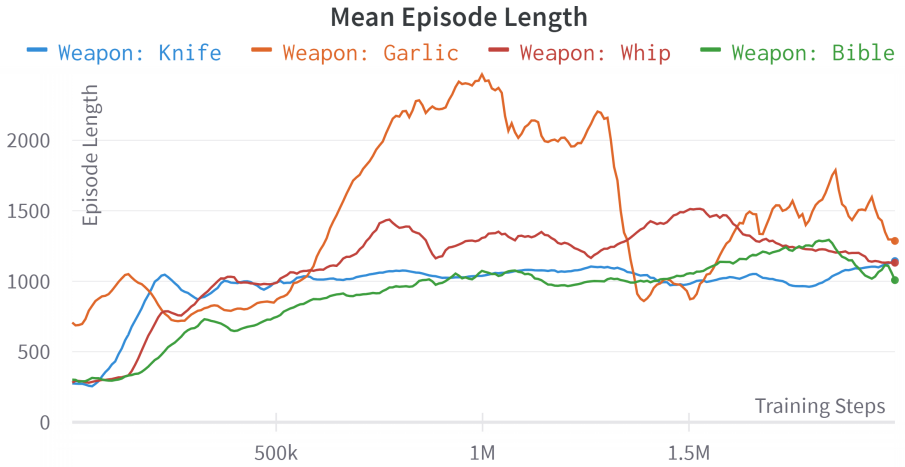

Figure 6: Pretraining agents trained for 2 Million steps for each weapon

The results from training agents across different environmental configurations are presented in Figure6 Notably, the Knife, Whip, and Bible agents all began with similar performance levels, which is confirmed by the pre-training test results in Table. 5. However, the Garlic agent initially outperformed the others, surviving over twice as long initially.

After reaching a performance where the agents survive for around 1000 time steps the performance fluctuates around that level with the exception of garlic agent. The Garlic agent's performance far surpasses the other agents until performance deteriorates back to a level comparable to the other weapons.

As demonstrated in Table6 the final performance of each agent showed significant improvement compared to their untrained state. The Garlic weapon and bible also show a much greater degree of variation in performance compared to Knife and Whip. Even amongst the high variation weapons Garlic has almost a two times larger variation than Bible.

Weapon	Mean Episode Length	STD
Garlic	720	20
Knife	280	25
Whip	278	20
Bible	283	31

Table 5: Initial performance of non-trained agents

Weapon	Mean Episode Length	STD
Garlic	1390	1115
Knife	1108	165
Whip	732.0	175
Bible	1044	619

Table 6: Mean Episode Length and Standard Deviation of baseline agents

5.2 Experiment 2

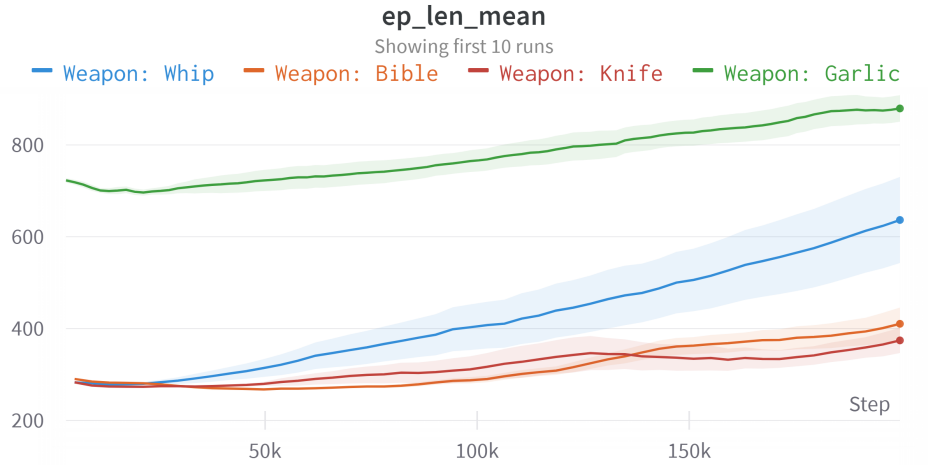

Figure 7: Baseline performance after 200k time steps for each weapon averaged over 10 training runs with standard error

Figure 7 shows the average of training 10 agents from scratch for each weapon. There is a lot of variation in the training of the agents this can be seen in more detail in the appendix. There seems to be a slight upward trend amongst all the weapons, with the Whip weapon showing the greatest degree of improvement. This upward trend indicates that the agents do have the ability to learn to varying degrees within the first 200k time steps.

