Rotation, Translation, and Cropping for Zero-Shot Generalization

Chang Ye
Game Innovation Lab
New York University
Brooklyn, USA
c.ye@nyu.edu

Ahmed Khalifa
Game Innovation Lab
New York University
Brooklyn, USA
ahmed@akhalifa.com

Philip Bontrager
Game Innovation Lab
New York University
Brooklyn, USA
philipjb@nyu.edu

Julian Togelius
Game Innovation Lab
New York University
Brooklyn, USA
julian@togelius.com

***Abstract*—Deep Reinforcement Learning (DRL) has shown impressive performance on domains with visual inputs, in particular various games. However, the agent is usually trained on a fixed environment, e.g. a fixed number of levels. A growing mass of evidence suggests that these trained models fail to generalize to even slight variations of the environments they were trained on. This paper advances the hypothesis that the lack of generalization is partly due to the input representation, and explores how rotation, cropping and translation could increase generality. We show that a cropped, translated and rotated observation can get better generalization on unseen levels of two-dimensional arcade games from the GVGAI framework. The generality of the agents is evaluated on both human-designed and procedurally generated levels.**

***Index Terms*—generalization, reinforcement learning, representation, A2C, zero-shot generalization, gvgai**

I. Introduction

The way in which a problem or data is represented has a large effect on how easy it is to be learned by a machine learning method. For example, it is common knowledge that when trying to learn features expressed as categorical variables, it makes all the difference in the world whether this is presented to the algorithm as a one-hot encoding or as different values of a single input. With a one-hot encoding, learning might work as intended, whereas the other encoding has much less chance of working.

This kind of knowledge, crucial as it may be, is commonly not the subject of a paper of its own, but introduced as a detail in papers focusing on some other method. For example, in the recent AlphaStar paper, one of the components was the use of a transformer architecture to process variable-length lists of units in the game [1].

The ability to learn straight from pixels, using the same information as people, is one of the reasons deep reinforcement learning has became so popular [2]. Yet, learning from pixels poses a number of challenges.In some ways, these outputs are not obvious for humans either; a human that does not know how to play videogames need to first learn that they "are" the character that moves around which they can control with the joystick, and if they play a game where the player character moves non-holonomically (such as a char) they need to learn that a particular direction of the joystick means something different depending on which way the player agent is facing. These conventions are largely carried over between games, which explains why people can so rapidly pick up a new game and start playing. However, just observe someone who does not know the perception and control conventions of a game genre try to play a game of that genre, to understand how non-obvious the conventions are. Which makes it even more surprising that deep reinforcement learning agents can learn to play these games so well.

Fig. 1: VizDoom's first-person perspective.

Over the last few years, several papers have started questioning what deep neural networks that learn to play from pixels actually learn. In "Playing Atari with Six Neurons", it is shown that surprisingly small networks with a few hundred parameters can learn to play many Atari games with a skill that rivals that of networks with hundreds of thousands, or even millions, of neurons [3]. This is accomplished by separating out the preprocessing and learning a library of sensory prototypes, allowing for an input that encodes how similar a particular observation is to other observations. This work questions what a giant neural network actually does if the policy can be encoded by a tiny neural network; perhaps most of the network is engaged in some kind of simple transformation of the input image?

Another way to investigate what deep networks, trained with reinforcement learning, learn from pixel data is through studies of their generalization capacity [4, 5]. These studies generally have rather negative results. For example, a set of experiments showed that networks trained at one or a small set of levels could not play other levels of the same game they

978-1-7281-4533-4/20/$31.00 ©2020 IEEE

had been trained on [6]. Training using procedural content generation, where each episode uses a new level, managed to create networks that generalized somewhat better, but not much better.

Anecdotal evidence suggests that this failure of generalization extends to various environments with a static third-person perspective[1]. On the other hand, games such as Doom (shown in figure 1), which are seen from the vantage point of the agent, do not seem to create the same generalization problem for deep reinforcement learning.

This paper builds on the hypothesis that deep reinforcement learning cannot easily learn generalizable policies for games with static third-person views, but that they can do so when the same game is seen through a more agent-centric view. It tests this hypothesis by training deep networks with reinforcement learning on multiple different games, with and without various perceptual modifications, in particular rotation and translation. For each version, we report performance on a training set of levels and, separately, on a larger test set.

It should be pointed out that while the hypothesis advanced here might seem obvious and the experiments somewhat simplistic, the hypothesis runs counter to received wisdom and implicit assumptions in the mainstream of deep reinforcement learning research. We are saying that deep reinforcement learning on games with static third-person representations, in general, does not work in the sense that it does not learn generalizable policies. This may or may not be because these network structures cannot learn the types of input transformations that are necessary for generalizable policies. In any case, we imply that input representation plays a much larger role than is commonly assumed.

