Are Gradient-based Saliency Maps Useful in Deep Reinforcement Learning?

Matthias Rosynski
Department of Production Engineering
University of Bremen, Bremen, Germany
m_rosynski@gmx.net

Frank Kirchner
Department of Computer Science
University of Bremen, Bremen, Germany
frank.kirchner@dfki.de

Matias Valdenegro-Toro
German Research Center for Artificial Intelligence
Bremen, Germany
matias.valdenegro@dfki.de

Abstract

Deep Reinforcement Learning (DRL) connects the classic Reinforcement Learning algorithms with Deep Neural Networks. A problem in DRL is that CNNs are black-boxes and it is hard to understand the decision-making process of agents. In order to be able to use RL agents in highly dangerous environments for humans and machines, the developer needs a debugging tool to assure that the agent does what is expected. Currently, rewards are primarily used to interpret how well an agent is learning. However, this can lead to deceptive conclusions if the agent receives more rewards by memorizing a policy and not learning to respond to the environment. In this work, it is shown that this problem can be recognized with the help of gradient visualization techniques. This work brings some of the best-known visualization methods from the field of image classification to the area of Deep Reinforcement Learning. Furthermore, two new visualization techniques have been developed, one of which provides particularly good results.
It is being proven to what extent the algorithms can be used in the area of Reinforcement learning. Also, the question arises on how well the DRL algorithms can be visualized across different environments with varying visualization techniques.

1 Introduction

Due to the success achieved in recent years by Deep Reinforcement Learning [7] [2], Industrial applications are becoming increasingly tangible [13]. Research is being conducted on 3D map reconstruction for autonomous cars where DRL can be one of the solutions [18] and also in the field of AUVs [3] as well as many other applications that interact with the real world. Once DRL algorithms are implemented on physical systems and interact with the real world, these systems can be dangerous for themselves and humans. For this reason, debugging tools are needed to understand why the agent behaves that way and whether the agent is making the right decision for the correct reason and not making a right decision for the wrong reason [11].

Deep reinforcement learning algorithms are nowadays interpreted and measured by the rewards the agents can get. For this reason these agents are called black box algorithms and are criticized. This makes them difficult to use in critical real-world applications.

This paper reveals that visualization techniques are a powerful debugging tool, that provides much more information than interpreting rewards. With the help of Guided Backpropagation it is even

1st I Can't Believe It's Not Better Workshop (ICBINB@NeurIPS 2020), Vancouver, Canada.

possible to locate the error in a certain layer or stream in a neural network. Moreover it is shown that Grad-Cam methods can deliver results very early in training in case of very badly trained networks. This is a big advantage especially for off-policy algorithms that take a long time to explore at the beginning of the learning process. With off-policy algorithms it can take a few days until the rewards start to increase so that the developer can determine if the neural network is learning at all [14].

Contributions. For DRL, the claim made by Adebayo et al. [1], that Guided Backpropagation does not visualize the desired regions, but through partial input recovery it works as a kind of edge detector, is shown not to be accurate. Also, the claim that gradient methods can be difficult to interpret, because, when answering the question "What perturbation to the input increases a particular output?", gradient methods can choose perturbations which lack physical meaning [5], is shown not always to be an issue.

Furthermore two new visualization techniques were developed, one of which provides particularly good results. These were compared and analysed with 4 other popular visualization techniques. Their advantages and disadvantages are discussed and different fields of application for the respective visualizations are suggested.

2 Related Work

In this section, previous work is presented and discussed. As already mentioned, there is currently not a lot of work dealing with the topic of visualization in the area of deep reinforcement learning algorithms. First two works from the field of deep reinforcement learning are presented and then and then visualization techniques from the field of image processing.

Semi Aggregated Markov Decision Processes. The authors which introduced Semi Aggregated Markov Decision Processes [17] used the Atari 2600 environments as interpretable testbeds, they developed a method of approximating the behavior of deep RL policies via Semi Aggregated Markov Decision Processes (SAMDPs). They used the more interpretable SAMDPs to gain insights about the higher-level temporal structure of the policy. From a user perspective, an issue with the explanations is that they emphasize t-SNE clusters and state-action statistics which are uninformative to those without a machine learning background.

Perturbation-based saliency methods. Another recently published work shows Perturbation-based saliency methods for the visualization of learned policies by Greydanus et al. [5]. Their approach is to answer the question, "How much does removing information from the region around location change the policy?". The authors defined a saliency metric for image location as:

$$\mathcal{S}_\pi(t,i,j) = \frac{1}{2}\left\|\pi_u\left(I_{1:t}\right) - \pi_u\left(I'_{1:t}\right)\right\|^2 \tag{1}$$

The difference $\pi_u\left(I_{1:t}\right) - \pi_u\left(I'_{1:t}\right)$ can be interpreted as a finite differences approximation of the directional gradient $\nabla_{\hat{v}}\pi_u\left(I_{1:t}\right)$ where the directional unit vector $\hat{v}$ denotes the gradient in the direction of $I'_{1:t}$.

In other words, is looking at how important were these pixels for the policy. By checking how strong the policy changes after removing some informations from the image. They use the same approach to construct Saliency Maps for the value estimate V^π too. Greydanus et al. [5] claims that gradient-based saliency methods do not yield well interpretable results. When answering the question "What perturbation to the input increases a particular output?", gradient methods can choose perturbations which lack physical meaning.

Gradients. The basic idea of backpropagation-based visualizations is to highlight relevant pixels by propagating the network output back to the input image space. The intensity changes of these pixels which have the most significant impact on network decisions. Despite its simplicity, the results of saliency map are normally very noisy which makes the interpretation difficult. [8].

Guided Backpropagation. The idea behind guided backpropagation is that neurons act like detectors of particular image features. It is interesting in what image features the neuron detects and not in what kind of features it doesn't detect. That means when propagating the gradient, the ReLu function set all the negative gradients to zero. [8] With other ways we only backpropagate positive error signals and we also restrict to only positive inputs.

Gradient-weighted Class Activation Mapping. One of the problems when it comes to using CAM is that a modern neural network not only consists of convolution layers but of different layers such as LSTM, MaxPooling, etc. GradCAM is an extension of CAM and it is broadly applicable to any CNN-based architectures. In order to obtain the class-discriminative localization map Grad-CAM $L^c_{\mathrm{Grad-CAM}} \in \mathbb{R}^{u \times v}$ of width u and height v for any class c. First, the gradients y^c of class c are calculated up to the desired convolution feature map activations A_k. An average pooling over the width and height dimensions is then carried out [9].

$$\alpha_k^c = \overbrace{\frac{1}{Z}\sum_i \sum_j}^{\text{global average pooling}} \underbrace{\frac{\partial y^c}{\partial A_{ij}^k}}_{\text{gradients via backprop}} \tag{2}$$

Weight α_k^c represents a partial linearization of the deep network downstream from A, and captures the "importance" of feature map k for a target class c. On the end it performs a weighted combination of forward activation maps with a ReLU fuction. [9]

$$L^c_{\mathrm{Grad-CAM}} = \mathrm{Re}\, LU \underbrace{\left(\sum_k \alpha_k^c A^k \right)}_{\text{linear combination}} \tag{3}$$

The result is a heatmap of the same size as the convolutional feature map. This feature map must then be enlarged to the size of the original image and place it over the image to get the final result. If Grad-Cam is multiplied by Guided Backpropagation with the Hadamard product we get Guided Grad-Cam [9].

Gradient Methods. Adebayo et al. [1] deals with the informative value of visualization techniques. In this work, various visualization techniques were compared and checked whether the techniques actually depend on the model, the training data or whether it partially reconstructs the image. In other words, their work claims that Guided Backpropagation and Guided GradCam act like an edge detector, which means that they show the desired positions without showing the learned model.

In their work they randomize the weights of a model starting from the top layer, successively, all the way to the bottom layer. This procedure destroys the learned weights from the top layers to the bottom ones. Their results indicate that Guided backpropagation and Guided GradCam do not visualize the learned model, but instead partially reconstruct the image. They interpret their findings through an analogy with edge detection in images, a technique that requires neither training data nor model.

Laplacian Operator in Image Processing. To understand one of the results we will derive the kernel of the Laplacian filter which is used for edge detection. One approach for the design of directionally invariant high-pass filters for image processing (also called edge detectors) is to use second-order derivation operators, e.g. the Laplace operator [16]. For continuous functions, the Laplace operator is defined by:

$$\mathbf{L} = \frac{\partial^2 I[x,y]}{\partial x^2} + \frac{\partial^2 I[x,y]}{\partial y^2} \tag{4}$$

We approximate the partial derivatives by difference equations and thus obtain :

$$\frac{\mathrm{d}^2\mathrm{I[x,y]}}{\mathrm{dx}^2} = \{\mathrm{I[x+\Delta x,y]} - 2\mathrm{I[x,y]} + \mathbf{I}\mathrm{[x-\Delta x,y]}\}/\Delta\mathrm{x}^2 \tag{5}$$

$$\frac{\mathrm{d}^2\mathrm{I[x,y]}}{\mathrm{dy}^2} \approx \{\mathrm{I[x,y+\Delta y]} - 2\mathrm{I[x,y]} + \mathrm{I[x,y-\Delta y]}\}/\Delta\mathrm{y}^2 \tag{6}$$

Thus with $\Delta x, \Delta y = 1$ (except for the sign) the mask for the Laplace operator is :

$$\mathbf{L}_1 = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \tag{7}$$

Numerous other approximations of the Lapace operator are possible. Examples are the following masks (their transfer functions have the form known from the Gaussian and binomial distributions)[16]

$$\mathbf{L}_2 = \begin{bmatrix} 0 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & 0 \end{bmatrix} \quad \mathbf{L}_3 = \begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix} \quad \mathbf{L}_4 = \begin{bmatrix} -1 & -2 & -1 \\ -2 & 12 & -2 \\ -1 & -2 & -1 \end{bmatrix} \tag{8}$$

3 Experimental Setup

Experiments were performed in two different environments a simple (Breakout-v0) and a complex one (Seaquest-v0) from OpenAI Gym [4]. Four different agents were implemented. DDDQN (4 frames as input) [10], Splitted Attention DDDRQN (with LSTM), and two on Policy gradient algorithms A3C (3 frames as input) [6] and an A3C Agent with LSTM.

The Splitted Attention DDDRQN was by far the best trained agent and receives most of the points (Seaquest-v0 9521 Points) that is why most of the results are referring to this agent [12].