Weapon	Mean Episode Length	STD
Knife	292	112
Whip	311	155
Bible	368	202

Table 7: Jumpstart performance with Garlic as Source Weapon

Table 7 shows the jump start performance of the Pretrain-Garlic agent in the other environmental configurations. Performance does not generalize well to the other environmental configuration showing performance comparable to untrained agents, but with a higher degree of variation. The higher variation is an indication that the learned behavior of the agents sometimes perform better than random behavior.

Figure 8 shows the results of transfer training from the Pretrain-Garlic agent. The

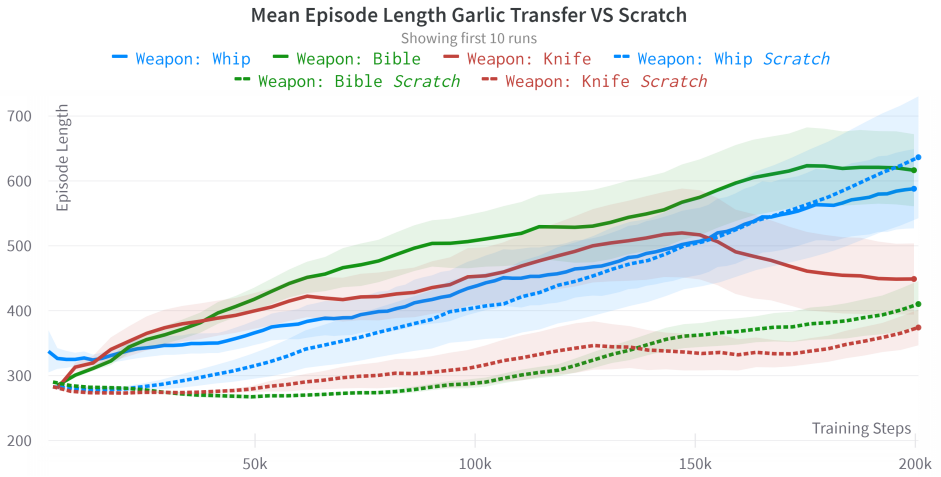

Figure 8: Training 200k time steps from Garlic base averaged over 10 runs versus from scratch

initial starting points for the training are all similar, but the transfer learning agents begin to learn much quicker than the from scratch counterparts.

Weapon	Mean Episode Length	STD
Garlic	924	204
Whip	1117	159
Bible	980	241

Table 8: Jumpstart performance with Knife as Source Weapon

Table 8 shows the jump start performance of the Pretrain-Knife agent in the other environmental configurations. Performance generalizes to the other environmental configurations maintaining almost identical performance to the original environmental configuration it was trained on. The standard deviation in jump start performance is also quite low compared to the mean which indicates consistency in its performance.

Figure 9 shows that the Bible and Whip transfer learning agents start of much better then the from scratch counterparts. The Bible and Whip agents have a flat curve on average which indicates a lack of learning. The Garlic agents on the other hand do show an upward trend in performance after pretraining.

Weapon	Mean Episode Length	STD
Garlic	860	202
Knife	516	215
Bible	537	367

Table 9: Jumpstart performance with Whip as Source Weapon

Table 9 shows the jump start performance of the Pretrain-Whip agent in the other environmental configurations. Performance generalizes quite well to other environmental configurations with a slight deterioration compared to the environmental con-