II. Background

Deep Reinforcement Learning has a lot of success in video games, especially arcade video games like the atari environment [2]. But, as has been mentioned, these works primarily focused on learning to play mostly deterministic environments [7]. With the goal of teaching agents a real understanding of an environment so that it can be robust and useful outside of simple video game situations, there has been a lot of research recently in improving the zero-shot and few-shot generalization ability of RL agents.

It isn't immediately obvious from the research that agents are bad at generalization. Results from agents such as AlphaStar [1], OpenAI's Hide and Seek [8], and agents from the Doom competition [9] all make it appear that agents can learn general policies that adapt very well to many situations. Without discussing how general these learned policies are, we note that there are key components in these environments that would help with generalization. Both Starcraft and Hide and Seek are environments that allow self-play between competing teams. This provides a natural curriculum for an agent to learn a robust and general policy. The other component is the observation space provided in these environments. In VizDoom and Hide and Seek the agent is shown an agent-centric view of the world, making it easier to see how the agents' actions affect the world. Hide and Seek also provides extra information about the global world state. For AlphaStar, the agent is shown a minimap of the entire game but it is also provided key summary information about the objects in the game world, which allow the agent to once again have immediate feedback on how its actions are affecting the environment.

Attention has only recently been growing over the difficulty most game environments provide for learning general policies. To combat this, a number of new environments and benchmarks have been released to provide test grounds for how easily each algorithm can learn a general policy. Of the many that have been introduced, several big ones are; Coin Run [4], Obstacle Tower [10], General Video Game AI (GVGAI) [11], and Maze Explorer [12]. These environments focus on having lots of games and levels in order to provide the training data for an agent to learn a general policy. With a large number of different environments to train on, the agent cannot simply memorize a sequence of actions to take for every environment. These frameworks are very useful but agents do not automatically learn general policies even in the presence of unlimited new levels for a simple game as found by [6]. While there have been some promising results on these environments, it must be noted that this is not a solved problem.

To make matters worse, there are many situations where it is impossible to generate a large number of different versions of an environment for training, so it is important for agents to learn as general a policy as possible from a small set of environments. To help with this, researchers have found ways to inject noise into the training process. Even in the original Atari Deep Q-Learning work, they would have the agents take a random number of noop (no action) steps at the beginning of a game to randomize the initial layout of the game [2]. This would prevent the agent from simply memorizing a sequence of actions without reacting to the environment. Another simple approach to increase noise during training is sticky actions [13]. Sticky actions introduce a parameter which is the probability that an agent's action will be repeated instead of a new action is being calculated. This introduces randomness into training and forces the agent to learn a policy that is not too brittle. More recently [14] experimented with smarter ways to introduce noise for the agent's actions. Instead of injecting noise when an agent is taking an action, which results in the agent collecting worse data, the agent collects data and noise is added during the network update. This helps the agent learn an improved policy but it still relies on lots of training levels.

Focusing more on the visual aspect, [15] proposes adding visual noise into the game environment. This requires them to know which part of the frame is from the background and which part is from objects. They then replace the background with either gaussian noise or video frames from the natural

[1]With a static third-person perspective, we mean one which does not change depending on the movement of the agent, or only does so rarely (for example, when moving between rooms in a flick-screen fashion).

world. With this noise, the agent's performance plummeted showing that they could not learn a general policy that ignored the background area of the screen. To further the understanding around this, [5] examined how pixels in a state observation can provide unnecessary information that agents use to memorize a brittle policy.

Recent work that has come out after this work has focused on data augmentation, in particular image translation, as a generalization technique. In two consecutive works, researchers proposed using random augmented observations both with and without additional loss functions during training [16, 17]. The simple techniques they used achieved the state-of-the-art performance on the majority environments on the DeepMind Control benchmarks. Kostrikov et al proposes using augmented observations to reduce the variance of Q-function estimation, so as to stabilize the training process [18]. Specifically, they apply a set of data augmentation techniques K times, and then calculate an average Q value estimation to reduce the variance. The success of these approaches along with our results suggest there is a connection between an agent's point-of-view and that point-of-view's data-augmentation characteristics.

III. Generalization Approach

In this work, we are looking at improving the agent's generalization through modifying the input representation and with no data augmentation or transfer learning. We are taking a step toward a better understanding of the optimal representation for reinforcement learning. In environments where the agent is embodied in the environment and shown a map of the entire world, it really struggles with learning from a static third-person point of view [6]. The network not only needs to learn the consequences of its actions but also needs to track a small blob of pixels to know its location. We propose to transform the input representation to be more centered around the agent as it is seeing it from its point of view. Doing that will reduce the number of tasks the agent needs to learn during training.