In order to better examine the visualization methods and to enable better interpretation, the original frames of the games are placed over the gradients with a 50% opacity. In this way it is possible to assign the gradients to the features and to better interpret the results.

With the off policy algorithms not only the output is visualized but also the Q-value and the Advantage stream. To proof the results of Ziyu Wang et al. [15]. In order to maintain comparability between the visualization techniques, the same state was always visualized in this work. This also applies to the visualization of the actor and critic agents, as well as the visualization of the value and advantage streams. Two new visualization techniques have been developed and the following visualization methods have been implemented. Some features of the implementation are discussed below.

Gradient and Guided Backpropagation. Compared to image processing, where the gradients of the guided backpropagation or gradient method are normalized over the image, the gradients in a video can also be normalized over the entire video. For this reason, both have been implemented and tested. First, the gradients output were normalized for each state. Then the gradient and guided back propagation method around the whole video was normalized and tested.
In the visualizations, the gradients were checked in relation to different output layers. In particular, this work examines the advantage and the value stream in the off-policy algorithms in more detail, as well as the last layer that delivered the best results. In the actor critic methods both outputs of the NN were examined.

Grad-Cam and Guided Grad-Cam. With the Grad-Cam and Guided Grad-Cam method, the visualization can be applied to different convolutional layers. An example of what the differences look like is also shown. The best results were achieved on the first convolutional layer across all agents and across all environments. For this reason, the results of the Grad-Cam methods are usually applied to the first convolutional layer.

G1Grad-Cam and G2Grad-Cam. Two new visualization techniques were developed and will be presented in this work, which are also compared. Both visualization techniques are a further development of Grad-Cam and Guided Grad-Cam. The idea behind it is the same as for guided back propagation, that neurons act like detectors of particular image features. And we are interested in what image features the neuron detects and not in what kind of features it does not detect. That means when propagating the gradient, we set all the negative gradients to 0. In the further course, Grad-Cam with the Guided Model is called G1Grad-Cam and GradCam with the Guided Model multiplied by Guided backpropagation is called G2Grad-Cam.

4 Experimental Results

Since a state usually contains several frames, the results of the gradient methods (unless other-wise stated) are related to the last frame in the sequence. With the Grad-Cam methods the results (unless otherwise stated) relate to the first convolutional layer. Furthermore, the word *stable* is defined in the context of visualization techniques in this paper as follows: gradients or visual highlights that can be

seen on most frames in a video. Gradients or visual highlights that can be seen on every 4th, 5th (or even less) frame in a video are called *not stable*.

First we discuss the visualization techniques, then we look closer at the class discriminative visualization algorithms and finally we make a hypothesis about the importance of negative gradients in backpropagation algorithms. Detailed results are all provided in the appendix (Figures 2 to 78).

4.1 Visualization Algorithms

Gradient. Gradients method delivers a lot of noise which often mixed with the important features. It is only very poorly suited for debugging neural networks in the deep reinforcement learning area. We get similar results with the agent in all environments. This can be seen in Figures: 2, 13, 14, 21, 29, 31, and 50.

Guided Backpropagation. Guided backpropagation delivers the best and most stable results under all environments and among all agents. Also, that it shows negative gradients is a very good indication of how well an agent is trained. It can be seen that the better the agent has been trained, the stronger the transition between negative and positive gradients.

Another interesting observation that guided back propagation (compared to the Grad-Cam methods) did not visualize the agent so well during the breakout game. Even in the DDDQN network, where the agent was well trained, only weak gradients could be recognized on the agent himself on Figures 19 and 3. In contrast, on Seaquest, which is a much more complex environment and the agent was not well trained, very strong gradients were displayed on the agent itself, as Figures 13 and 14 show.

Another comparison that has been made in this work is how the visualization of the guided backpropagation algorithm changes with respect to normalization. In other words, how does the visualization change when the normalization of the gradients is carried out over a frame compared to when the normalization was created over the entire video.

The results showed that when the frame was normalized, the gradients in the video flashed more strongly. With normalization across the entire video, the transitions from the frames were smoother. In this way, the less known / visited states can be recognized because the gradients on which are weaker to recognize. This method is also the most suitable for visualizing the differences between the advantage and the value stream in the off-policy algorithms. Guided back propagation was able to identify most of the features in any environment and these were also very stable across the video.

During the development of the split attention DDDQN agent, the guided back propagation method for debugging was also used. An interesting finding that was made during development was that although the neural network had several errors, results were still very good (over 3000-3500 points, similar results were achieved by the normal attention DDDRQN network). Only after the Advantage and Value Stream was visualized did it become apparent that the network only learned on the Advantage side and that the gradients on the value side were very strong but chaotic. Based on the points, the errors that were made during programming would not have been noticed. Altogether two errors could be discovered and they could even be localized in the value stream. One mistake was incorrectly linking the layer and another when merging the two streams.

When developing the A3C with LSTM network, guided back propagation was also used to debug the network. Before a working A3C with LSTM was developed in this work, attempts were first made to train the agent with bidirectional layers. This was unsuccessful. In the visualization, no gradients were displayed when the gradients from the output layer to the input layer were calculated (the output gradients during printing were also 0). Then the network with guided backpropagation was examined more precisely with this visualization technique, because in contrast to the Grad-Cam methods, the gradient methods calculate the gradients from each individual layer to the input, this means that the individual layers can be examined. It turned out that the agent started to learn in the first three layers and no longer from the bidirectional layers. When these layers were replaced by LSTMs, the agent immediately began to learn even in the higher layers.

Grad-Cam. This method also showed stable results (based on the first convolutional layer), even if the results were often issued in inventory, especially with less well-trained agents, as seen in Figure 46.

Looking at the higher layers, some features could also be visualized, but here the visualization was inverted more often, as shown in Figure 41, and the specified position of the features in the visualization algorithm was less precise, shown in Figures 33 and 37.
One advantage over the guided backpropagation method which is noticeable, is that with less well trained agents, the visualization (even if mostly often inventoried) was able to visualize the features faster and without interference. However the G1Grad-Cam could beat these results.

Grad-Cam was able to achieve better results in the Breakout environment (see Figure: 4), where the agent could be visualized relatively poorly with guided backpropagation. Another advantage of the grad cam method (at least for neural networks without LSTM) compared to guided backpropagation is that, as can be seen in Figure 4, the time flow can be visualized. Means how important the previous frames were.

Due to the negative gradients that exist in guided backpropagation, a superimposition or averaging of the gradients from all input frames would not provide a similar result, since the gradients would cancel each other out with guided backpropagation.

Since the Grad-Cam has a ReLU function, only positive results are displayed. However, this has the disadvantage that we cannot differentiate as well if an agent is well trained. The visualization differences are minimal as can be seen in the Actor Critic between Figures 69 and 70 and different between the Splitted Attention DDDRQN (See Figure 23) and the DDDQN Agent . However, the results with Grad-Cam are better with poorly trained agents and have less disruption than with guided backpropagation. One reason for this could be that the grad-cam carried an average pooling across the height and width of the dimension are carried out. With average pooling, the tendencies could be displayed better and there would be fewer disturbances due to averaging. This could be the reason why by Grad-Cam by weakly trained networks achieve better results than with guided backpropagation.

A disadvantage compared to guided back propagation, which can be seen in all agents is that not all features can be visualized. By the breakout environment the area of the image where the agent breaks through the last line and receives many points could be not visualized (see Figure 3 and Figure 4). In Seaquest, Grad-Cam was unable to visualize the oxygen bar, nor the divers collected.

Guided Grad-Cam and G2Grad-Cam. The combination of guided back propagation and one of the Grad-Cam methods has proven to be very unstable. One reason is that not all features are displayed in the Grad-Cam method, as has already been mentioned. But the main reason is that the results are shown often inverted by the Grad-Cam method. Due to the inversion, the areas with the features have a value of 0, which means that the multiplication with the guided back propagation also results in 0.

Basically, with these two methods, no additional knowledge can be obtained that has already been obtained with guided back propagation or Grad-cam.

G1Grad-Cam The G1Grad-cam method gave much better results for the DDDQN agent in the game breakout and also the time flow than with the Grad-Cam method, as shown in Figure 6 (G1Grad-Cam) versus Figure 4 (Grad-Cam). Interestingly, the G1Grad-Cam method has problems visualizing the features in game Seaquest.

The G1Grad-Cam method shows less inverted results than the Grad-Cam method. However, especially with less well-trained agents, the results are not stable or not exactly in position. Basically, the method only convinced the DDDQN agent in the game Breakout. Here G1grad-Cam was able to display the ball and the agent as well as their past positions very clearly than all other visualization techniques.

However, the upper breakthrough area could not be visualized as the guided back propagation method did.

An interesting aspect of the poorly trained splitted attention agent is that in Seaquest the G1Grad-Cam method gives the best interpretable results over all visualization methods, as shown in Figure 47 (in contrast to the well trained one, where it gives no results). The neural network was saved after 325 episodes (for comparison, the well-trained neural network was saved after 5300 episodes). After 325 episodes the neural network has done about 250 000 steps and 75% of them have chosen a random action (because of exploration). After this short period of time, the G1Grad-Cam method was able to deliver well interpretable results.

For comparison: After 1500 episodes it can be determined via the rewards that the neural network is learning. This means that with the G1Grad-Cam method it is possible to recognize up to 5 times faster if the neural network is learning at all. This makes it a helpful debugging tool.

4.2 Action Discriminative Visualization Algorithms

The fact that the agent has learned to recognize features does not mean that the agent has learned how to behave. The agent recognizes e.g. the oxygen bar and understand when it ends that the episode will end, but it has not learned what to do about it. It is similar with the fishes that the agent recognizes them but does not necessarily mean that the agent will avoid the fish or knows what to do with it. For this reason, action discriminative algorithms were examined more closely.

The investigations did not give any indication that Grad-Cam and G1Grad-Cam, which are action discriminative algorithms, have a visible visual relationship between the state and the action taken. The expectation that Grad-Cam would emphasize a particular fish because the agent would swim away from a fish or target a fish, could not be confirmed.

4.3 Hypothesis about Negative Gradients in Guided Backpropagation and their Meaning

Apparently negative gradients also play a role. With the DDDQN agent, the negative gradients were time-dependent. The newest frame has the most positive and the oldest most negative gradients and the negative gradients are always shown where the ball was or will be. One reason for this could be the different architecture of the neural network which only calculates the error of the selected action but not the error of the non-selected actions compared to the other developed CNN. With the other agents, it can be observed that the negative gradients did not form in a time-dependent but position-dependent manner. The strong negative gradients have formed around the features. It can also be seen that the stronger the positive and negative gradients, the better the agent.