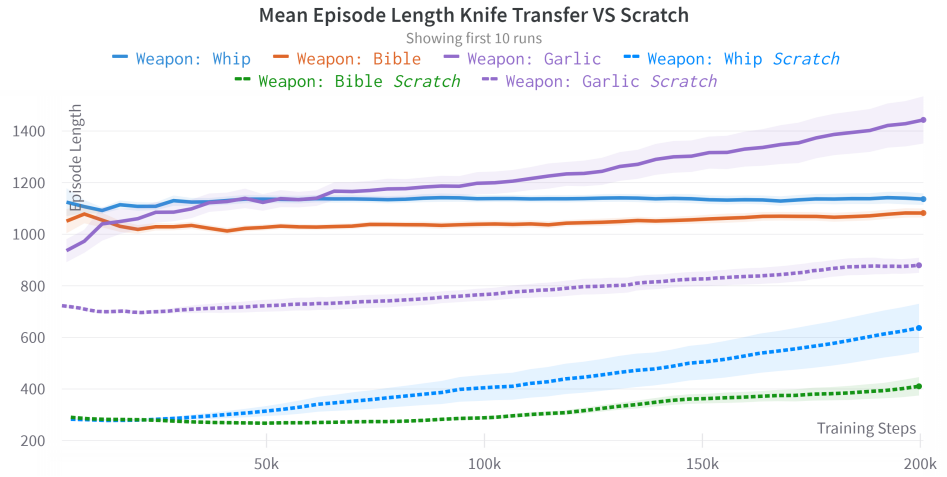

Figure 9: Training 200k time steps from Knife base averaged over 10 runs versus from scratch

figuration it was trained on.

Figure 10 shows that all transfer learning agents start of much better then the from scratch counterparts. Agents trained in each environmental configurations all have a upward trend with the garlic and knife weapons having the most significant trend.

Weapon	Mean Episode Length	STD
Garlic	875	446
Knife	893	311
Whip	969	378

Table 10: Jumpstart performance with Bible as Source Weapon

Table 10 shows the jump start performance of the Pretrain-Bible agent in the other environmental configurations. Performance also generalizes quite well to other environmental configurations with almost no deterioration in performance compared to the environmental configuration it was initially trained on.

Figure 11 shows that all transfer learning agents start of much better then the from scratch counterparts. With the whip starting point deteriorating quickly during training before beginning to recover. While the performance doesn't increase relative to the starting point it does still relearn faster than the from scratch counterpart. All environmental configurations have an upward trend with the Whip and Garlic weapons having the most significant trend.

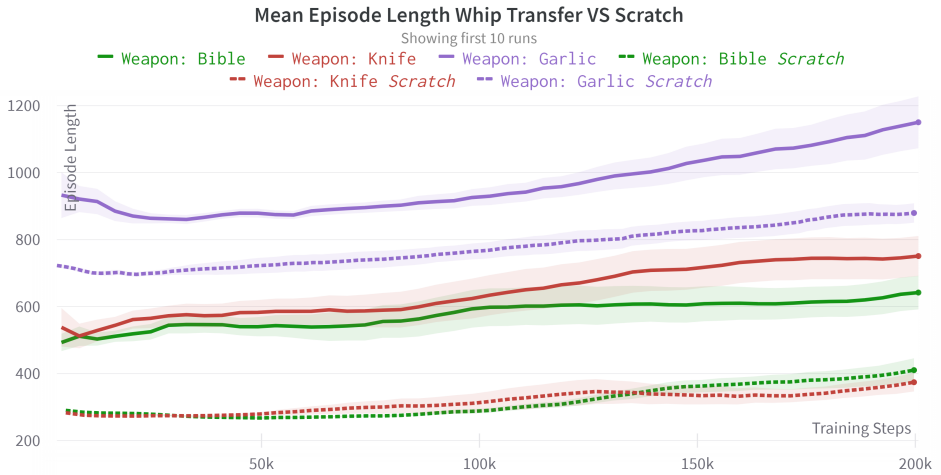

Figure 10: Training 200k time steps from Whip base averaged over 10 runs versus from scratch

5.3 Agent Analysis

Results of the recording the X and Y position of the player over 30 evaluations for each weapon of the pre-trained agents from experiment one. Lists of the X and Y positions were accumulated in a list during each evaluation. The Table 11 shows the average of the average X and Y positions in the accumulated lists. As well as the average standard deviation of the X and Y position lists. These results can be used to give a rough indication of trends present in the movement of the agents. The amount of kills was also recorded during the previous evaluations. The results can be seen in Table12.

	Garlic	Knife	Whip	Bible
Average Average X	-486	-548	-433	-590
Average Average Y	202	-1360	141	405
Average X STD	314	266	277	276
Average Y STD	202	716	95	191