We propose always giving the agent an agent-centric[2] view when possible and further propose that cropping the agent's view to just its immediate surroundings can greatly improve its ability to learn in Deep Reinforcement Learning (DRL). We propose three techniques to do this that can be applied to any environment with a visible agent even if the only state information available is a pixel image. We propose rotating, translating, and cropping the observation around the agent's avatar. These are quick transformations that can be applied to the observation image and they only require knowledge of the location and the direction of the agent's avatar. If the location information isn't available from the environment, a simple object detection algorithm can be used to find the avatar image on the screen. For the avatar's direction, if relevant to the environment, it can be extracted from the agent's actions as the agent usually need to change its direction before it starts changing its location. Rotation keeps the agents always facing forward, so any action it takes always happens in the same relative direction to it. Next, translation translates the observation around the agent so its always in the center of its view. Finally, cropping shrinks the observation down to just local information around the avatar.

These changes at first can appear like obvious transformations, but we did not find anywhere in the literature discussing how observation perspective affects learning for DRL. We not only recommend these techniques for people working with map-like views, but also measure their effectiveness and discuss where and when they are useful. A local, agent-centric view, allows for better learning in our experiments and the policies learned generalize much better to new environments even when trained on only five environments. This implies the agent was able to learn from the correct objects in the environment instead of just memorizing states from pixels.

In practice we also found it was necessary to randomize the agent's initial orientation, and to replace the agent's avatar with a square. We believe the randomization is necessary to stop the agent from memorizing an opening sequence. We believe the agent was using the transformed avatar's layout to determine its global orientation, but also found that the agent performed better in every single experiment if the avatar was replaced.

In the following subsections, we are going to explain in detail these three transformations. These transformations can be used by themselves or combined together to combine their effects.

A. Translation

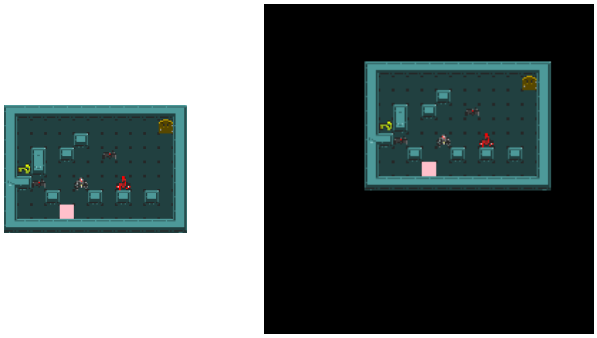


Fig. 2: The left observation is being translated around the player's avatar (pink rectangle) to the right observation.

Translation is the process of centering the observation image around the player's avatar. The idea is the player's avatar should always be in the center of the observation image after this transformation. Figure 2 shows the translation process where the observation appearing on the left was padded with black pixels and centered around the player's avatar (pink rectangle). We can see that the center pixel of the new observation is the player's avatar. Translation will restrict the avatar to learn the relative position to other game objects such as a key, door, or enemy in figure 2 [19] which is useful for

[2]Note that "agent-centric" is not the same as "first-person". With an agent-centric view, we mean any view that puts the representation of the agent (a.k.a. its avatar) at the center; in the examples here, we use a third-person agent-centric view.

the agent to understand the game and take the corresponding action. In a lot of video games, having a relative position is enough to win the game as you usually need to move the avatar toward a certain target to interact with it.

B. Rotation

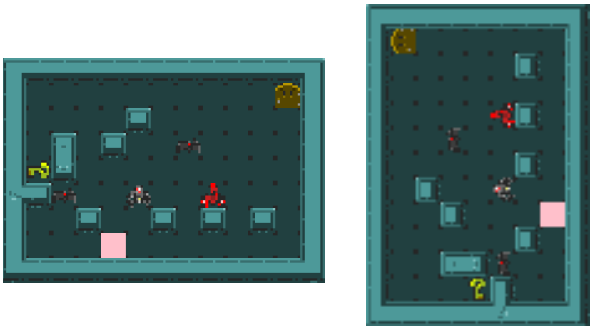


Fig. 3: The left observation is being rotated using the player's avatar (pink rectangle). In the original the player is looking right, the observation is rotated so that the agent is always looking up.