A possible explanation could be that with guided backpropagation the gradients resemble a matrix or the kernel of an edge detector. And the Laplacian Operator has strong positive and negative values in the matrix to detect edges. Backpropagation could create a matrix that has learned to better identify features with strong negative and positive values, just like with a Laplace operator. Figure 1 shows that strong positive gradients are followed by strongly negative gradients and some small positve again (compare in Equations 7 and 8, Equation L_1 and L_3). The Laplace operator has a similar structure. The difference is that the Laplace operator detects edges and not more complex features. Guided backpropagation would create a kind of Laplace matrix for features based on this hypothesis. This could be a possible explanation for the role of negative gradients.

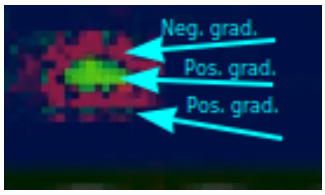

Figure 1: The agent is covered with positive gradients followed by negative and positve again.

5 Conclusions and Future Work

Visualisation techniques. First of all, the assumptions made in the paper "Visualizing and Understanding Atari Agents" [5], that guided backpropagation cannot be used for visualization techniques and the assertion made in the paper "Sanity Checks for Saliency Maps" [1], that guided backpropagation and guided Grad-Cam (at least in image processing) do not visualize the learned model but work similarly to an edge detector do not seem apply to deep reinforcement learning policies.

Visualization methods are very well suited as an additional, if not main debugging tool. Especially Guided Backpropagation was used during the development of Splitted Attention DDDQN with bidirectional LSTM agents and the A3C with LSTM agent to detect various errors in the neural network. Not only can errors be detected but also an assignment could be made in which stream the error was located and which layer caused problems. This kind of debugging is not possible with the usual reward evaluation. Especially since the faulty neural network could achieve better results than the original one despite the errors it contains, these errors would not have been noticed.

Furthermore, it was also shown with two agents (Figure 68 and Figure 16) that Guided backpropagation can be used to detect if the agent has learned to avoid an area. The agent obtained a high reward,

but this is not the expected behavior. The agent has learned to stay in the water at a certain height. The agent always shoots to the left or to the right if something comes up there. This is an important insight because it means that if the environment changes the agent will probably not be able to cope with these changes as well as an agent that has a lower score, but swims specifically towards the fishes. Here a simple reward function could even lead to misinterpretations. Guided backpropagation could show the different strength of the gradients on the fish that swam at different heights, which could lead to the conclusion that the agent does not visit certain areas of the environment.

The Grad-Cam and especially the G1Grad-Cam method achieves very early results. This is useful during the training process of the neural network. For example, if an agent has to be trained for 14 days in a HPC cluster, G1Grad-Cam can quickly provide initial information after a few minutes / hours as to whether the agent recognizes any features at all or whether there are indications of errors in the neural network. The rewards that are currently used as the main debugging tool are hardly usable at the beginning, especially with off-policy algorithms. Because the agent is exploring at the beginning, the rewards are random, but the neural network learns to recognize the features even if it does not yet know what they mean or how the agent has to behave. This is a big time advantage over the Reward function that took 2 days with the Split Attention Agent to show the first signs that the Rewards are increasing.

The error does not necessarily have to be in the neural network, but also a faulty implementations can be detected in this way (during this work wrong stacking of the frames could be visualized on the neural network and the error could be corrected by the DDDQN agent). The G1Grad-Cam and Grad-Cam method can save time when designing neural networks during debugging compared to a pure interpretation by the Reward function.

However, no evidence was found that using G1/Grad-Cam methods which are action discriminative methods, can visualize the difference in importance between the features, that has the biggest impact on the action (agent's decision). No feature was highlighted that is more important for the decision of an action.

Guided backpropagation was able to recognize most of the features. All features detected by the Grad-Cam methods and more. Even though the gradients are sometimes hardly visible and only after long training the DDDQN agent could visualize itself in the game Breakout.
This method is especially suitable to evaluate the agent at the end (Does the agent recognize all important features? Are there differences in intensity between the same features in different states? How strong are the gradients on the features?) or for debugging purposes. Guided backpropagation is better suited for debugging than the Grade-Cam method, because individual streams could be examined and individual layers. In this way the error can be identified more quickly under certain circumstances (as in the development of the Split Attention Agent).

Negative gradients. A hypothesis about possible explanation for the importance of negative gradients that forms around features were put forward in this work. The explanation was compared to a Laplacian filter which revealed the edges. However, edges were not revealed here, but entire features it was concluded that through backpropagation a kind of Laplacian kernel feature detector was visualized. Where the negative gradients would be produced by a second derivative to better detect the features.

Future work. After having shown in this work that visualization techniques are a basic important tool for debugging neural networks, there are still some important questions left. First of all the features that are important for the agent could be visualized. What could not be shown in this paper is that in action discriminative methods, the features that have a greater importance for the selected action are more strongly emphasized. A possible reason for this could be that by averaging all feature maps of the Grad-Cam methods the differences between the normal features and the features that directly influenced the agent's decision were very small. Here, a special implementation of Guided backpropagation might help to better identify the differences, by reprogramming the Guided backpropagation to an action discriminative algorithm. This would give a better understanding of the advantages shown in this work, that guided backpropagation can better evaluate the intensity of how well a neural network recognizes the feature. In combination with a modified action discriminative Guided backpropagation algorithm, it might be possible to get more meaningful results if it is possible to visualize the feature that had the greatest influence on the current action.

References

[1] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps, 2018.

[2] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, Nov 2017.

[3] Shane Barratt. Active robotic mapping through deep reinforcement learning. *arXiv preprint arXiv:1712.10069*, 2017.

[4] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[5] Sam Greydanus, Anurag Koul, Jonathan Dodge, and Alan Fern. Visualizing and understanding atari agents, 2017.

[6] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. *CoRR*, abs/1602.01783, 2016.

[7] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.

[8] Weili Nie, Yang Zhang, and Ankit Patel. A theoretical explanation for perplexing behaviors of backpropagation-based visualizations. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3809–3818. PMLR, 10–15 Jul 2018.

[9] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. *International Journal of Computer Vision*, Oct 2019.

[10] Mohit Sewak. Deep q network (dqn), double dqn, and dueling dqn. In *Deep Reinforcement Learning*, pages 95–108. Springer, 2019.

[11] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps, 2013.

[12] Ivan Sorokin, Alexey Seleznev, Mikhail Pavlov, Aleksandr Fedorov, and Anastasiia Ignateva. Deep attention recurrent q-network. *CoRR*, abs/1512.01693, 2015.

[13] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.

[14] Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99, pages 1057–1063. MIT Press, 1999.

[15] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado van Hasselt, Marc Lanctot, and Nando de Freitas. Dueling network architectures for deep reinforcement learning, 2015.

[16] Hanyue Liang Yilun Chen, Rui Zhu. Filter in der bildverarbeitung.

[17] Tom Zahavy, Nir Ben-Zrihem, and Shie Mannor. Graying the black box: Understanding dqns. In *International Conference on Machine Learning*, pages 1899–1908, 2016.

[18] Karel Zimmermann, Tomas Petricek, Vojtech Salansky, and Tomas Svoboda. Learning for active 3d mapping, 2017.

A Detailed List of Experiments

The agent and the environment are shown on the left and the visualization techniques on the right.

Agent	Env.	Q-Values		1st conv. Layer			
		Gradient	Guid. back.	Grad-C.	Guid. Grad-C.	G1Grad-C.	G2Grad-C.
DDDQN [10]	Breakout-v0	✓	✓	✓	✓	✓	✓
DDDQN [10]	Seaquest-v0	✓	✓	✓	✓	✓	✓
Split. At. DDDQN (bi. LSTM)	Seaquest-v0	✓	✓	✓	✓	✓	✓
A3C [6]	Breakout-v0	✓	✓	✓	✓	✓	✓
A3C [6]	Seaquest-v0	✓	✓	✓	✓	✓	✓
A3C with LSTM [6]	Seaquest-v0	✓	✓	✓	✓	✓	✓

Table 1: Combinations of visual explanation experiments

In addition, there are the different settings of the visualization techniques, e.g. which layer or stream was visualized with gradient methods or up to which convolutional layers were the Grad-Cam method applied:

Agent	Env.	Val. Stream	Adv. Stream	1st conv. L.	2st conv. L.	3st conv. L.
		Guided backpropagation		Grad-Cam		
DDDQN [10]	Breakout-v0	✓	✓	✓		
Split. At. DDDQN (bi. LSTM)	Seaquest-v0	✓	✓	✓	✓	✓

Table 2: Combinations of different settings for visual explanation experiments

The past frames in the input of the neural network for the guided backpropagation algorithm are also examined:

Agent	Env.	t-1	t-2	t-3	t-9
DDDQN [10]	Breakout-v0	✓	✓	✓	
Split. At. DDDQN (bi. LSTM)	Seaquest-v0				✓

Table 3: Combinations of time-dependent visual explanation experiments

An other question also arises: When does the neural network show first signs of feature recognition? Is it possible to find out faster if the neural network or the agent in general has been programmed correctly? Than about the interpretation of the reward function, which is only possible after a long time because of the exploration of the agent in off-policy algorithms.
For this reason, a neural network that was after only 325 episodes / 250 000 steps saved, is also tested (for comparison: the well trained agent had trained over 5300 episodes and about 13 000 000 steps):

Agent	Env.	Gradient	Guid. back.	Grad-C.	Guid. Grad-C.	G1Grad-C.	G2Grad-C.
Split. At. DDDQN (bi. LSTM)	Seaquest-v0	✓	✓	✓	✓	✓	✓

Table 4: Combinations of time-dependent visual explanation experiments

B Detailed Experimental Results

The visualized results are presented in red or green color. Meanwhile red is used for negative gradients, green has been used for positive ones. Since the Grad-Cam methods have a ReLu function, the results are always positive. All Images are taken at the same time for better comparison.

B.1 DDDQN: Breakout-v0

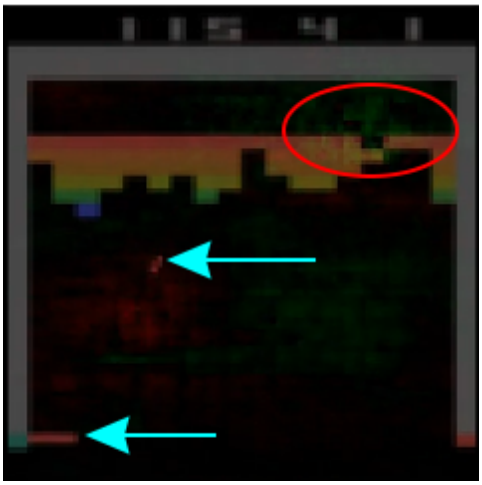

Figure 2: Gradient visualization method. In the red oval you can see that the NN-Agent started to understand which region gives him higher future rewards and where it has to shot. This is because if the agent breaks through the last line, the ball will bounce off the box and the ceiling and get a lot of reward.