Table 11: Movement Analysis of pre-trained agents

	Garlic	Knife	Whip	Bible
Average Kills	10.7	1.1	3.3	3.6
Kills STD	5.6	0.9	2.0	1.9

Table 12: Average Kills by Environmental Configuration

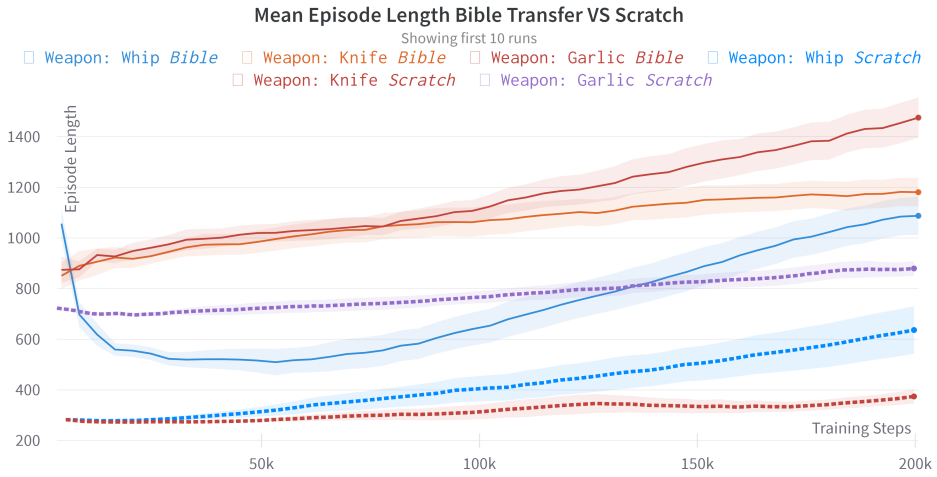

Figure 11: Training 200k time steps from Bible base averaged over 10 runs versus from scratch

6 Discussion

The following discussion provides an analysis of the experiments conducted in this thesis, aiming to address the research questions concerning time to threshold, jump start performance, and asymptotic performance in the context of direct policy transfer across environmental configurations. By examining the results and their implications, we gain insights into the applicability and effectiveness of policy transfer in openSURVIVORS.

6.1 Results Summary

The results section indicates significant improvements in agent performance following training across various environmental configurations. Initial performance levels were similar for Knife, Whip, and Bible agents, but the Garlic agent initially demonstrated superior survival times. The Garlic agent's performance fluctuated significantly and was initially exceptional before regressing to levels comparable to other agents. Garlic's fluctuation will be looked at in more detail later in the discussion.

Average training results of from scratch agents revealed substantial variation, with a slight upward trend suggesting that the agents learn to varying extents within the first 200k time steps. Notably, the Whip agent showed the most significant rate of improvement within these first 200k time steps.

Upon testing jump start performance, the Pretrain-Garlic agent exhibited no significant improvement in other environmental configurations compared to untrained agents, albeit with greater variation. In contrast, the Pre-Train-Knife agent maintained almost identical performance across different environmental configurations, demonstrating a high consistency as well. The Pre-Train-Whip and Pre-Train-Bible

agents also generalized well to other environments with a small performance deterioration.

The results from the experimentations to not reveal any clear answers relating to asymptotic performance and time to threshold. The final, or asymptotic, performance of the agents could not be conclusively determined within the scope of this study, since the performance did not converge. However, the agents trained using policy transfer did consistently demonstrate higher performance levels than those trained from scratch, but this higher performance seems to mostly be attributed to the great increase in jump start performance.

Policy transfer from the Pre-Train-Garlic agent did show faster learning compared to the the non policy transfer agents. Due to the great increase in jump start performance the differences in learning speeds are hard to compare, though some results do stand out. The Whip and Garlic agents emerged as the most effective target models for transfer learning, showing the highest upward performance trends from policy transfer.

6.2 Environmental Differences and Policy Adaptation

To further understand the observed variations in performance and the potential for policy generalization, it is important to consider the qualitative and quantitative differences between the environmental configurations in openSURVIVORS.

Qualitatively, the environmental configurations exhibit distinct characteristics as the differences between the weapons Garlic, Knife, Whip, and Bible. These variations contribute to the uniqueness of each configuration and provide distinct opportunities for exploitation.

6.2.1 Consistent Environment Dynamics

The main dynamic present in the openSURVIVORS environment other than the weapons and the movement of the player character are the enemies. These spawn in at a random point[3] on a circle with the player defined as the center. Every 30 seconds the amount and type of enemies that spawn change. The first wave consists only of the weak bat enemy. The second wave starts including the more difficult Goblin enemy.