Rotation is the process of orienting the observation to face the same direction as the player's avatar. Figure 3 shows the rotation transformation on the left observation (the original observation). The rotated observation shown on the right is the original observation rotated towards the player direction (which is right in that state). An important note when using rotation transformation, the action space the agent is taking has to be unrotated before it is returned to the framework. For example: in figure 3 the avatar is facing right so the observation was rotated by $90°$. If the avatar wants to move up, in the new observation this is technically right in the original framework. To solve that problem, any action happening that is taking place in the environment has to be rotated in the negative direction of the rotation degree. Rotation helps the agent to learn navigation as it simplifies the task. For example: if you want to reach for something on the right, the agent just rotates until that object is above and then moves up. If that object becomes to the left, the same strategy can be applied (rotate then move up). This is not the case without rotation where we need to move in a different direction depending on the location of the target object.

C. Cropping

Cropping is the process of only showing the observation around the player and not the full observation. Cropping by default is a translation technique as the new observation is centered around the avatar. Figure 4 shows a 5x5 cropping transformation being applied to the left observation (the original game observation) to a smaller view that is centered around the player (the center pixel is the center of the avatar).

The cropping helps to reduce the state space of what the agent is seeing to a smaller subset which can help the agent to learn a generalized policy. Neural Nets can be interpreted

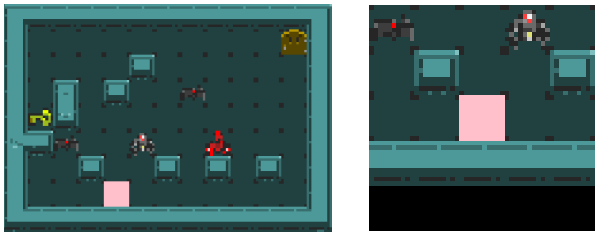


Fig. 4: The left observation is being cropped using the player's avatar (pink rectangle) position to the right observation.

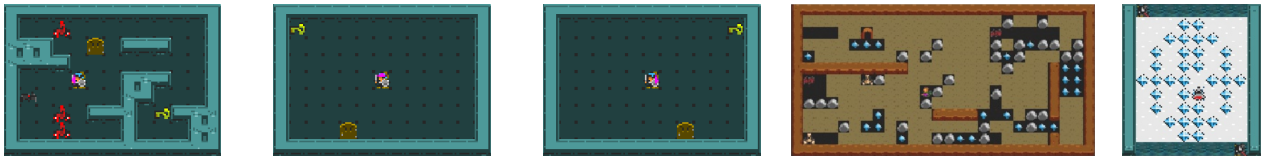
as behaving as Locality-sensitive hashing (LSH) functions and they intelligently learn to recognize similar states. Larger environments with many combinatorial arrangments of agent and object locations make it more difficult for the agent to understand what states are functionally the same. In a cropped view the agent can directly see the effect of its action, assuming its actions are local, and it can directly match actions to states.

In many video games actions and interactions mostly happen locally, cropping focuses the agent to this area. This helps in many games but obviously also provides a disability in the form of missing information and lack of a global context. Another reason is that cropping observation could be considered as a data augmentation which is helpful for generalization as it learns a broader set of state-action values, also referred to as the Q-values [20].

IV. Experimental Methods

In this work, we use the OpenAI Gym [21] interface of the GVGAI Framework [11]. We test our techniques on three different games and one game variant:

- **Zelda:** is a GVGAI port for the dungeon system in The Legend of Zelda (Nintendo, 1986). The goal of the game is to get a key and reach the exit door while avoiding hitting enemies. The agent also can use its sword to kill enemies for additional scores. Figure 5a shows an example of a human-designed Zelda level.
- **Simple Zelda:** similar to the Zelda game but it only has a key and the door. The agent's goal is to get the key and reach the door. In this game, there are no walls so the agents don't need to learn navigation. It just needs to learn the goal of the game. Also, all the game levels are designed such that the player starts in the center of the map and the key and door are both either on the left of the agent or the right shown in figures 5b and 5c.
- **Boulderdash:** is a GVGAI port for Boulder Dash (Data East, 1984). The goal is to collect 10 different diamonds then reach the goal while avoiding getting killed by enemies or the falling boulders. Figure 5d shows an example of one of the training levels in Boulder Dash.
- **Discrete Solarfox:** is an adapted version of a GVGAI port for Solarfox (Midway Games, 1981). The goal of the game is to collect all the diamonds without hitting the borders of the map or enemy bullets. A complication