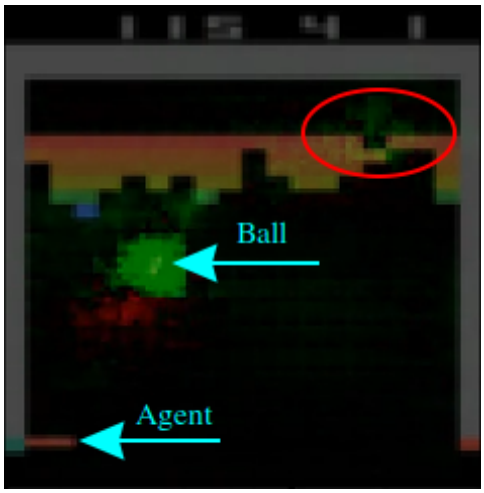

Figure 3: Guided backpropagation visualization method. The ball is with this method well heighlighted. In the red oval you can see that the NN-Agent started to understand which region gives him higher rewards and where it has to shot

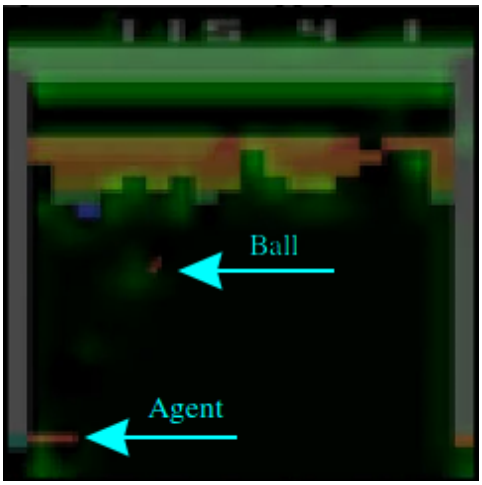

Figure 4: Grad-Cam. Some visual highlights on the agent as well as on the ball are clearly visible. Furthermore you can see the direction the ball is coming from as all 4 frames are visualized with the Grad-Cam method.

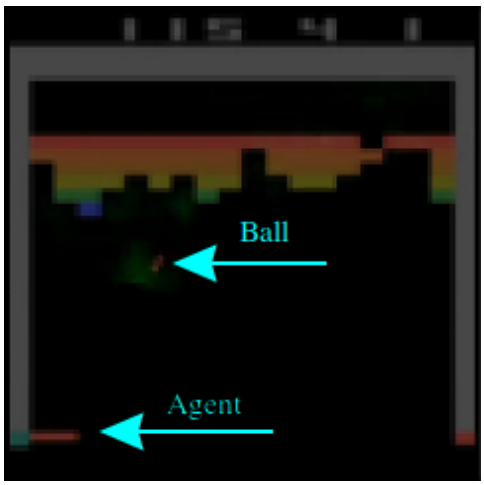

Figure 5: Guided Grad-Cam. Since Guided Grade-Cam is a multiplication of the Grade-Cam method and backpropagation, it is also obvious that we only visualize sub areas of the original source. The negatives and positive gradients of the current frame calculated by the guided backpropagation method are multiplied by the degree-cam method, which does its job on all frames. And only if both have positive high values this area will be highlighted.

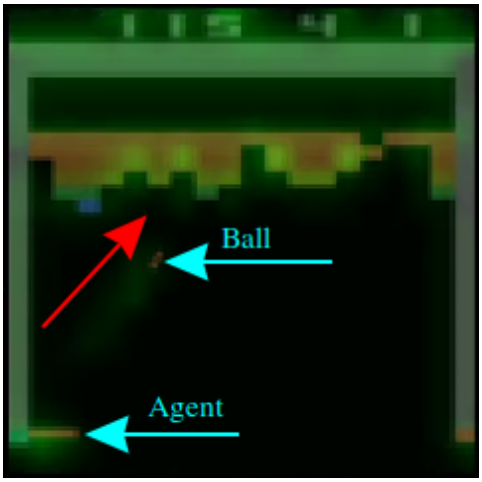

Figure 6: G1Grad-Cam. Comaprad to the Grad-Cam method we can see much more stable visualized features. We can see clearly the old positions of the ball and when the agent moves the old position of the Agent because we have four frames as input. This is different to the Gradient methods where we cant ovalep the positive and negative gradients, thats why we are using the gradient methods on the last frame.

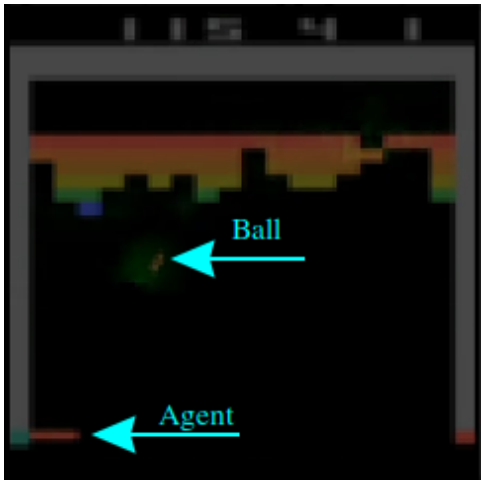

Figure 7: G2Grad-Cam. Since Guided Grade-Cam is a multiplication of the Grade-Cam method and backpropagation, it is also obvious that we only visualize sub areas of the original source. The negatives and positive gradients of the current frame calculated by the guided backpropagation method are multiplied by the degree-cam method, which does its job on all frames. And only if both have positive high values this area will be highlighted.

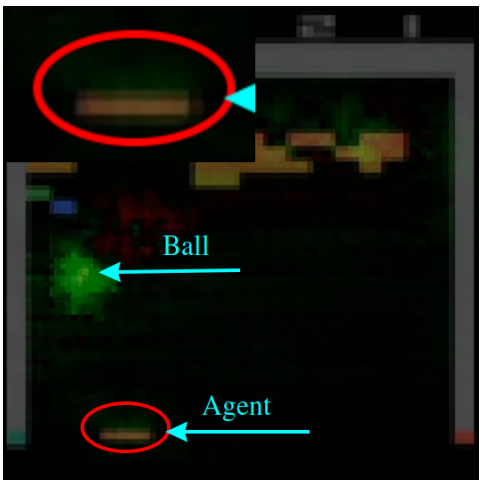

Figure 8: Guided backpropagation (advantage stream). In the advatage straem we can see that the NN is slightly more focusing on his own position than in the value stream

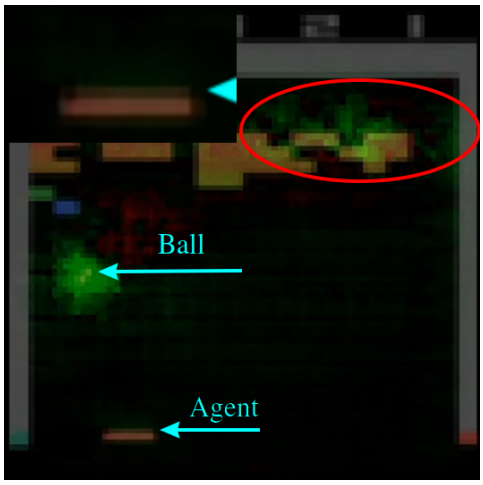

Figure 9: Guided backpropagation (value stream) . In the value stream we see that the gradients are stronger on focused on the future rewards. This is because if the agent breaks through the last line, the ball will bounce off the box and the ceiling and get a lot of reward.

Figure 10: Guided backpropagation (t-1). We can see here that the old position of the ball (frame t-1) has positive gradients, whereas the current position of the ball is very much wrapped in negative gradients.

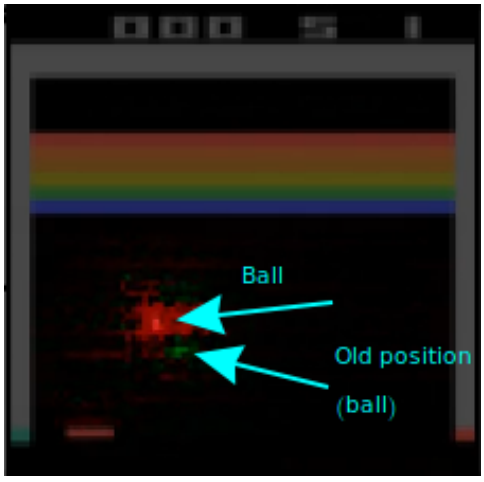

Figure 11: Guided backpropagation (t-2). We can see here that the old position of the ball (frame t-2) has positive gradients, whereas the current position of the ball is very much wrapped in negative gradients.

Figure 12: Guided backpropagation (t-3). We can see here that the old position of the ball (frame t-3) has positive gradients, whereas the current position of the ball is very much wrapped in negative gradients.

B.2 DDDQN: Seaquest-v0

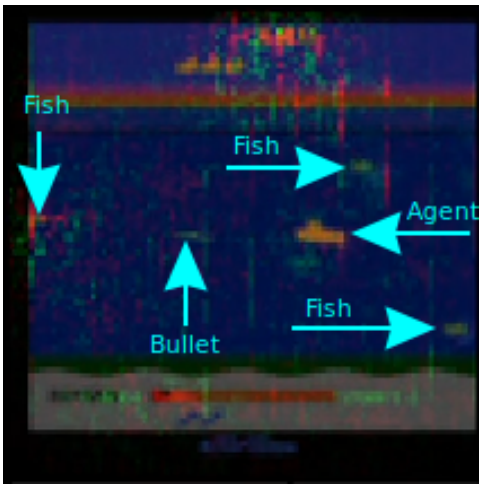

Figure 13: Gradient (advantage stream). We can see on the Agent some gradients more than on the value Stream, but in general very noisy results.

