Using human gaze data for training a DQN to play Atari games

Eosandra Grund
egrund@uos.de

Leon Lemke
llemke@uos.de

Natalia Scharfenberg
nscharfenber@uos.de

September 2022

Abstract

Deep reinforcement learning models that learn directly from raw image inputs are plagued by sample inefficiency and slow learning speeds. Using human gaze data to supplement image inputs is a possible way to let a DQN agent achieve human-level performance on complex sequential tasks such as Atari games faster and more efficiently. Using the Atari-HEAD dataset (Zhang et al. 2019), we demonstrate that attention maps of human players and saliency maps of a DQN trained on Asterix show a degree of similarity that is higher than chance. We show that supplementing the training of a DQN agent with human attention data leads to a moderate performance improvement over our baseline. Lastly, we experiment with foveating input images to mimic human vision, and show how peripheral blurring can be integrated into the training of a DQN.

1 Introduction and related approaches

Deep reinforcement learning (DRL) has recently achieved considerable success at complex tasks that require sequential decision making such as learning to play classic video games like Atari and Doom from only pixel and game score inputs (Mnih et al. 2015; Kempka et al. 2016). Mnih et al. (2015) show that a deep Q-network (DQN) trained on an Atari 2600 game can achieve human-level (or better) performance on this game after seeing 50 million game frames. This amount of training data corresponds to a total of around 38 days of game experience. Finding ways to make DRL more sample-efficient, and cutting down training time is thus an actively pursued research

area. An approach that has been put forward with the aim of cutting down training time, and increasing sample efficiency of DRL algorithms, is supplementing training data with data of human participants who perform the same task (e.g. Silver et al. 2016; Hester et al. 2018; De la Cruz et al. 2019).

Different ways of incorporating human data in the training of deep RL agents have been proposed. Silver et al. (2016) first train a supervised learning (SL) policy network to predict actions taken by experts using 30 million expert actions. Next, a reinforcement learning policy network improves the SL policy network through self-play. In a similar vein, Deep Q-learning from Demonstrations (Hester et al. 2018) leverages human demonstration data by preceding training with a pre-training phase, during which the agent learns exclusively from demonstration data using a combination of temporal difference and supervised losses. In a different approach, De la Cruz et al. (2019) use human demonstration data to pre-train a neural network to learn state features and then transfer learned features to a DRL agent.

Human demonstration data used by the studies discussed above are limited to the information about actions taken by human (expert) players, but tracking participants' eye movements while they play a game provides another rich source of information about how humans accomplish visual tasks. Zhang et al. (2018) draw attention to the fact that human visual processing is different from how deep neural networks process input insofar as the human visual system processes visual input sequentially; in primates, this involves rapid eye movements (saccades) that bring the fovea on one particular part of the visual scene at a time (Itti and Koch 2000). This serial processing is caused by limited processing resources, as the amount of visual information we receive from the environment vastly exceeds what the primate brain is able to process. Humans have high-resolution foveal vision in the central 1–2 degrees of visual angle of the receptive field, and low-resolution peripheral vision. When we examine a complex scene, we shift our processing focus from one location to another, and we attend to only a limited set of features in the visual input. Zhang et al. (2019) argue that the ability to attend to a reduced number of features in a human-like fashion could be beneficial for RL agents that are developed to train on raw image inputs. When discussing the possibility of training RL agents on human eyetracking data, it is important to note that eyetracking data only give us insight into one type of attention—namely, overt attention. Overt attention is observable externally, i.e., we can see when a saccade is directed towards a particular object in the visual scene. By contrast, when covert shifts of attention happen, we do not gain direct access to them through eyetracking data (Zhaoping 2014, p. 62). Still, when eyetracking data are processed in order to create continuous gaze distribution heatmaps, these are referred to as "attention maps."

Das et al. (2017) conducted a large-scale study of features of visual input that humans vs. neural networks attend to, and concluded that humans and convolutional nets do not attend to the same regions of the input images. However, Guo et al. (2021) show that saliency maps of Proximal Policy Optimization (PPO) agents trained to play Atari games become more and more similar to human attention maps the longer

an agent is trained, and that there is a strong positive correlation between game score and similarity of the agent's saliency maps to human attention maps. Therefore, we may expect that incorporating information about human attention in the training of DRL agents, or training DRL agents to direct their attention in a human-like manner, will boost their performance. Das et al. (2017) have already shown that supervising for attention can boost performance of Visual Question Answering (VQA) models. Their VQA models are trained by explicit supervision at the attention layer via an $\ell 2$ loss, using human attention maps as the ground truth (Das et al. 2017). In the realm of deep reinforcement learning, Zhang et al. (2018) train a gaze prediction network on eyetracking data collected while humans played Atari games. The predicted gaze probability distribution is then used to mask image inputs for an imitation learning network. This network learns to imitate human actions and is trained on a combination of game frames and game frames masked with attention.

In a novel approach, Saran et al. (2020) introduce a coverage-based gaze loss function (CGL)—an auxilary loss term that penalizes the agent if it fails to attend to features that humans attend to, but does not incur penalties for attending to features humans do not attend to. Similarly to other studies, Saran et al. (2020) run experiments with Atari agents, and report an increase in performance without increasing model complexity.