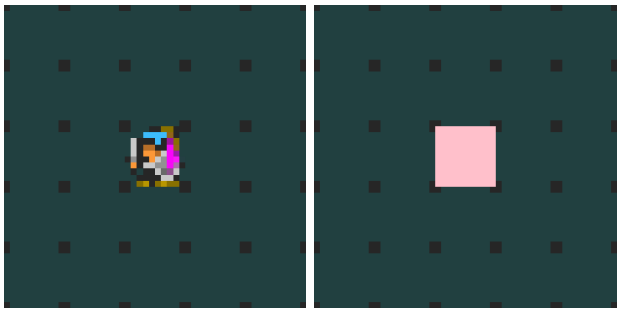


(a) human-designed zelda level (b) simple zelda training level (key and door always on the left) (c) simple zelda test level (key and door always on the right) (d) human-designed boulderdash level (e) human-designed solarfox level

Fig. 5: Examples of game levels from zelda, simple zelda, boulderdash, and solarfox.

is that the avatar is always moving; if no new input is given, it keeps moving in the same direction as the last frame. We modified this game by increasing the avatar speed by factor of 7 as the framework only returns the avatar location in integer values (while actual speed was $1/7$ in the original game). Figure 5e shows an example of one of the training levels in Solarfox where the player controls the spaceship.

To evaluate our methods. We employ the *Advantage Actor-Critic* (A2C) algorithm. Specifically the implementation from Open AI Baselines [22]. The neural network has the same structure as in Mnih et al. [23] with a body consisting of three convolutional layers followed by a single fully-connect layer. We trained the agents until convergence (which took between 200 million frames to 400 million frames). We configured the A2C algorithm to use a step size 5, 84 by 84 wrapped frame, 4 frames per stack and a constant learning rate of 0.007 with the RMSProp optimizer.



(a) Original view (b) View with replacing avatar

Fig. 6: Examples of observation with or without replacing avatar

In preliminary experiments, we found that agents trained on Simple Zelda levels barely won on the test set. When we investigated the playtraces we found that the agent is simply memorizing to go left (where the key and the door are in the training set) instead of understanding where the key and the door location are and trying to move toward them. This was still happening in the cropped view with rotation and translation, where the agent has no idea where it could find the key and the door and doesn't have an idea what is left (because of rotation). we discovered that the agent uses the avatar rotated pixels to memorize the solution for different levels similar to Song et al's work [5] where the agent used the scoreboard to solve the game. To avoid that problem, we simply replace the avatar with a square of a certain color shown in figure 6.

For Zelda, we train the agent on the 5 human-designed levels and test it on a different 45 human-designed levels. While for Simple Zelda, we train the agent on levels where the key and the door are on the left side of avatar and test it on levels where the key and the door are on the right side of the avatar. The idea behind that is to test generality in its most simple form where the agent needs to understand where to go and not just memorize the sequence of actions to win the level. The number of levels in train and test set is the same which is 1190 levels which reflect all the possible levels where the key and the door are assigned on either side of the avatar. For Boulderdash and Solarfox, the agent is trained on the 5 human-designed levels that come with the framework and tested on 50 different generated levels using the generator from Justesen et al.'s work [6]. For all the experiments, the avatar starts with a random direction uniformally sampled from all four directions to allow the agent not to memorize the starting direction. Similar to the noop random initialization in Mnih et al. [2] work.

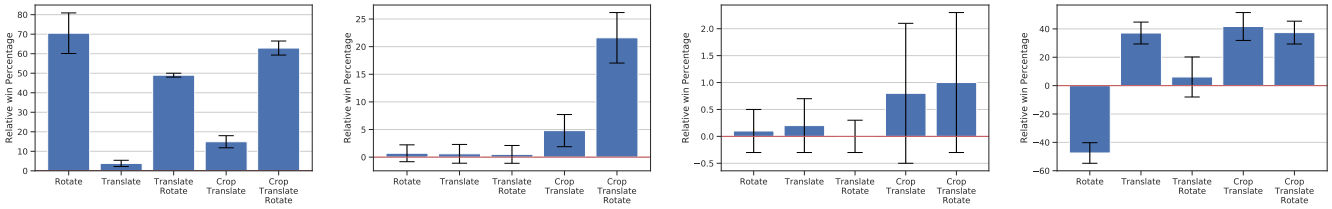
We trained 3 models for all the possible combinations of our proposed transformations (Translation, Rotation, and Cropping) on all the proposed problems (Simple Zelda, Zelda, Boulderdash, and Solarfox). We end up with having 6 total experiments instead of 8 because you can't do Cropping without Translation. For the avatar's location, we extract it from the game engine itself which will be replaced in future work with a simple OpenCV image tracking function.