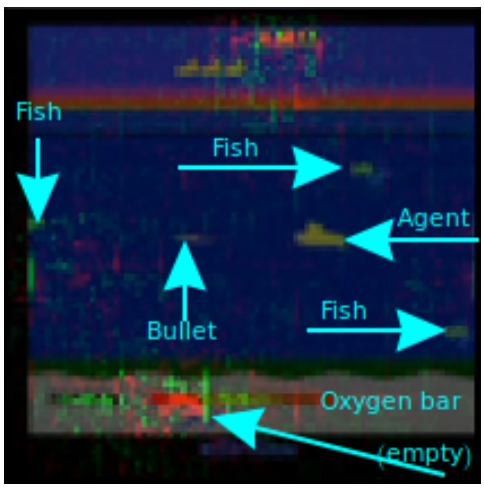

Figure 14: Gradient (value stream). We can see on the Oxygen bar some gradients more than on the advantage Stream, but in general very noisy results.

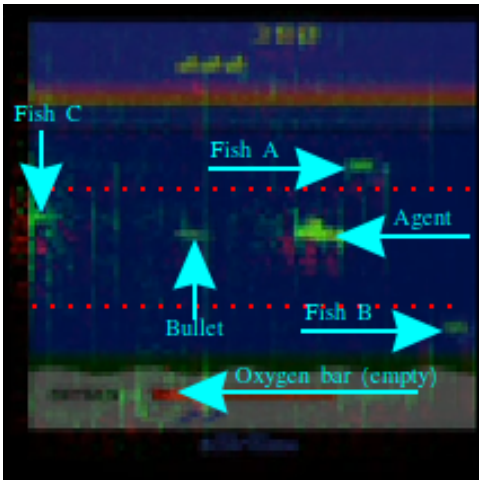

Figure 15: Guided backpropagation (advantage stream). In this picture we can see that the NN envelops more gradients around fish C than on fish B or A which have hardly any gradients. This is because the agent has learned that if it stays on the height between the red lines and only shoots to the left or right when something comes up it survives longer and pays no attention to the rest of the environment. The agent has learned his behaviour by heart and would not be able to react as well as an agent who makes less points but reacts better to the environment. If the environment were to change slightly (e.g. the fish would suddenly swim from the top right to the bottom left) the agent could hardly react to it because it ignores everything that is not in the area between the lines.

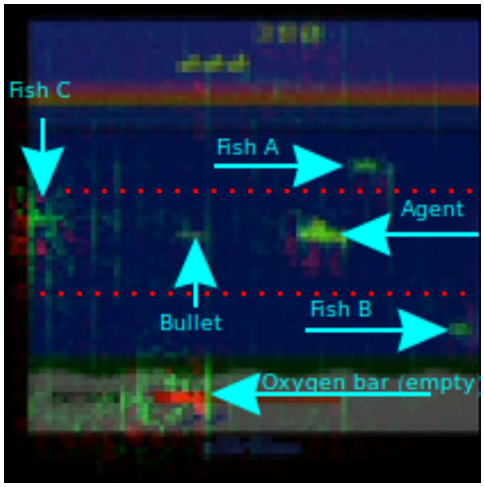

Figure 16: Guided backpropagation (value stream). In the Value Stream there is a small difference to the advatage stream. We see that the agent has no oxygen and has to appear. The DDDQN agent has never learned how to surfaced in seaquest, however he realises that the oxygen bar is an important feature that is related to the end of the episode, but the agent has not learned how to do what. And since the agent has learned to leave the area between the lines, he will not be able to learn how to do it.

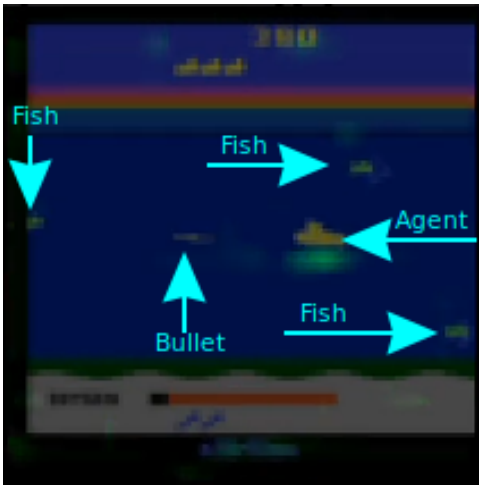

Figure 17: Grad-Cam

Figure 18: Guided Grad-Cam

Figure 19: G1Grad-Cam. Compared to Breakout-v0, the DDDQN agent did not give good results for Seaquest-v0.

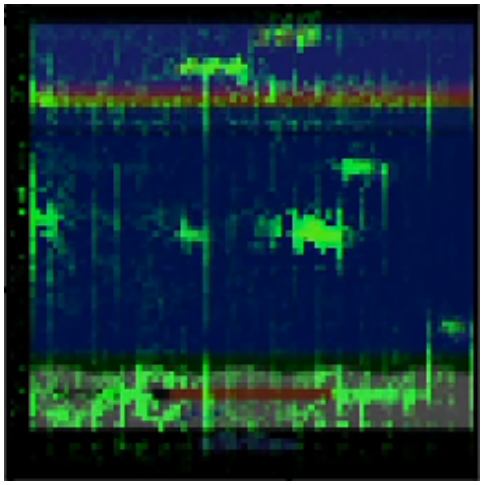

Figure 20: G2Grad-Cam

B.3 Split At. DDDQN: Seaquest-v0

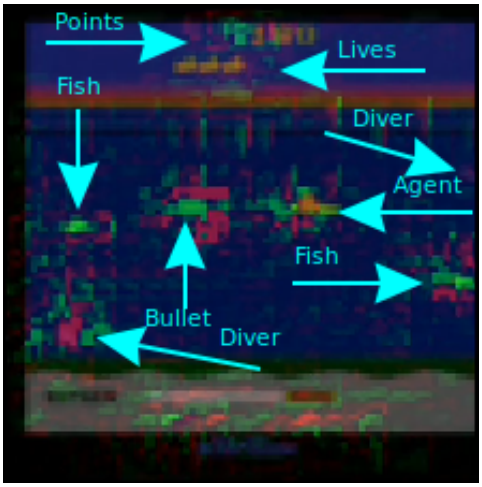

Figure 21: Gradient. Very noisy results.

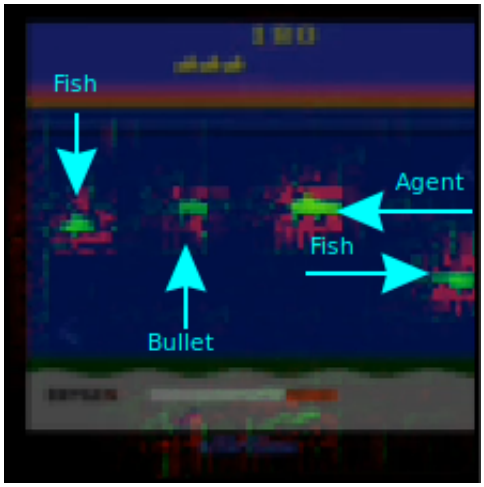

Figure 22: Guided backpropagation. Since this agent has a very well trained NN network and has achieved the best results in the game, you can can see here very clearly how the positive gradients on the features are surrounded by negative gradients.

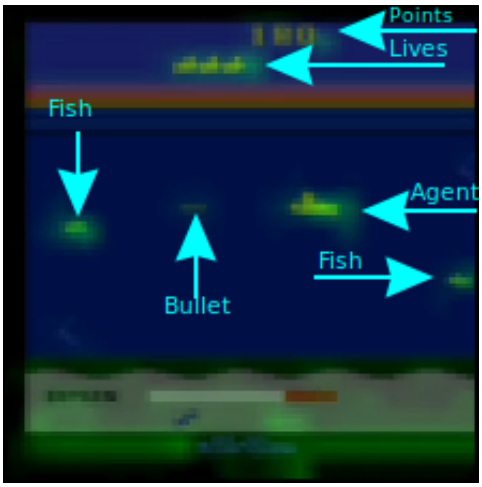

Figure 23: Grad-Cam. Good visible highlights of the most important features.

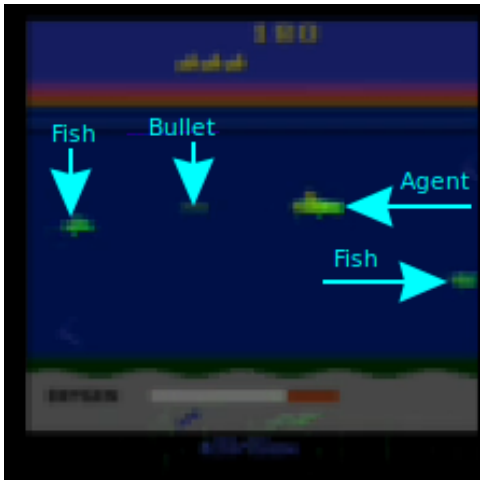

Figure 24: Guided Grad-Cam

Figure 25: G1Grad-Cam. No visible highlights over the whole video.

Figure 26: G2Grad-Cam. No visible highlights over the whole video.

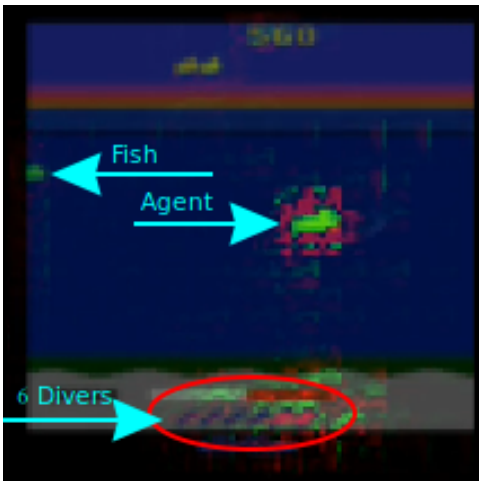

Figure 27: Guided backpropagation (advantage stream)

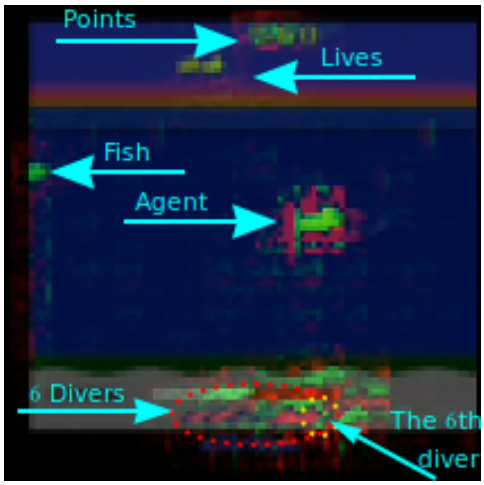

Figure 28: Guided backpropagation (value stream). The agent has learned to pay more attention to the collected divers in the Value Stream. Here he has collected all 6 divers, which means that when he shows up he will get extra reward. Furthermore you can see that the fish in the value stream has more gradients than in the advantage stream.