Like the studies above, our project is motivated by an interest in human attention, and the possibilities of incorporating human attention-like mechanisms in the training of DQN agents. Our project is structured into three parts. The first part is dedicated to training a baseline DQN agent, and comparing its saliency maps to human attention maps in order to assess their similarity. Our first hypothesis is that any given saliency map of the trained DQN will have more similarity to the human attention map for the same game frame than for a randomly selected frame. However, based on findings reported by Guo et al. (2021), we still expect to find major differences between the two as we train our DQN on Asterix—an Atari game where the DQN algorithm does not achieve human-level performance (cf. Mnih et al. 2015). Choosing Asterix allows us to augment the baseline DQN by training a gaze prediction network and then incorporating attention data into training. Our second hypothesis is thus that the DQN that is trained on game frames enhanced with human attention data will learn quicker or achieve a higher game score than our baseline DQN.

Furthermore, when using images as input for deep reinforcement learning, object detection and recognition are part of the learning process of the algorithm. Li et al. (2018) show that using object characteristics as additional inputs for a DQN can increase the performance up to 20% when playing games with a lot of different objects. Relatedly, Pramod et al. (2022) foveate images (apply peripheral blurring around an object), let a state-of-the-art deep neural network recognize the object, and find that human peripheral blur is optimal for object recognition. Therefore for our third part, we augment a DQN by foveating its input images. To make the learning process more human-like, the agent sees only a single region of the image in high resolution at a given time, and we train a neural network to select the frame region where the

agent should direct its attention. Our goal is to observe how the agent behaves, and determine whether it looks at the same image regions as humans do.

The remainder of this report is organised as follows. Section 2 introduces our baseline DQN implementation that was trained to play Asterix using OpenAI Gym (Brockman et al. 2016). Section 3 is dedicated to the comparison of game frame features that are most salient for human players vs. the trained baseline DQN. Section 4 introduces a gaze prediction convolutional-deconvolutional network that we train to predict where a human player would direct their attention, and use this information to train a two-channel DQN that receives game frames masked with attention. Section 5 presents an approach that peripherally blurs game frame images to mimic the way human visual system samples the visual field. Lastly, we discuss our findings in Section 6.

The code for our models, as well as the weights of the trained models, are made available at https://github.com/egrund/DQN_vs_human_gaze. A dataset of saliency maps that was created by giving game frames from the Atari-HEAD dataset (Zhang et al. 2019) to our trained baseline DQN is available at https://osf.io/eyskv/?view_only=9dab6ccb848d471a8f0e46dfbf8ee195.

2 Baseline DQN model

All of our gaze-augmented DQN models are compared against the baseline model introduced in this section unless explicitly mentioned otherwise.

2.1 Methods

Our baseline model features the CNN architecture proposed by Mnih et al. (2015) and van Hasselt et al. (2015). The model was implemented in TensorFlow (Abadi et al. 2015) and TensorBoard (Varadarajan et al. 2022). The model was trained using a dueling network (Wang et al. 2015), prioritized experience replay (Schaul et al. 2015), and the Adam optimizer (Kingma and Ba 2014) with a learning rate of 0.00025. The buffer contained at most 200,000 samples. During training, on each iteration 3,000 new samples were added to the buffer, and the agent trained on 25,600 samples each iteration. The initial value of the exploration parameter ϵ was set to 1, and it was decayed linearly to 0.1 over 300 iterations. Each iteration one Polyak averaging step was performed with the value of α set to 0.0102.

Our baseline model was trained on Asterix using the stochastic version of the Atari 2600 environment from OpenAI Gym (Brockman et al. 2016). The game frame images were converted to grayscale, and resized to 84×84 pixels. We furthermore skipped 4 frames after each step and stacked 4 frames (non-skipped), so that the input to the network had the shape of $84 \times 84 \times 4$. For training, all rewards were divided by 10. Results are calculated using standard non-normalized rewards.

2.2 Results

The agent was trained for approximately 1,700 iterations. The average return (over 50 runs, $\gamma = 1$) for the agent after the full 1,700 iterations is 975. In contrast to this, the agent has an average return of 1309, with its peak being 2200, after roughly half the iterations. Thus, we opted to train the agent for the shorter amount of iterations.

Figure 1: Left: Training loss of the baseline DQN. Middle: Sum of rewards from all new samples divided by 10. Right: sample correction value, which can be used as a measure of how long the agent was able to play before the game terminated (the lower the value, the longer the game). Smoothed data from a single run.

3 Comparing attention maps of human players and a trained DQN

3.1 Methods

3.1.1 Dataset

The Atari-HEAD (Atari Human Eye-Tracking And Demonstration) dataset (Zhang et al. 2019) contains 117 hours of gameplay data recorded while four amateur participants played twenty Atari games. The size of the game screen was 64.6 × 40.0 cm (1280 × 840 pixels), and it was positioned at the distance of 78.7 cm from the players (cf. Figure 2). Participants played each game for up to two hours or until the game terminated. Eye-tracking data were collected using an EyeLink 1000 eye tracker at 1000 Hz. For every game frame, in addition to the gaze positions of the participants, their in-game actions (keystrokes), decision times, and the immediate reward returned by the game environment were recorded (Zhang et al. 2019).