6.2.2 Garlic: Environmental Configuration

The Garlic weapon is the much stronger than the other weapons which is shown by the higher relative performance even in untrained agents seen in Table 5. This is because garlic damages enemies in a circle surrounding the player and allows the agents to defeat the bat enemies quite easily making it difficult for the bat enemies to pose a

[3]Due to the seeding of the environment this is consistent from episode to episode

threat to the player. This means that the primary threat to the Garlic agents are the Goblin enemies that start spawning at at the 30 second mark. Since the game is run with a 0.064 delta time per frame, then there are 300 / 0.064 = 469 time steps before the Goblins begin to spawn. Considering the time it takes for the Goblins to reach the player after spawning an average survival time of 720 time steps starts to emerge as a natural conclusion. Considering the similar movement speed of the bat and goblin enemies shown in Table 2 the time it takes to reach the player and kill them is likely the time that the agents in the other environmental configurations survive which is consistently around 280 time steps which can be seen again from Table 5. 469 + 280 = 749 which is roughly equivalent to the average survival of the untrained garlic agents at 720 time steps.

It is therefore likely that the only threat to the Garlic agents are the Goblin enemies and the Bat enemies can be mostly ignored. This sets the stage for the analysis of the Garlic agents both as a source for policy transfer and a target for policy transfer.

6.2.3 Garlic: Jump start Performance

Since the agent doesn't need to avoid the Bat enemies the Garlic agent doesn't learn to avoid the bat enemies. This explains why the jumpstart performance of the Pre-Train-Garlic agent is so low. However, it is still higher than a random policy indicating that it is more promising than starting from scratch. Notably, the the environmental configuration that has the highest relative jump start performance from Garlic is the Bible weapon. As mentioned in the method section the Bible weapon is the most similar weapon to Garlic attacking in a radius surrounding the player.

It is expected that as a target for policy transfer the Garlic agents should perform quite well since they will at the very least have performance greater than or equal to an average of around 700 timesteps. This is also the case empirically seen from Table 8, 9, and 10 showing an average jump start performance of 924, 860, and 875 respectively.

6.2.4 Garlic: Time to Threshold

The Pre-Train-Garlic agent also shows the most promising increase in training speed relative to from scratch compared to the other Pre-trained agents. Comparing this improvement in speed up compared to the other Pre-trained agents does however require the consideration of some important factors. Most importantly the initial performance of the Pre-Train-Garlic agent is much lower than other pre-trained agents. The final performance of the agent's is also much lower than the jump start performance of the other Pre-trained agents. Both of these facts make it in unreasonable to relate these findings to the time to threshold and asymptotic performance metrics. However since the goal of RQ2, which pertains to these two metrics, is to evaluate the eligibility of policy transfer across environmental configurations it is still valuable to analyse the results in that light. From this perspective since the agents still learn faster on average than the from scratch counterparts it indicates that there could still be an advantage to utilizing agents with low jumpstart performance across environmental configurations, regardless of the time to threshold and asymptotic performance

achieved. Further experimentation that properly addresses the two metrics would still support these claims to a much greater extent.

6.2.5 Knife: Environmental Configuration

The knife weapon is potentially the most difficult weapon to utilize effectively. The knife weapon does not provide a consistently safe area around the player that the agent can exploit such as in the case of Garlic. This forces the agent to learn to avoid both the Bat and Goblin enemies to have any chance at survival. This proposition is also supported by the fact that the knife weapon achieves the least kills amongst all environmental configurations indicating an even greater reliance on the ability to dodge.

6.2.6 Knife: Jump Start Performance

The Jumps start performance of the Pre-Train-Knife agent is the highest amongst all environmental configurations. The performance on the whip environmental configuration is even the same as the performance on its original environmental configuration. Jumpstart performance only drops slightly in the Garlic and Bible configurations. The survival time of all these agents corresponds with the beginning of the third wave of enemies which starts at around $600 / 0.064 = 938$ time steps. The third wave is when the number of Goblins increases making avoiding them much more difficult.

Since the knife weapon requires the ability to dodge it is not surprising that the the knife environmental configuration has a varying jump start performance as target for policy transfer.

6.2.7 Knife: Time to Threshold

Due to the incredible jump start performance from the Pre-Train-Knife agent the time to threshold metric loses a lot of its value as a metric. It is also unknown what the asymptotic performance of the agents are. The performance of the pretrained agents can be used as an indication for what the performance achievable by further training. Utilizing the pretrained agent's performance as a proxy for the asymptotic performance shows that policy transfer from Pre-Train-Knife on average results in equal performance in 200k time steps to the performance acheived in training from scratch for 2 million timesteps. However, this results have some important caveats.