V. Results

For each trained model we test it for 20 times on every level in the training set and test set. Table I shows the results from concatenating these data by showing the mean and the standard deviation between the three different models. The low standard deviation shows the model stability during the training process where it achieves almost the same results. Figure 7 shows the results of all the algorithms on the test set as a relative performance with respect to the original model performance on the test set. Positive values indicate that this transformation is helpful to the system, while negative values indicate that this transformation is hurting the system, and near-zero values means they are no different.

crop	translate	rotate	simple zelda		zelda		boulderdash		dsolarfox	
			train	test	train	test	train	test	train	test
0	0	0	$100.0 \pm 0.1\%$	$0.0 \pm 0.0\%$	$76.7 \pm 17.6\%$	$0.4 \pm 0.8\%$	$19.0 \pm 29.1\%$	$0.0 \pm 0.0\%$	$87.7 \pm 15.1\%$	$49.0 \pm 7.1\%$
0	0	1	$100.0 \pm 0.0\%$	$70.5 \pm 10.4\%$	$81.0 \pm 14.8\%$	$1.1 \pm 1.3\%$	$12.3 \pm 11.6\%$	$0.1 \pm 0.4\%$	$29.5 \pm 17.8\%$	$1.5 \pm 1.5\%$
0	1	0	$100.0 \pm 0.1\%$	$3.8 \pm 1.6\%$	$78.7 \pm 18.6\%$	$1.0 \pm 1.5\%$	$30.0 \pm 12.6\%$	$0.2 \pm 0.5\%$	$87.0 \pm 14.5\%$	$86.1 \pm 3.0\%$
0	1	1	$100.0 \pm 0.0\%$	$49.0 \pm 1.0\%$	$75.0 \pm 19.2\%$	$0.9 \pm 1.4\%$	$28.3 \pm 11.7\%$	$0.0. \pm 0.3\%$	$74.0 \pm 16.9\%$	$55.1 \pm 12.2\%$
1	1	0	$100.0 \pm 0.0\%$	$14.9 \pm 3.1\%$	$66.7 \pm 21.8\%$	$5.2 \pm 2.8\%$	$8.3 \pm 12.3\%$	$0.8 \pm 1.3\%$	$95.7 \pm 9.0\%$	$90.7 \pm 6.8\%$
1	1	1	$99.9 \pm 0.1\%$	$62.9 \pm 3.6\%$	$65.7 \pm 19.7\%$	$22.0 \pm 4.5\%$	$10.7 \pm 12.4\%$	$1.0 \pm 1.3\%$	$85.5 \pm 14.8\%$	$86.4 \pm 3.8\%$

TABLE I: Test and train result of different combination in simple zelda, zelda, boulderdash and discrete solarfox



(a) Simple Zelda (b) Zelda (c) Boulderdash (d) Discrete Solarfox

Fig. 7: Relative performance of different transformations with respect to the original model performance on the test set

In the simple Zelda, all the experiments scored 100% win rate on the training set which was not surprising as the task is pretty simple and there is no navigation obstacles. Looking on the test set, the cropped, rotated and translated observation achieves the second highest win rate on test levels. Surprisingly, the rotated observation achieves the best win rate on the test set. But the first and second win rate are close due to the high variance in the rotated observation test. Followed by having rotation and translation in the third place. Looking at the rest of the experiments, we find that the agent struggles on the test levels without having the rotation.

In Zelda, we found that the overall performance on both training set and test set are dropped, especially on the test set. We think that this performance drop is due to the complexity of this game as it has more tasks (navigation through walls and avoiding randomly moving enemies) that it needs to master. Also, the training set size is a lot smaller compared to the simple Zelda experiment. All the trained agents are having similar good performance on the training set with not a huge difference but on the test set the gap is big. The cropped, rotated and translated observation achieves the highest win rate across all approaches. Surprisingly, all the other experiments perform pretty badly especially if it doesn't have cropping. We believe the agent simply overfit to the training levels due to the small training set and couldn't figure a general strategy. On the other hand, when the observation was cropped, the new observation might be more general and more frequently appearing in other levels which helped it to generalize better.

In Boulderdash, the agent struggles to learn to play the game well on the training set which didn't help it to work on the test set with a 0% win rate. We think this bad performance is because the agent doesn't have any indication about how many diamonds it has collected so far, and it might be impossible to make sure it collects all the diamonds. (On some levels, it is impossible to collect all the diamonds.) This can be noticed from the slightly lower performance of the agent on the training set when it has cropping compared to the rest of the experiments. We think either having a visual indicator or adding memory to the agent might improve its ability to learn and play the game.

In Discrete Solarfox, the agent learns to play the game pretty good on the training set in most of the cases except with having rotation only. On testing, it is clear that translation is the key component towards generalizing in that game. We think it is due to the nature of that game and the need of relative locations between the avatar and different game objects to perform well in the game. The avatar needs to be far away from the edges of the screen and enemy bullets, while being close to gems to collect them. The case of having only rotation might made it harder for the agent to extract these informations from the running game.