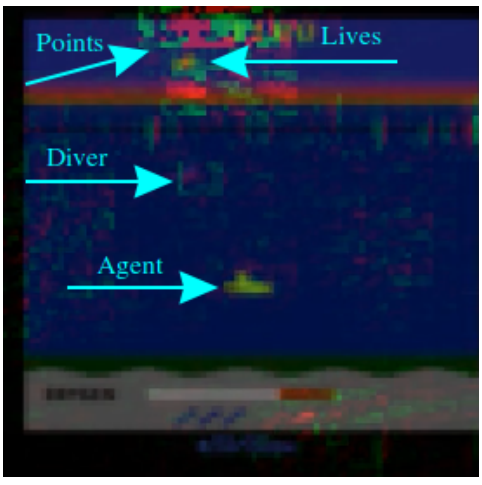

Figure 29: Gradient. In this frame we see that the agent also recognizes the divers that contain a long term reward when the agent collects 6 divers and brings them to the surface.

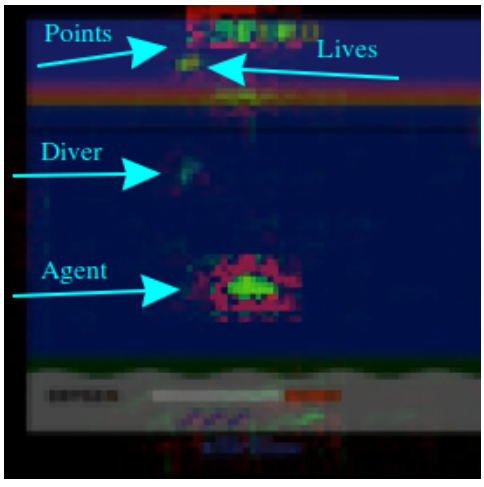

Figure 30: Guided backpropagation. In this frame we see that the agent also recognizes the divers that contain a long term reward when the agent collects 6 divers and brings them to the surface.

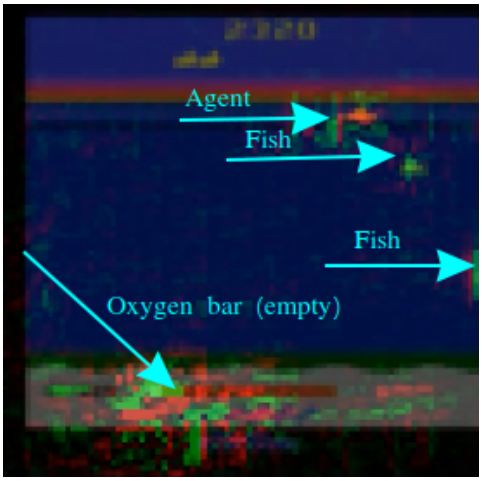

Figure 31: Gradient. Here we see that the agent with the gradient method also pays attention to the empty oxygen bar. This is also the reason why the agent appeared.

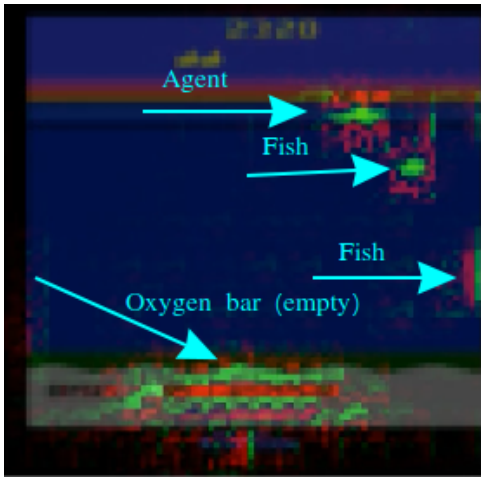

Figure 32: Guided backpropagation. In this figure we can see that the agent with the gradient method also pays attention to the empty oxygen bar - in contrast to the gradient method the oxygen bar is completely wrapped in gradients.

Figure 33: Grad-Cam 2nd convolutional layer. When we visualise the second layer, with the very well trained NN we always see no connections that we can interpret but not as well as we saw with the first layer

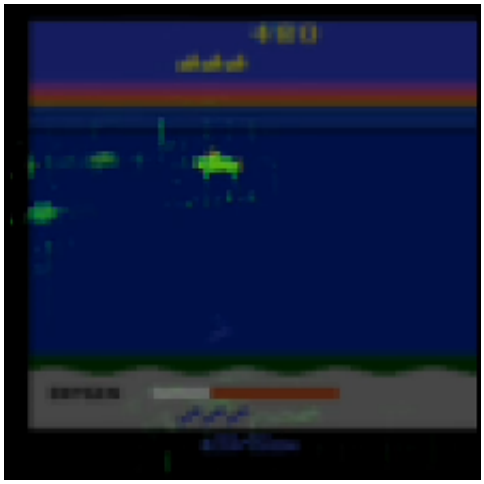

Figure 34: Guided Grad-Cam 2nd convolutional layer

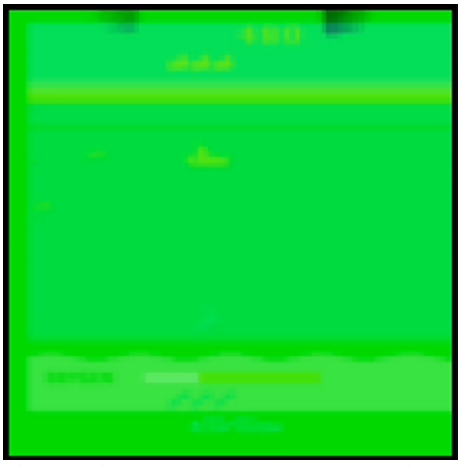

Figure 35: G1Grad-Cam 2nd convolutional layer. As in the first layer, this method does not show well interpretable results

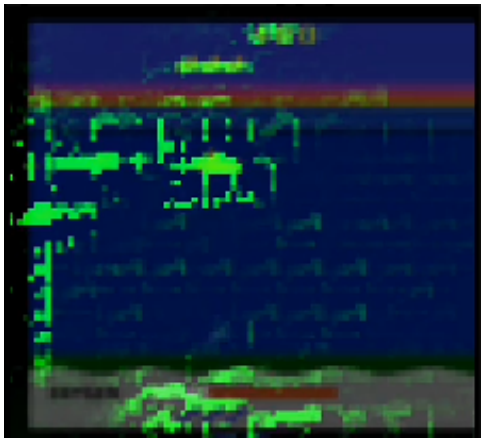

Figure 36: G2Grad-Cam 2nd convolutional layer

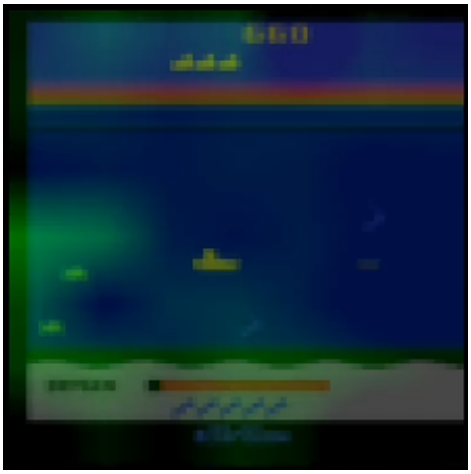

Figure 37: Grad-Cam 3rd convolutional layer. Hardly interpretable results. Partially invented highlights of the features.

Figure 38: Guided Grad-Cam 3rd convolutional layer

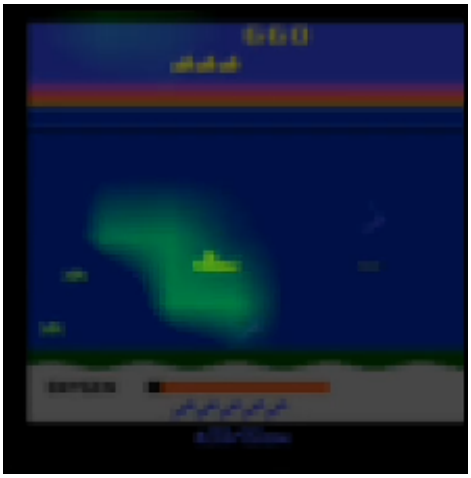

Figure 39: G1Grad-Cam 3rd convolutional layer. Better results in terms of agent position than the Grad-Cam method.

Figure 40: G2Grad-Cam 3rd convolutional layer

Figure 41: Grad-Cam 2nd convolution layer (inverted gradients). In the 3rd layer we can often observe inverted gradients with the Grad-Cam method as shown in this figure.

Figure 42: Guided backpropagation t-9. In this layer we see the visualization of the frame t-9 projected on the current frame.

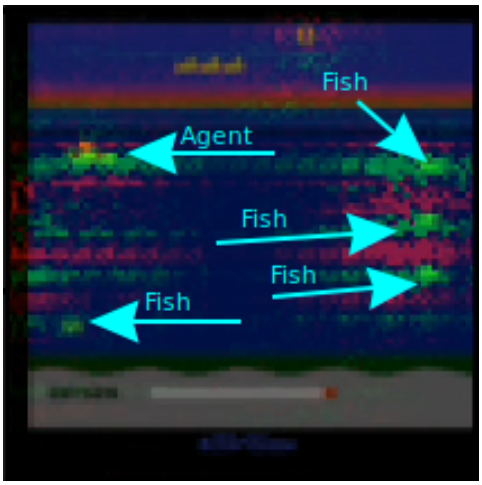

Figure 43: Gradient; episodes: 325. First gradients form around the features. Although the agent still plays randomly and does not know what these features mean, he begins to understand that they have an influence on the decisions of the agent. There is little difference between the gradient method and Guided backpropagation.

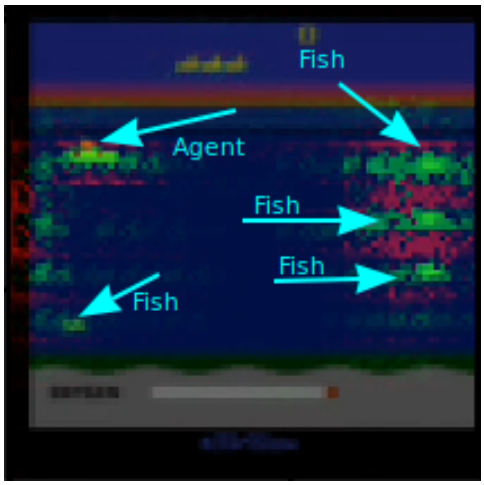

Figure 44: Guided backpropagation; episodes: 325. First gradients form around the features. Although the agent still plays randomly and does not know what these features mean, he begins to understand that they have an influence on the decisions of the agent. There is little difference between the gradient method and Guided backpropagation.