The first caveat is the variation within the groups of agents trained for 200k time steps. The Whip and Bible weapons have almost no variation staying very consistently at the initial level for the entire 200k steps. The Garlic weapon has a lot more variation between training runs as it climbs up towards a survival of 1500 time steps which is greater than the final performance achieved by the Pre-Train-Garlic agent. The Whip and Bible weapons also do not increase in performance over the course of the 200k time steps which also indicates that it is the jump start performance that is the main contributor to the result. As mentioned previously the 1k timestep point is consistent

with the beginning of the third wave. The difficulty of learning to overcome this wave might be a bottleneck that requires very specific strategies to overcome.

The second cavaet relates to the variability in performance within of the agents across the 2 million time steps they were trained for. Each weapon, with the exception of Knife achieved a maximum performance higher than that achieved through policy transfer. The maximum performance of Garlic was 2500 which is significantly greater than that achieved by any policy transfer agent with Garlic as a target environmental configurations.

6.2.8 Whip and Bible: Environmental Configurations

The Whip and Bible environmental configurations show quite similar results to the Knife environmental configuration. Whip is similar to the knife weapon as states in previous descriptions, while bible is similar to Garlic.

6.2.9 Whip and Bible: Jump Start Performance

The jumpstart performance from Pre-Train-Whip and Pre-Train-Bible are similar to that of Pre-Train-Knife with, but with performance deteriorating across environmental configurations to a greater degree. Both Whip and Garlic require learning the ability to dodge, but as seen in Table12 Both weapons are provided almost three times as much safety from kills compared to knife. The movement of the whip weapon also seems to be coupled with the characteristics of the whip weapon. Since the whip weapon attacks to the left and the right of the character movement in these directions should be safer on average. This is corroborated by the data from Table11. The whip agent is the only agent that moves more in the X direction than the Y direction with almost a three times higher average standard deviation in the X direction compared to the Y direction.

6.2.10 Whip and Bible: Time to Threshold

Similarly to the analysis of the Garlic and Knife weapons the policy transfer does show an improvement over training from scratch. Whip is more similar to Garlic in this regard as the jump start performance is lower and the final performance is lower than the pre-trained agents performance. In this context Bible is more similar to Knife due to the similarity in jump start performance. However the agents trained from a bible base undergo a much greater degree of variation between agents where the Knife weapon is seemingly overcome the 1000 time step bottleneck that was suggested in the Knife section. This could indicate that the policies that use Pre-Train-Bible as a base are on average to change. Figure 11 shows that the transfer to Whip also undergoes a deterioration before beginning to regain performance. While the final performance is the same as the initial performance there might be interesting qualitative differences before and after the deterioration and improvement.

6.3 Generalization across Environmental Configurations

Attributing the ability to generalize across environmental configurations to any specific causes is challenging. The space of possible policies that can be learned by PPO is vast, and the policies explored in this thesis represent only a tiny subset of the possible policies and the behavior that they can define. The relationships between how behavior of different policies exploit the characteristics of the different weapons definitely plays an important part in the generalization capabilities of the agents. This is further complicated by the potential of certain strategies learned by the agents at different stages of the game generalizing differently across environmental configurations. For example behavior learned by an agent trained to utilize the whip weapon, which provides a greater degree of safety directly to the left and right of the player, could generalize well to other environmental configurations in early waves but deteriorate quickly in later waves.

Overfitting is a common problem in reinforcement learning that also increases with the amount of training as agents get more specialized[20] Overfitting can be thought of as the antithesis of generalization where the policies that are generated are tightly coupled with the specifics of the observed states. In the case of openSURVIVORS this could be policies primarily utilizing the information in the pixels that are different across environmental configurations. This could lead to the transfer across environmental configurations to be more difficult. Overfitting to environmental configurations in openSURVIVORS does not seem to be a problem in this thesis due to the generality of agents produced. However, overfitting might become a greater problem as training times increase. The agent's that were trained in this thesis were not trained until optimal performance and the more they are trained the more specialized they potentially become.

6.4 Training Instability

Even though PPO was used, which is known to be quite stable in its training due to the clipping in its objective function, the training did seem to have some instabilities present. The most notable indicator of this instability is the Garlic weapon. Garlic did have the highest variation in performance compared to all the other weapons. This high variation might be a contributing factor to the deterioration in performance after around 1.3 million training steps.