A novel aspect of the Atari-HEAD dataset is that participants played each game in a "semi frame-by-frame" mode, i.e. the game paused at every frame until the player took an action. Zhang et al. (2019) argue that the frame-by-frame game mode helps resolve the state-action mismatch that happens when the game is running continuously at

60 Hz (the default ALE setting). Since human visual processing takes about 250–300 milliseconds before a keystroke is executed, an action a_t may have been intended for the state $s_{t-\Delta t}$ 250–300 ms earlier. Frame-by-frame game mode alleviates this problem; furthermore, it reduces participants' fatigue, and allows them to attend to more regions of every game frame than is possible in the continuous game mode.

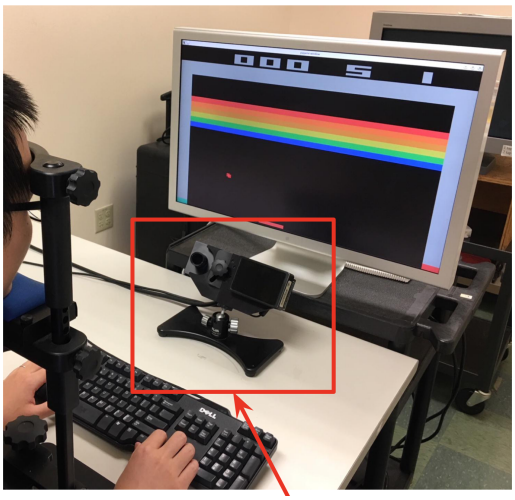

Figure 2: Collecting human data for the Atari-HEAD dataset (Zhang et al. 2019). Image from Ruohan Zhang's website.

3.1.2 Human gaze heatmaps

For each of the twenty Atari games, all of the game frames that participants saw during gameplay are included in the Atari-HEAD dataset as `.png` images, and for every game frame, participants' gaze positions are provided as a list of x, y coordinates.

To read the eyetracking data into Python, we used the method `read_gaze` from the `data_reader` module which was made publicly available in a GitHub repository[1] accompanying the Atari-HEAD dataset (Zhang et al. 2019). In order to create discrete fixation maps from these data, we prepared 2D arrays of the same dimensions as the game frame images. In these arrays, every element's value was set to 1 if its coordinates corresponded to participant's gaze position, and to 0 everywhere else. In order to obtain a continuous gaze probability distribution, the array was then smoothed with a Gaussian filter with standard deviation $\sigma = 7$ (this value corresponds to one degree of visual angle, cf. Le Meur and Baccino 2013; Zhang et al. 2019)[2]. An example of a resulting continuous gaze distribution is shown as a heatmap in Figure 3 along with a discrete fixation map and the game frame image.

[1] https://github.com/corgiTrax/Gaze-Data-Processor

[2] One degree of visual angle is an estimate of the size of the fovea. The standard deviation of 7 was computed to reflect the experimental setup (size of the screen and viewing distance) of the Atari-HEAD dataset.

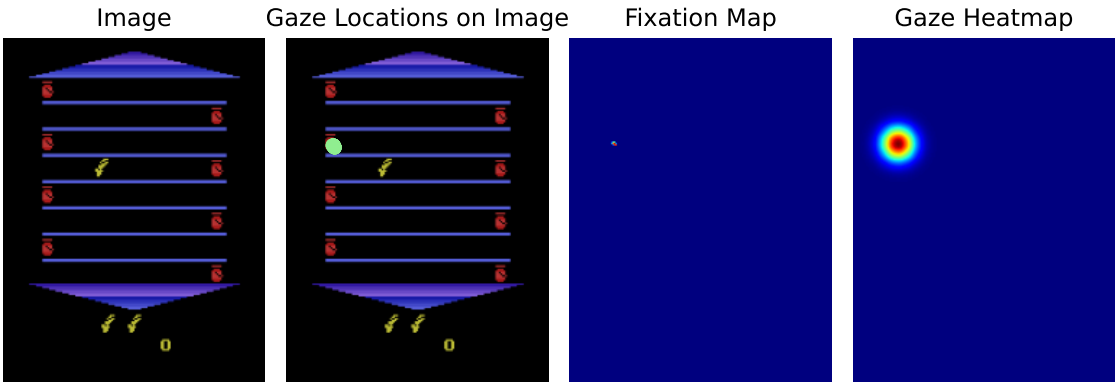

Figure 3: Outer left: Asterix game frame. Left: Scattered gaze locations of a single player in the Atari-HEAD dataset overlaid over the game frame. Right: The corresponding discrete fixation map. Outer right: The continuous gaze heatmap. Red color represents the most attended areas, dark blue the least attended.