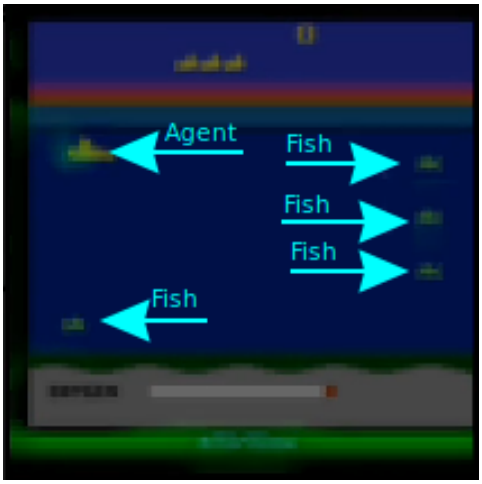

Figure 45: Grad-Cam; episodes: 325

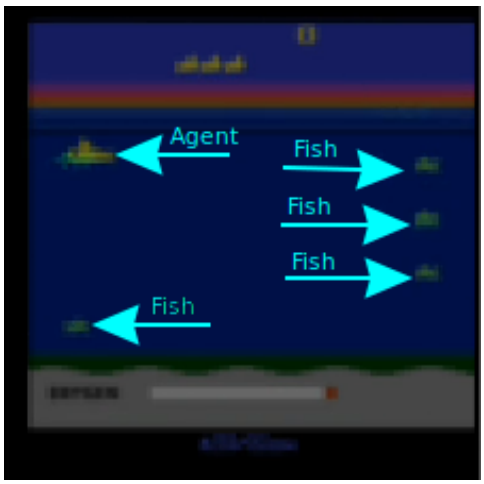

Figure 46: Guided Grad-Cam; episodes: 325. The Grad-Cam method gives clearly better and more interpretable results than the gradient methods.

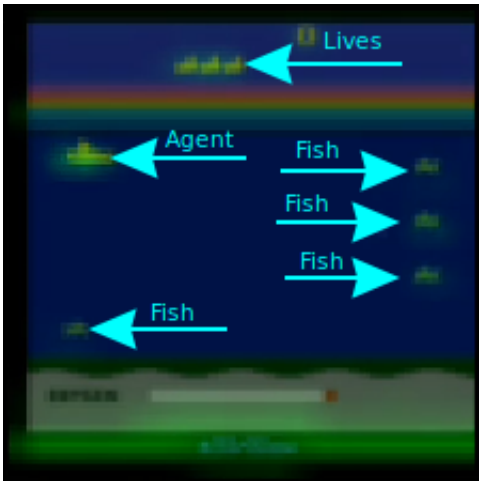

Figure 47: G1Grad-Cam; episodes: 325. The G1Grad-Cam method gives the best results we can on the fish and the agent and also on the number of lives of the agent see some highlights.

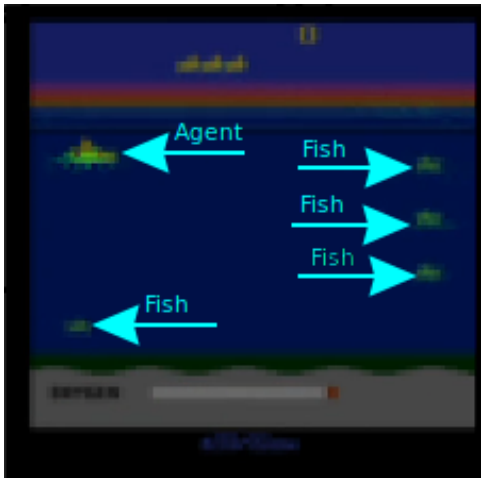

Figure 48: G2Grad-Cam; episodes: 325

B.4 A3C: Breakout-v0

Figure 49: Actor: Gradient. Gredients so small that you can't see them without a strong multiplication factor.

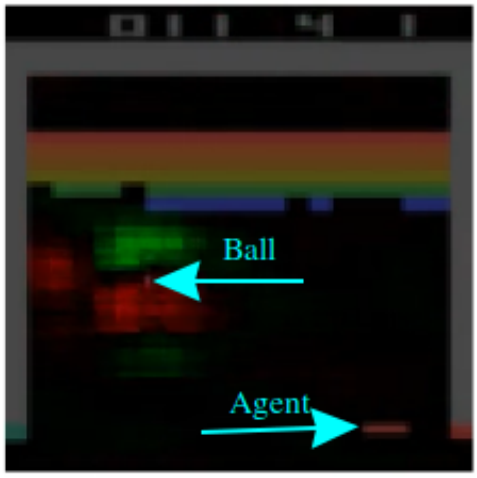

Figure 50: Critc: Gradient. You can see some arbitrary gradients around the ball.

Figure 51: Actor: Guided backpropagation. Gredients so small that you can't see them without a strong multiplication factor.

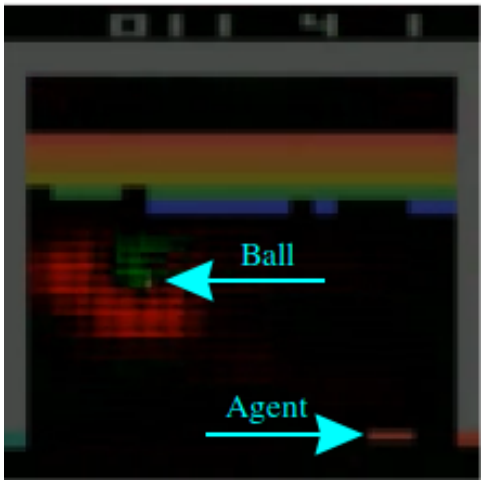

Figure 52: Critc: Guided backpropagation. Positive gradients are formed around the ball followed by negative gradients.

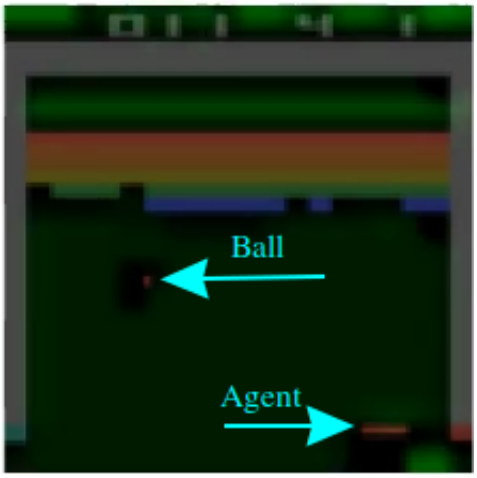

Figure 53: Actor: Grad-Cam. Inverted gradients. Around the ball and the agent you can see that the neural net recognizes the agent and the ball (invented). This is the only visualization method on the actor side that recognizes both the ball and the agent.

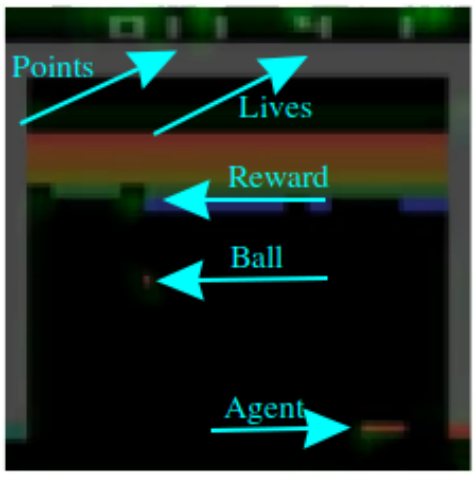

Figure 54: Critic: Grad-Cam. The ball and the agent are well highlighted by this visualization technique.

Figure 55: Actor: Guided Grad-Cam. Due to the inverted results of the Grad-Cam method no results will be shown here.

Figure 56: Critic: Guided Grad-Cam

Figure 57: Actor: G1grad-Cam. This method shows us a very strong highlighting of the ball which is also very stable. However, only the ball is highlighted and not the agent.

Figure 58: Critic: G1grad-Cam. This method shows us a very strong highlighting of the ball which is also very stable. However, only the ball is highlighted and not the agent.

Figure 59: Actor: G2grad-Cam. As we have not seen any gradients in the Guided Back-propagation method we do not see any highlighting here either.

Figure 60: Critic: G2grad-Cam. Slight gradients can be seen on the ball.

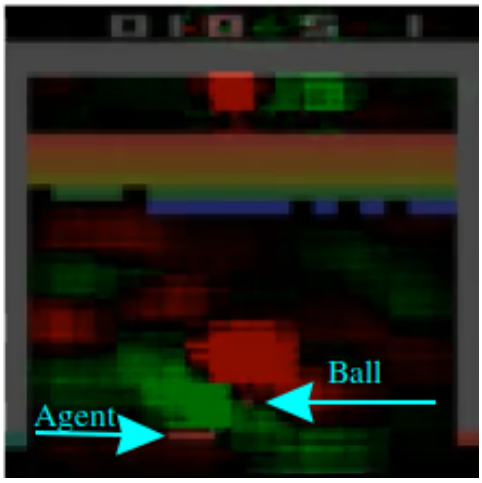

Figure 61: Actor: Gradient. Visualization of the actor gradient with a multiplication of 250.

Figure 62: Critic: Gradient.

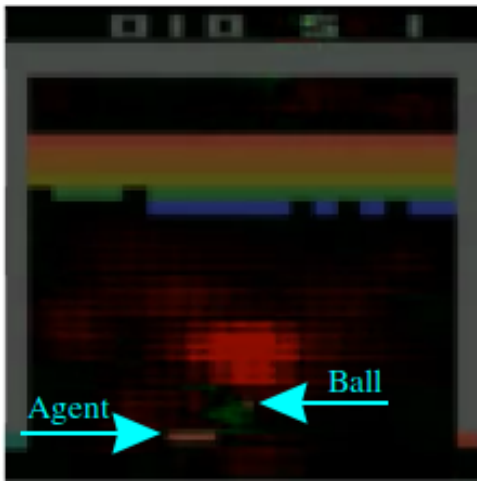

Figure 63: Actor: Guided backpropagation. Visualization of the actor gradient with a multiplication of 250.