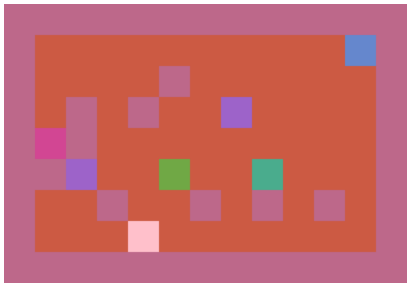


Fig. 8: Obvservation with every object replaced, it did not improve generalization.

As mentioned several times, replacing the avatar with a square was shown to always improve testing performance significantly. Extrapolating from this, it seems further removing orientation information by replacing every object in the game with a square would further assist in generalization (see Figure 8). However, the results are similar to simply replacing the avatar. This is ideal as only replacing the avatar is much easier than replacing every object when only pixel data is available.

VI. Discussion

Cropping, rotation and translation improve the generalization. However, the win rate is still not promising on more complex problems (Zelda and Boulderdash). We think that could be because of the small training set, the small capacity of our network (3 convolutional layers and 1 fully-connected layer), or the need for memory. From Cobbe's work [24], larger structures such as IMPALA-CNN [25] significantly improve generalization comparing to the structure we used in the project which we could adopt for future work.

The cropping, rotation and translation could be used in many types of games, However, the techniques have some trade-offs and limits. The cropping will throw away global information. This problem is not affecting the performance in Zelda and Solarfox because the actions in these 2 games only affect relative objects with no global effect on the environment. In other games with global effects, cropping might not work. For instance, imagine a scenario where the avatar needs to kill an enemy before it gets stronger, if we use the cropped view, the avatar might not be able to see if there is an enemy to go and kill it before it gets stronger. The rotation technique is based on the assumption that the avatar has a direction it's facing. Therefore, this technique cannot be applied to the games without this property.

The experiments also expose the weakness of the current neural network structure that we are using. The neural network is not always focusing on the area we want it to focus such as the objects in the surrounding. It focuses on tiny object details which was the reason to replacing the avatar in all our experiments. A similar situation also happens in Song et al's work [5]. The agent focus on the scoreboard instead of the object we want them to focus. By blacking out the scoreboard, the performance on generalization is significantly improved. However, the neural network itself should be able to figure out how to focus on important areas like the selective filtering [26] in human's visual system and the attention mechanism [27, 28]. It could be a future research direction on generalization.

As mentioned above, the neural network simply could be viewed as a LSH function. Another interest research direction is combining LSH with feature extraction techniques such as autoencoder to test whether it can achieve similar performance comparing to the regular neural networks.

VII. Conclusion

This work demonstrates the importance of an agent's perspective when learning. Our three proposed simple changes make a big difference in the policies that the agent learns. This highlights how little is still understood about what causes an agent to learn brittle or robust policies in deep reinforcement learning. This work advances the state-of-the-art for zero-shot generalization as well as formalizes some deep learning tribal knowledge on how to design useful state observations.

The results demonstrate the importance of all three transformations: rotation, translation, and cropping. Giving the agent a narrow, agent-centric view, where it's always facing forward allows it to more accurately learn the effect of each of its actions and the effect of the environment on it. Training on only five levels, it is then able to beat up to 90% of the new levels it had never seen before in a highly stochastic game. That is a huge improvement over what has been possible with so little data.

For future work, we would like to continue to test these generalization effects on different games. We would like to continue and improve our understanding of the effects of each of these transformations. It is also important to test these techniques on games where the actions have larger effects on the game state and/or the global game information has more influence on the win rate than local information does. This would give more insight into the efficacy of these techniques in a more diverse set of situations. Finally, since the data-augmentation is a side-effect of our techniques, we would like to apply random data augmentation techniques. Instead of hard coding the augmentation techniques, we could adopt a similar model to Ha and Schmidhuber's World Models [29]. Specifically, we could apply random data augmentations to the input of the vision model so that the model could learn a better representation, similar to the recently published Network Randomization[30]. All of these refinements should help everyone's understanding of some of the real factors that allow for robust policies in some environments and impossible situations in others.

Acknowledgments

Ahmed Khalifa acknowledges the financial support from NSF award number 1717324 ("RI: Small: General Intelligence through Algorithm Invention and Selection."). The authors thank Per Josefsen and Nicola Zaltron, who created for the 45 human-designed Zelda levels.

References

[1] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev *et al.*, "Grandmaster level in starcraft ii using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, 2019.

[2] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, 2015.

[3] G. Cuccu, J. Togelius, and P. Cudré-Mauroux, "Playing atari with six neurons," in *Autonomous Agents and Multi-Agent Systems*. IFAAMAS, 2019.