3.1.3 Perturbation-based SARFA saliency

For determining and visualising which parts of the input image a deep reinforcement learning agent attends to, several methods have been proposed in the literature. Gradient-based methods, for example Integrated Gradients (IG, Sundararajan et al. 2017) are widely used to determine which input features are of the most importance for a neural network's performance. However, they can be difficult to interpret; moreover, changing input in the direction of the gradient may result in perturbed images that are excessively unrealistic (Greydanus et al. 2018). A newer approach—one that is specifically suited to understanding and visualising behaviour of deep reinforcement learning agents—is *Specific and Relevant Feature Attribution*, or SARFA (Puri et al. 2020). Saliency maps generated by SARFA are easily interpretable by humans (Puri et al. 2020), and are more directly comparable to human gaze heatmaps than the IG attribution masks (cf. Figure A1 in Appendix). This motivated our choice of this method for the comparison of game frame features that humans and DQN agents attend to.

SARFA is a perturbation-based method, which means that it measures how the model's output changes when the input is perturbed. To perturb Asterix game frames, we implemented the approach from Greydanus et al. (2018) as it is compatible with SARFA and is widely use for perturbation-based approaches to Atari agents.

Since the DQN was trained on grayscale images that were resized to 84×84 pixels, the following analysis was similarly carried out on resized grayscale game frames. To perturb every game frame image, we created a Gaussian mask with $\sigma = 2.8$ around every second pixel (in both vertical and horisontal directions, so that one fourth of all image pixels are perturbed during the analysis) and a perturbation image, which

determines the type of the perturbation. The value of 2.8 is in the range of 2.8–3.6 which is consistent with scaling down the value of σ used for preparing human gaze heatmaps (cf. Section 3.1.2), so that the saliency maps of the DQN agent can be directly compared with the human gaze heatmaps. The smaller value of σ was chosen because it produced the most insightful results as assessed by comparing saliency maps created with different values of σ (cf. Figure A2 in Appendix).

Since perturbing every individual pixel of every game frame would have been prohibitively expensive computationally, we used only one fourth of every image's pixels, and then upsampled each resulting saliency map to the full Atari game frame resolution. Greydanus et al. (2018) suggest to carry out computations in patches, but our approach has worked similarly well, and succeeded at cutting down the computation cost.

A black image was used for perturbations because from the different perturbation options we tried (blurred, white, random, black, cf. Figure A3) it was the one that contained the most information with only the blurred image producing similar results (cf. Figure A4 in Appendix). This was expected, as game frames are dark and all the interesting parts are light, so a lighter perturbation does make areas more interesting, while a darker perturbation as blurring (stays dark) or black removes information. Then we calculated

$$\text{image} \times (1 - \text{mask}) + \text{perturbation} \times \text{mask} = \text{perturbed image}$$

for every mask to get the perturbed images. An example showcasing the individual components of the equation above is given in Figure 4.

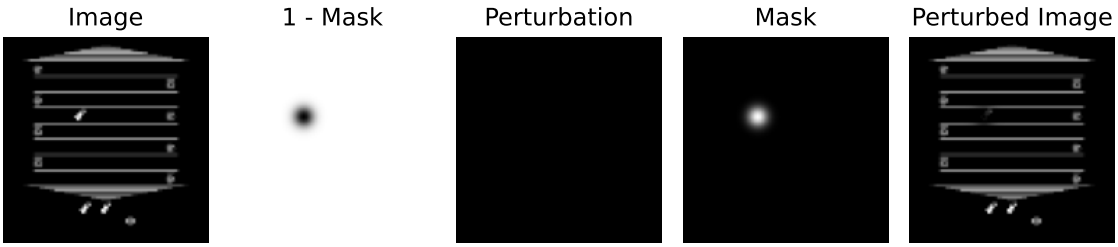

Figure 4: Example of images used during the perturbation process of a resized grayscale frame from Asterix.

Because of frame stacking, the DQN always takes in as in input a stack of four consecutive game frame images, which are all perturbed at the same spot. These images are given to the DQN to get the new Q-values, which can then be used to calculate the SARFA saliency for the perturbed pixel. For the calculation we used the function `computeSaliencyUsingSarfa` from the `sarfa_saliency` module which was made available in the GitHub repository of Puri et al. (2020)[3]. The function takes

[3] https://github.com/nikaashpuri/sarfa-saliency

both the old Q-values before the perturbation and the new Q-values after perturbation to calculate the SARFA saliency for the pixel in the center of the perturbation. The SARFA calculation consists of three parts:

1. $$K = \frac{1}{1 + D_{KL}}$$

   where D_{KL} is the Kullback–Leibler divergence between old and new Q-values

2. Δp is the difference between the softmax of the old Q-value and the new (perturbed) Q-value for the original action

3. Saliency as the harmonic mean of K and Δp:

$$S[f] = \frac{2K\Delta p}{K + \Delta p}$$

The higher the resulting value is, the more salient the area around the pixel is. Since a mask was created only for one fourth of all image pixels, each saliency map was then upsampled to the full input frame resolution. In order to save computational costs, saliency maps were stored as images, and then loaded from disk for subsequent comparisons. An example of a SARFA saliency map and the stack of input images that were used for its creation is given in Figure 5. Further details of perturbation-based methods and SARFA calculation can be obtained from the original papers by Greydanus et al. (2018) and Puri et al. (2020).