The deterioration might also occur from the new policy deviating too much from the previous policy during optimization, which could mean that the clipping fraction hyper parameter used in PPO was too large. Further experimentation on the clipping fraction in the openSURVIVORS environment could help narrow down the cause of the instability.

The agents are implicitly incentivised to gather experience which increases their power and leads to easier survival. Garlic is the environmental configuration that collects the most experience. This means that there are more samples in the training trajectories that contain experience collection and power increases. Since the location of experience is often closer to enemies it is also much riskier behavior to try to collect experience. This fact both supports the increased variance in performance of the

garlic environmental configuration and the instability in performance during training.

6.4.1 New Environment

openSURVIVORS is a new environment and therefore there do not exist any baseline performance results on how agents can perform in this environment in the different environmental configurations. The relative difficulty of the different environmental configurations is also unknown, but can be estimated by the results of the different experiments performed in this thesis.

One estimate of difficulty is the speed at which the PPO agents learn. Figure 7 which shows the average training performance across ten different training runs from scratch shows that the relative training speeds are different across the different environmental configurations. This adds weight to the claim that the different environmental configurations are of differing difficulties as well as there being non trivial differences between the environmental configurations.

Another estimate of difficulty is the initial and final average performance of the agents across the different environmental configurations, where performance is defined as the average survival time. When considering the average survival time it is also important to consider how consistent the performance is, where a consistent high average would indicate a simpler environmental configuration versus an inconsistent low average would indicate a more difficult one. Garlic presents an example of a simpler environmental configuration which is also backed up by the qualitative descriptions given earlier in the discussion.

How different the environmental configurations are from each other is presented qualitatively in the environment description and some quantitative results can help back up these qualitative differences such as the movement of the agents that utilize the different weapons and the amount of kills achieved.

6.5 Discussion Summary

Through a comprehensive analysis of agent performance across environmental configurations in the openSURVIVORS game, several key findings were revealed:

The initial survival times varied based on the weapon characteristics, with the Garlic agent displaying higher survival due to its weapon's superior capabilities. The fluctuation in Garlic agent's performance was associated with the minimal threat from Bat enemies in the corresponding environmental configuration.

While the Garlic agent struggled to adapt its policy in other environments, indicating lower jump start performance, the Knife agent demonstrated high levels of policy generalization, exhibiting superior jump start performance across different configurations.

Overall, the policy transfer proved to be beneficial, with agents trained using this method consistently outperforming those trained from scratch. This was true even

for agents with relatively low jump start performance, implying that policy transfer may offer speed advantages in learning new environments.

6.5.1 Research question 1

The findings suggest that direct policy transfer can reduce the time to threshold across different environmental configurations. This is evident from the higher performance levels of agents trained using policy transfer compared to those trained from scratch, implying that they reach a certain performance threshold faster.

6.5.2 Research question 2

In terms of jump start performance, the benefits of policy transfer are weapon-dependent. While the Knife agent showed high jump start performance across different environments, the Garlic agent struggled to adapt its learned policy to other configurations. The asymptotic performance, however, could not be conclusively determined within the scope of this study as the performance did not converge. Still, the consistent higher performance of agents trained using policy transfer offers promising indications.

7 Conclusion

This thesis has explored the applicability and effectiveness of direct policy transfer in reinforcement learning, with a specific focus on environmental configurations within the game openSURVIVORS. Two primary research questions were investigated: the extent to which direct policy transfer can decrease the time to threshold and improve jumpstart and asymptotic performance across environmental configurations.

Results demonstrated that the effectiveness of policy transfer is closely tied to the nuances of the given environmental configuration and the unique characteristics of the weapon used. The Garlic agent's superior initial performance highlighted the importance of environmental dynamics in shaping agent behaviour and policy. In contrast, the Knife agent exhibited a high degree of policy generalization, consistently demonstrating superior jumpstart performance across different configurations.

While the asymptotic performance could not be conclusively determined within the scope of this study, policy transfer consistently outperformed learning from scratch, suggesting that it might offer speed advantages in learning new environments. These findings provide strong evidence towards the effectiveness of policy transfer in reinforcement learning, providing insights in line with the research goals of this thesis.

Despite these significant strides, several areas require further exploration. Future work could aim to conclusively determine the asymptotic performance of agents trained with policy transfer, which could not be assessed within the current study due to non-convergence of performance. This would provide a more definitive understanding of the long-term impacts of policy transfer on agent performance.

Additionally, while this study examined policy transfer within the context of the openSURVIVORS game, further research could explore policy transfer across a wider range of environments, tasks, and games to validate the generalizability of these findings.