[4] K. Cobbe, O. Klimov, C. Hesse, T. Kim, and J. Schulman, "Quantifying generalization in reinforcement learning," in *International Conference on Machine Learning*, 2019, pp. 1282–1289.

[5] X. Song, Y. Jiang, S. Tu, Y. Du, and B. Neyshabur, "Observational overfitting in reinforcement learning," in *ICLR*, 2020.

[6] N. Justesen, R. R. Torrado, P. Bontrager, A. Khalifa, J. Togelius, and S. Risi, "Illuminating generalization

in deep reinforcement learning through procedural level generation," in *Deep RL Workshop NeurIPS 2018*, 2018.

[7] N. Justesen, P. Bontrager, J. Togelius, and S. Risi, "Deep learning for video game playing," *Transactions on Games*, 2019.

[8] B. Baker, I. Kanitscheider, T. Markov, Y. Wu, G. Powell, B. McGrew, and I. Mordatch, "Emergent tool use from multi-agent autocurricula," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=SkxpxJBKwS

[9] M. Wydmuch, M. Kempka, and W. Jaśkowski, "Vizdoom competitions: Playing doom from pixels," *Transactions on Games*, vol. 11, no. 3, 2018.

[10] A. Juliani, A. Khalifa, V.-P. Berges, J. Harper, E. Teng, H. Henry, A. Crespi, J. Togelius, and D. Lange, "Obstacle tower: a generalization challenge in vision, control, and planning," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. AAAI Press, 2019, pp. 2684–2691.

[11] R. R. Torrado, P. Bontrager, J. Togelius, J. Liu, and D. Perez-Liebana, "Deep reinforcement learning for general video game ai," in *Computational Intelligence and Games*. IEEE, 2018.

[12] L. Harries, S. Lee, J. Rzepecki, K. Hofmann, and S. Devlin, "Mazeexplorer: A customisable 3d benchmark for assessing generalisation in reinforcement learning," in *Conference on Games*. IEEE, 2019.

[13] M. C. Machado, M. G. Bellemare, E. Talvitie, J. Veness, M. Hausknecht, and M. Bowling, "Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents," *Journal of Artificial Intelligence Research*, vol. 61, 2018.

[14] M. Igl, K. Ciosek, Y. Li, S. Tschiatschek, C. Zhang, S. Devlin, and K. Hofmann, "Generalization in reinforcement learning with selective noise injection and information bottleneck," in *Advances in Neural Information Processing Systems*, 2019.

[15] A. Zhang, Y. Wu, and J. Pineau, "Natural environment benchmarks for reinforcement learning," *arXiv preprint arXiv:1811.06032*, 2018.

[16] M. Laskin, K. Lee, A. Stooke, L. Pinto, P. Abbeel, and A. Srinivas, "Reinforcement learning with augmented data," *arXiv preprint arXiv:2004.14990*, 2020.

[17] A. Srinivas, M. Laskin, and P. Abbeel, "Curl: Contrastive unsupervised representations for reinforcement learning," *arXiv preprint arXiv:2004.04136*, 2020.

[18] I. Kostrikov, D. Yarats, and R. Fergus, "Image augmentation is all you need: Regularizing deep reinforcement learning from pixels," *arXiv preprint arXiv:2004.13649*, 2020.

[19] C. Kwok and D. Fox, "Reinforcement learning for sensing strategies," in *International Conference on Intelligent Robots and Systems*, vol. 4. IEEE, 2004.

[20] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3-4, 1992.

[21] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," 2016.

[22] P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, Y. Wu, and P. Zhokhov, "Openai baselines," 2017.

[23] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, 2016.

[24] K. Cobbe, C. Hesse, J. Hilton, and J. Schulman, "Leveraging procedural generation to benchmark reinforcement learning," *arXiv preprint arXiv:1912.01588*, 2019.

[25] L. Espeholt, H. Soyer, R. Munos, K. Simonyan, V. Mnih, T. Ward, Y. Doron, V. Firoiu, T. Harley, I. Dunning *et al.*, "Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures," in *ICML*, 2018.

[26] D. E. Broadbent, *Perception and communication*. Elsevier, 2013.

[27] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[28] X. Wang, R. Girshick, A. Gupta, and K. He, "Non-local neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7794–7803.

[29] D. Ha and J. Schmidhuber, "Recurrent world models facilitate policy evolution," in *Advances in Neural Information Processing Systems*, 2018, pp. 2450–2462.

[30] K. Lee, K. Lee, J. Shin, and H. Lee, "Network randomization: A simple technique for generalization in deep reinforcement learning," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=HJgcvJBFvB