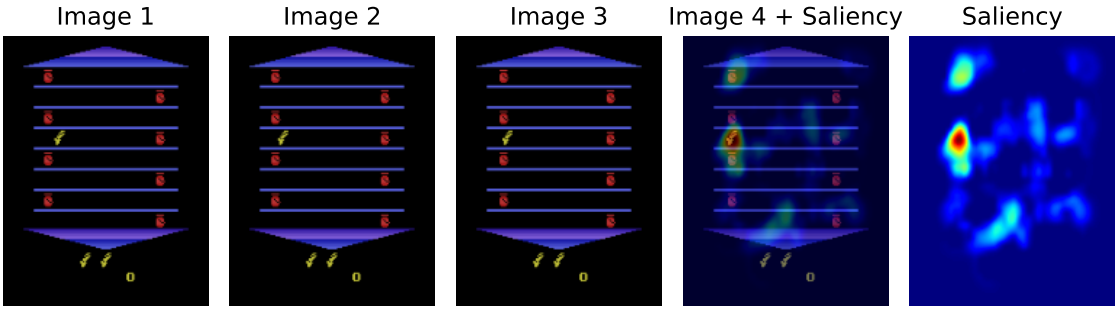

Figure 5: From left to right: Four consecutive game frame images the DQN agent receives as input. The last image (Image 4) is overlaid with the SARFA saliency map scaled up to the original frame size. Red areas correspond to higher values and higher saliency. Outer right: The computed SARFA saliency map scaled up to the original game frame size.

3.1.4 Basic binary comparison

To compare the SARFA saliency heatmap with the human attention maps, we first tried two very simple approaches. Each SARFA saliency heatmap was normalised by rescaling all values to the interval $[0, 1]$ by dividing them by the maximum value. After that, each heatmap was thresholded using the threshold of 0.2. Thresholding means that all values smaller than the chosen threshold level are replaced by 0, and all values larger than the threshold level are replaced by 1. The threshold value is usually chosen so that the top 2%, 5%, 10% or 20% salient pixels of the map are kept (cf. Le Meur and Baccino 2013). Thus, through the thresholding process, all pixels are classified as salient (fixated) or not salient (not fixated, cf. Le Meur and Baccino 2013). The process of creating a binary saliency map is shown in Figure 6. As the final step, the maps are flattened. Human gaze heatmaps can be converted to binary maps in the same manner. Alternatively, flattened fixation maps (cf. Figure 3) can be used for the comparison.

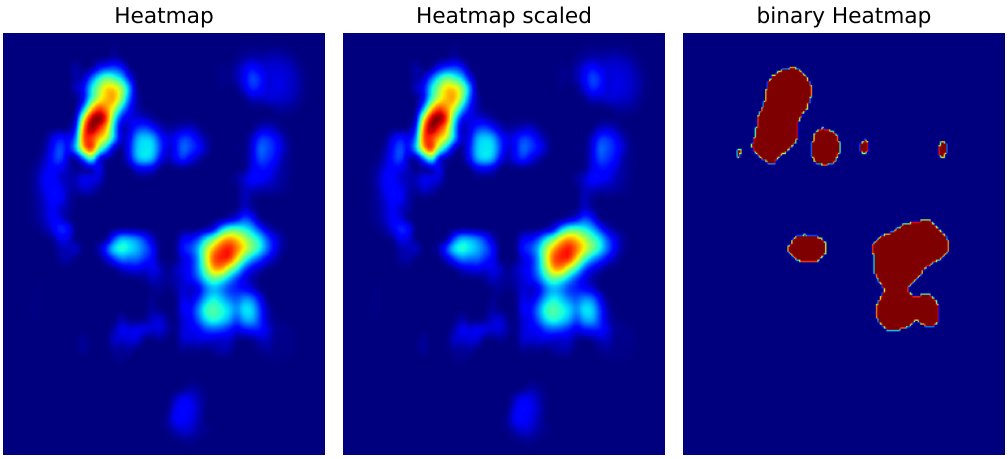

Figure 6: Left: The original heatmap. Middle: The normalised heatmap rescaled to the interval $[0, 1]$. (The normalised map does not look different from the original one because the relative values stay the same). Right: The binary map created by using a threshold of 0.2.

For comparing the SARFA saliency maps with the human gaze maps, we consider the human gaze maps to be the ground truth, and compute the True Positive Ratio (TP) and the True Positive and True Negative Ratio (TP TN):

$$\text{TP} = \frac{\text{number of true positive}}{\text{number of true positive } + \text{ number of false negative}}$$

$$\text{TP TN} = \frac{\text{number of true positive } + \text{ number of true negative}}{\text{total number of pixels}}$$

3.1.5 Area Under ROC Curve (AUC)

AUC is metric for binary classification problems. Its values are constrained between 0 and 1. We use it because it is simple to understand and compute, and used by Zhang et al. (2019); as are Information Gain (cf. Section 3.1.6) and Pearson's Correlation Coefficient (cf. Section 3.1.7). Moreover, AUC is the most widely used metric for evaluating saliency maps (Bylinskii et al. 2019). When used for saliency maps, it measures the true and false positive rates of the salient vs. not salient classification (Bylinskii et al. 2019).