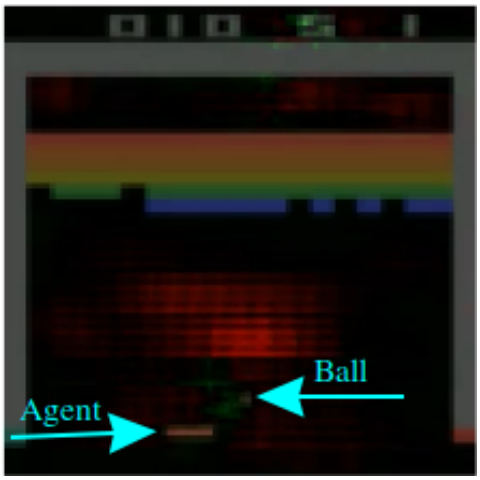

Figure 64: Critic: Guided backpropagation

B.5 A3C: Seaquest-v0

Figure 65: Actor: Gradient. Too small gradients to visualize them just like in the game Breakout-v0.

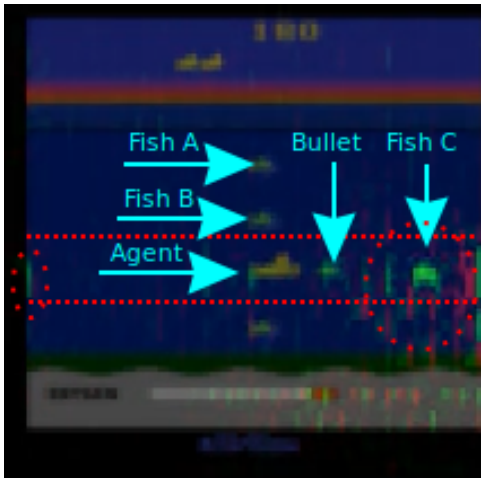

Figure 66: Critic: Gradient. Here you can see very clearly that the agent has learned to turn and shoot only left and right and never leaves the red area. You can see that the fish in this area are very strongly highlighted and outside this area there are almost no gradients to be seen. Sometimes gradients also appear on the left and right sides of the agent, where no fish can be seen (red circle).

Figure 67: Actor: Guided backpropagation. Too small gradients to visualize them just like in the game Breakout-v0.

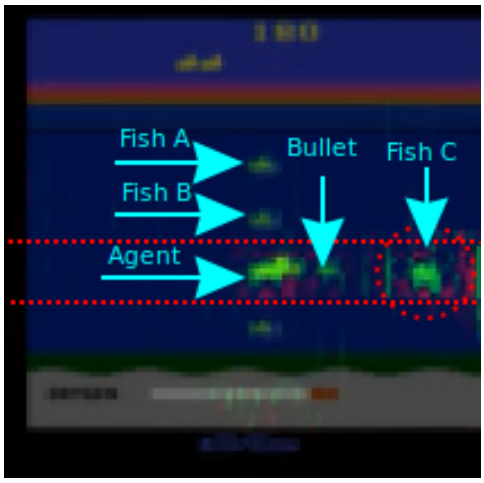

Figure 68: Critic: Guided backpropagation. Here you can see very clearly that the agent has learned to turn and shoot only left and right and never leaves the red area. You can see that the fish in this area are very strongly highlighted and also the agent itself. Outside this area there are almost no gradients to be seen.

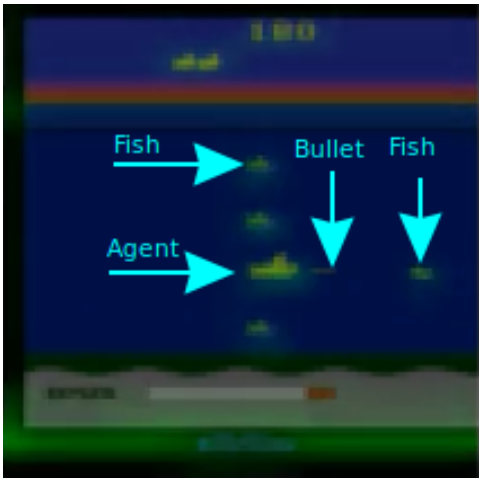

Figure 69: Actor: Grad-Cam. The fish and the agent itself are clearly highlighted. This method gives better results for the actor as well as for the Speil Breakout-v0 than the gradient methods, where the gradients are too small to visualize them without a multiplication factor.

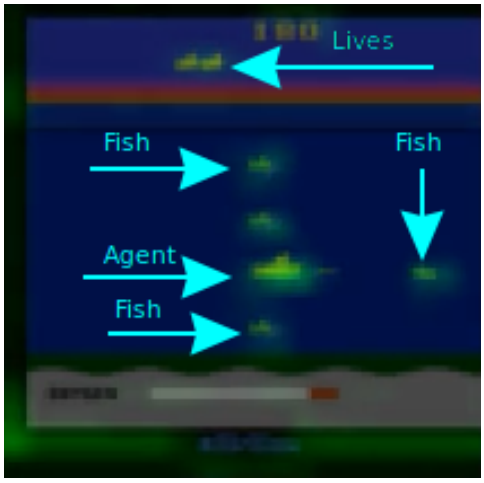

Figure 70: Critic: Grad-Cam. The fish and the agent itself are clearly highlighted. The fish and the agent itself are clearly highlighted.

Figure 71: Actor: G1Grad-Cam. Close to the important features some gradients are visualized.

Figure 72: Critic: G1Grad-Cam. Close to the important features some gradients are visualized.

B.6 A3C with LSTM: Seaquest-v0

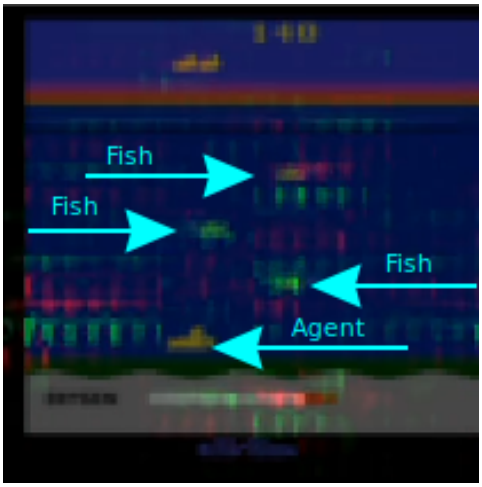

Figure 73: Actor: Gradient

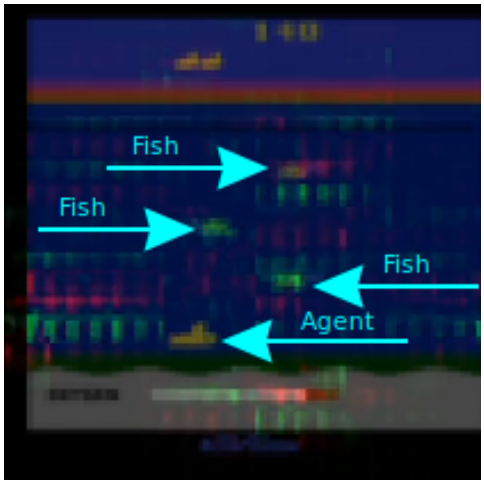

Figure 74: Critic: Gradient

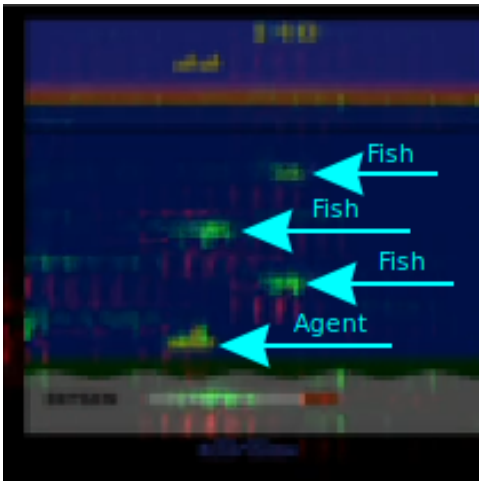

Figure 75: Actor: guided backpropagation

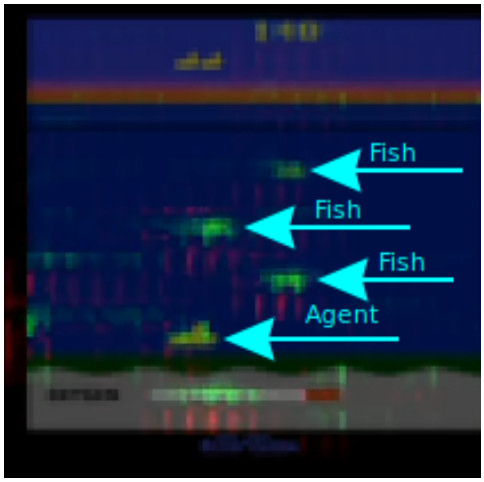

Figure 76: Critic: guided backpropagation

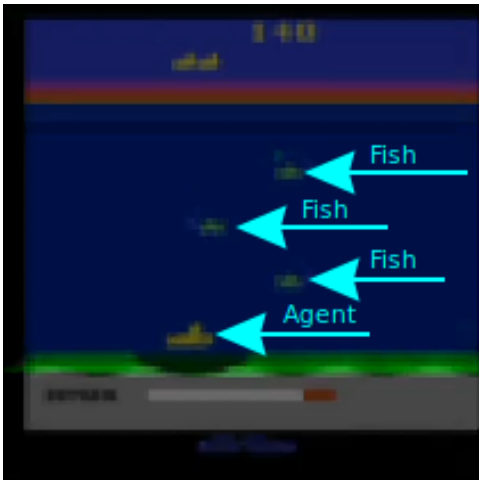

Figure 77: Actor: Grad-Cam

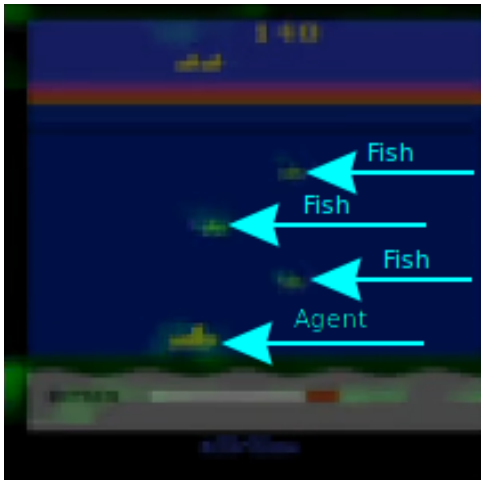

Figure 78: Critic: Grad-Cam