More in-depth exploration of the role of environmental characteristics in shaping agent behaviour and policy is also warranted. This could entail designing more nuanced and complex environments, incorporating a broader range of dynamics and challenges for the agent to overcome.

In conclusion, the research presented in this thesis adds valuable insights to the current understanding of policy transfer in reinforcement learning, highlighting the critical role of environmental dynamics and paving the way for more targeted, effective strategies for training reinforcement learning agents across diverse tasks and environments.

References

[1] D. Silver, A. Huang, C. J. Maddison *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. [Online]. Available: https://doi.org/10.1038/nature16961.

[2] O. Vinyals, I. Babuschkin, W. M. Czarnecki *et al.*, 'Grandmaster level in StarCraft II using multi-agent reinforcement learning', *Nature*, vol. 575, no. 7782, pp. 350–354, Oct. 2019. DOI: 10.1038/s41586-019-1724-z. [Online]. Available: https://doi.org/10.1038/s41586-019-1724-z.

[3] OpenAI, C. Berner, G. Brockman *et al.*, 'Dota 2 with large scale deep reinforcement learning', Dec. 2019. arXiv: 1912.06680 [cs.LG].

[4] C. Sharma, 'Transfer learning and its application in computer vision: A review', Mar. 2022.

[5] 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. [Online]. Available: http://jmlr.org/papers/v10/taylor09a.html.

[6] Z. Zhu, K. Lin, A. K. Jain and J. Zhou, *Transfer learning in deep reinforcement learning: A survey*, 2022. arXiv: 2009.07888 [cs.LG].

[7] K. Hornik, M. Stinchcombe and H. White, 'Multilayer feedforward networks are universal approximators', *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989. DOI: 10.1016/0893-6080(89)90020-8.

[8] J. Schulman, F. Wolski, P. Dhariwal, A. Radford and O. Klimov, *Proximal policy optimization algorithms*, 2017. eprint: arXiv:1707.06347.

[9] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus and N. Dormann, 'Stable-baselines3: Reliable reinforcement learning implementations', *Journal of Machine Learning Research*, vol. 22, no. 268, pp. 1–8, 2021. [Online]. Available: http://jmlr.org/papers/v22/20-1364.html.

[10] J. Schulman, *Proximal policy optimization*, Sep. 2020. [Online]. Available: https://openai.com/blog/openai-baselines-ppo/.

[11] R. S. S. Oliver G. Selfridge and A. G. Barto, 'Training and trackig in robotics', in *Proceedings of the Ninth International Joint Conference on Artificial Intelligence,*, 1985, pp. 670–672.

[12] W. Zhang, L. Deng, L. Zhang and D. Wu, 'A survey on negative transfer', *IEEE/CAA Journal of Automatica Sinica*, vol. 10, no. 2, pp. 305–329, 2023. DOI: 10.1109/jas.2022.106004.

[13] V. Kurin, S. Nowozin, K. Hofmann, L. Beyer and B. Leibe, *The atari grand challenge dataset*, 2017. arXiv: 1705.10998 [cs.AI].

[14] D. Silver, A. Huang, C. J. Maddison *et al.*, 'Mastering the game of go with deep neural networks and tree search', *Nature*, vol. 529, no. 7587, pp. 484–489, 2016. DOI: 10.1038/nature16961.

[15] B. Baker, I. Akkaya, P. Zhokhov *et al.*, *Video pretraining (vpt): Learning to act by watching unlabeled online videos*, 2022. arXiv: 2206.11795 [cs.LG].

[16] A. A. Rusu, S. G. Colmenarejo, C. Gulcehre *et al.*, *Policy distillation*, 2016. arXiv: 1511.06295 [cs.LG].

[17] S. Narvekar, B. Peng, M. Leonetti, J. Sinapov, M. E. Taylor and P. Stone, *Curriculum learning for reinforcement learning domains: A framework and survey*, 2020. arXiv: 2003.04960 [cs.LG].

[18] *Weapons*. [Online]. Available: https://vampire-survivors.fandom.com/wiki/Weapon.

[19] *Enemies*. [Online]. Available: https://vampire-survivors.fandom.com/wiki/Enemy.

[20] C. Zhang, O. Vinyals, R. Munos and S. Bengio, *A study on overfitting in deep reinforcement learning*, 2018. arXiv: 1804.06893 [cs.LG].