To compute AUC, the same preprocessing as described in Section 3.1.4 was done: agent's saliency maps were converted to flat binary arrays, and fixation maps or gaze heatmaps from human participants were used as ground truth. For working with flattened fixation maps, we use a variant from Judd et al. (2009) called AUC-Judd (Bylinskii et al. 2019). When both are converted to binary arrays, we can calculate AUC by viewing the saliency map as a binary classifier, and the human gaze heatmap as the ground truth.

3.1.6 Information Gain (IG)

Information Gain is used frequently in diverse fields, and has recently been used to measure saliency model performance by Kümmerer et al. (2015). Bylinskii et al. (2019) showed that IG has similar properties to the Kullback-Leibler divergence. IG tests if the model performs better than a systematic bias (e.g., a center prior baseline), by calculating the information gain of the saliency over the baseline for each fixated location. If it returns a score above zero, the saliency map predicts the fixation locations better than the baseline.

Because IG is a location-based metric, we do not have to convert the gaze data into a continuous distribution (i.e., a heatmap). Instead we can directly use fixation maps. The prior baseline is created by creating a black image of the same dimensions as the fixation map, setting the value of the center pixel to 1, and applying a Gaussian filter with $\sigma = 0.7$. This is the same value as in Section 3.1.2 for the gaze heatmap creation, so the prior baseline is equivalent to the observer fixating on the center of the picture.

3.1.7 Pearson's Correlation Coefficient (CC)

Pearson's Correlation Coefficient is often used in sciences to measure the dependence of two variables. It is symmetric and restricted to the interval $[-1, 1]$ where -1 represents a perfect negative and 1 a perfect positive linear correlation. Because CC is a distribution-based metric, we have to transform the gaze data into a continuous distribution, which we do by creating a gaze heatmap. CC is a measure of the linear relationship between our gaze heatmap and the saliency map, where positive values imply their similarity. For creating a correlation heatmap we used the following

formula, which calculates the value for pixel i (Bylinskii et al. 2019):

$$V_i = \frac{P_i \times Q_i^D}{\sqrt{\sum_j (P_j^2 + (Q_j^D)^2)}}$$

With P being the saliency and Q^D the gaze heatmap. The result has high values for positive linear correlations. An example of a correlation heatmap is shown in Figure 7.

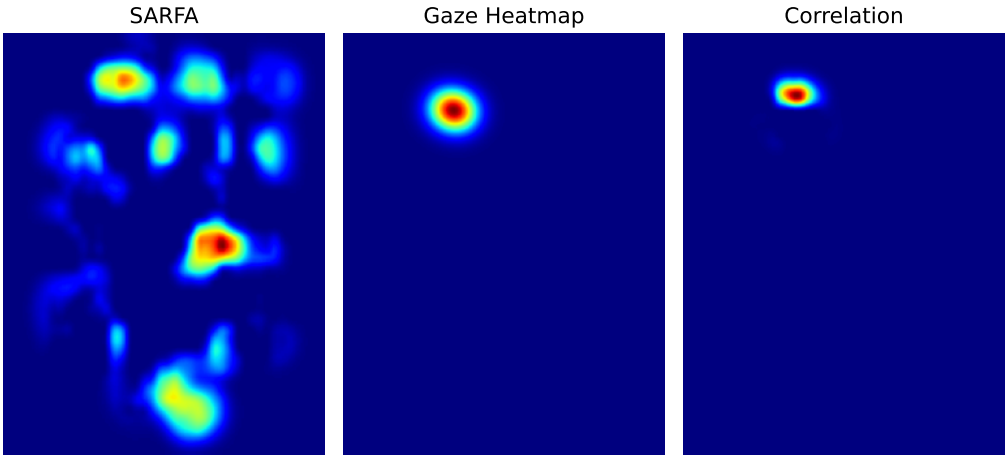

Figure 7: Left: A SARFA saliency heatmap. Middle: A gaze heatmap. Right: The resulting correlation heatmap from the left and middle heatmaps. The correlation of these heatmaps is -0.0110.

3.2 Results

Because the SARFA saliency is always computed by taking into account four consecutive frame images, we similarly used the gaze data from all four images to compare against the saliency by combining them as if they were from one image. For each method we used, we always compared the similarity value to two random values created by firstly calculating the value for a randomly assigned (RA) gaze map and saliency map, and secondly by calculating the resulting value for the matching saliency map all of whose pixel values are shuffled. Then we did a z-test to check if the first value is significantly bigger than the RA value (if the p-value is smaller than 0.05). We also compared them to a value analysis of several saliency heatmap comparison techniques by Bylinskii et al. (2019). All heatmaps used for the analysis were made sure to be floats in the interval $[0, 1]$ in order to have a valid comparison. The stochasticity was controlled for by setting a random seed before the comparison calculations. Trials *160_RZ_9166697, 167_JAW_2356024, 163_RZ_9932825, 315_RZ_216627, 171_JAW_3395791* and *243_RZ_593078* from the Atari-HEAD dataset were used, and from these trials, only frames 0 to 4000 with a step size of 4 were analysed. We did

not use all of the Asterix data from the Atari-HEAD dataset because of limited computational resources. The decision to go only up to frame 4000 was made because our model only achieved a maximum game score of 2200, and at frame 4000 humans already had a score of over 4500. Guo et al. (2021) show that similarity drops significantly when using images unseen by the agent for comparison with humans. Therefore, we might have no similarity at all, because the human players were able to reach very high scores because of the frame-by-frame game mode.

Our exact results are listed in Table 1 and visualized in Figure 8. As you can see, most of the results support that the heatmaps are significantly more similar than the randomly assigned heatmaps. But we have to keep in mind that because of the big sample size, even small and minor differences can become significant. There significance does not mean the SARFA saliency map is very similar to the gaze heatmap. Our Information Gain score of about 0 implies that our SARFA saliency maps are as similar to the gaze heatmaps as the center prior. The Area under ROC score of 0.6 is worse than the center prior (0.78) and therefore a rather small value in saliency science (Bylinskii et al. 2019). And the correlation of 0.16 is a small correlation to begin with, but also very small in the context of saliency maps (Bylinskii et al. 2019). Therefore, we should have seen higher similarity values for our saliency maps.

In Figure 8 it is visible that there is a lot of overlap between the Q1-Q3 intervals in comparison to RA and Shuffled. This shows that the heatmaps are more similar on average but most of them are not still dissimilar. Figure 9 shows an example of a highly correlated pair and a very low correlated pair, and the differences are noticeable. The highly correlated pair (CC = 0.735) has a high correlation value for saliency science (Bylinskii et al. 2019), but values in this range were also found among the random assignments (cf. Figure 8).

Another striking result is the absence of difference between the values in True Positive and True Negative for fixation maps ($p = 0.7756$), and the comparatively big p-value when comparing to heatmaps. This might be explained by the SARFA saliency maps on average having a bigger space covered (average pixel value of 0.07) in comparison to the gaze heatmaps (average pixel value of 0.02). Because both are rescaled to the interval $[0, 1]$, the bigger value does not just mean the salient areas have bigger values.

	Data	Mean	RA Mean	p-value	Shuffled Mean
TP	F	0.3984 ± 0.0100	0.2816 ± 0.0092	5.46e−64	0.1204 ± 0.0028
TP	H	0.3728 ± 0.0070	0.2774 ± 0.0065	6.08e−86	0.1208 ± 0.0011
TP TN	F	0.8767 ± 0.0010	0.8773 ± 0.0011	0.7756	0.8788 ± 0.0010
TP TN	H	0.8692 ± 0.0011	0.8660 ± 0.0010	1.49e−05	0.8603 ± 0.0010
AUC	F	0.6378 ± 0.0050	0.5797 ± 0.0045	3.20e−64	0.4999 ± 0.0013
AUC	H	0.6278 ± 0.0036	0.5795 ± 0.0032	2.10e−87	0.5001 ± 0.0002
IG	F	0.0009 ± 0.0000	0.0007 ± 0.0000	1.08e−18	0.0004 ± 0.0000
CC	H	0.1634 ± 0.0045	0.0977 ± 0.0037	6.11e−108	0.0001 ± 0.0001

Table 1: TP: True positive. TP TN: True Positive and True Negative. AUC: Area under ROC curve. IG: Information Gain. CC: Pearson's Correlation Coefficient.
Data: fixation map (F), heatmap (H). Mean: the resulting value. RA Mean: comparison with a randomly assigned saliency map from the dataset. p-value: z-test for Mean is bigger than RA Mean. Shuffled Mean: calculated for saliency maps whose pixel values have been shuffled.

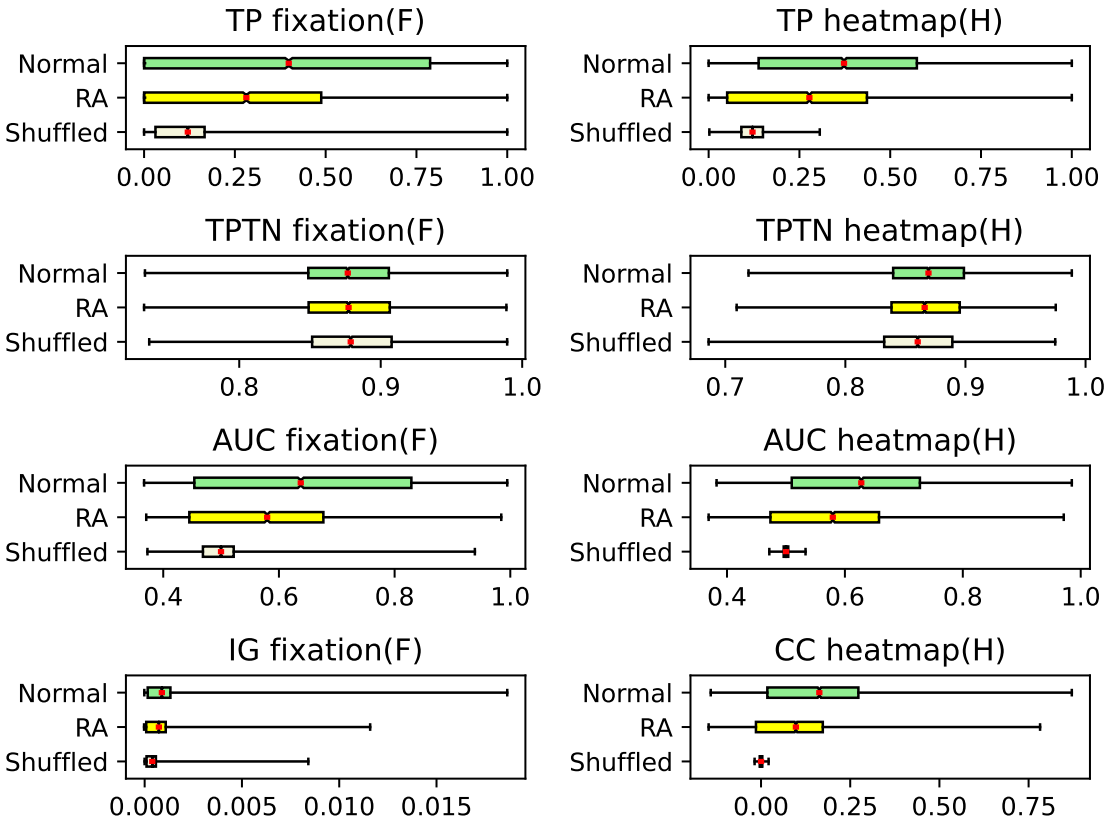

Figure 8: Visualizing the results of Table 1.
Red line: Mean with the notches showing the confidence interval. Boxes: Q1–Q3. Whiskers extend from minimum to maximum.
TP: True positive. TP TN: True Positive and True Negative. AUC: Area under ROC curve. IG: Information Gain. CC: Pearson's Correlation Coefficient.
Normal: Matching data, i.e., saliency map is for the same game frame as the human gaze heatmap or fixation map. RA: randomly assigned. Shuffled: matching but the saliency map's pixel values are shuffled.

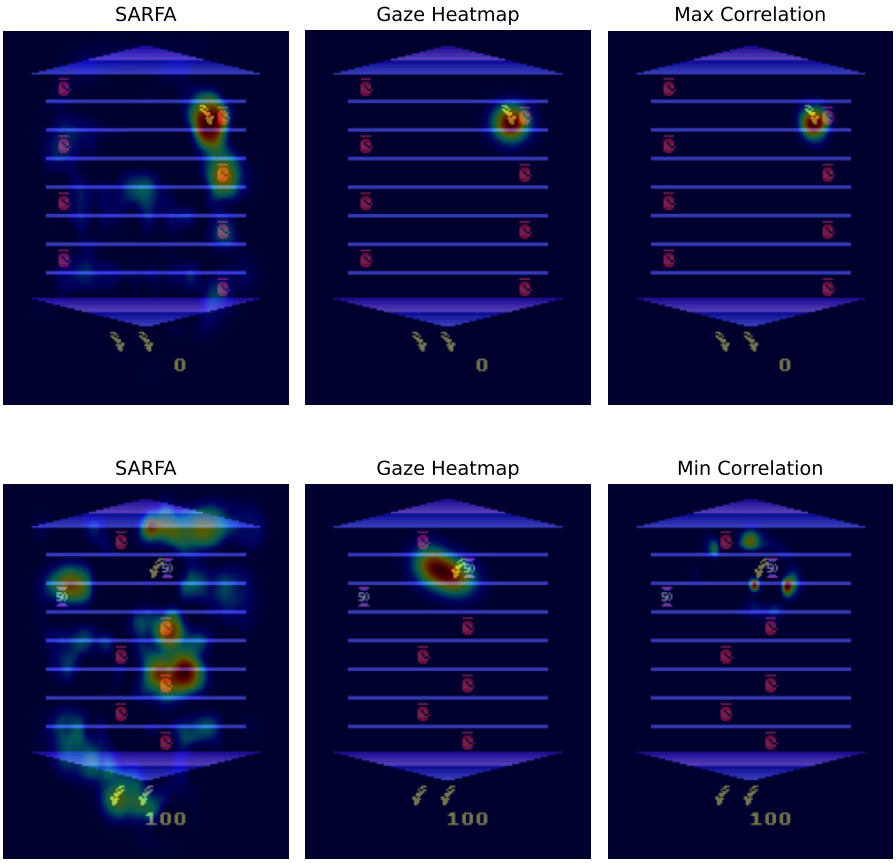

Figure 9: Showing the pairs with maximum (top, CC = 0.7354) and minimum (bottom, CC = −0.0839) correlations found in the entire saliency dataset. For each line from left to right: SARFA saliency heatmap of the trained DQN agent, matching human gaze heatmap and the correlation heatmap of the saliency and gaze heatmap.

4 Using human attention maps to train a DQN

4.1 Methods

4.1.1 Gaze prediction network

As Zhang et al. (2019) show, it is possible to train a convolution-deconvolution neural network (Deng et al. 2020; Palazzi et al. 2019) to predict human gaze positions for Atari game frames with high accuracy, provided that a large amount of human eyetracking data is available (Zhang et al. 2019 report an average AUC value of 0.971 for the Atari-HEAD data)[4].

Our implementation features the same architecture as Zhang et al. (2019): the gaze prediction network takes in as input a stack of four consecutive grayscale game frame images, and passes the inputs through three convolutional layers which use ReLU as the activation function. Convolutional layers are followed by three deconvolutional layers (also with ReLU activations), and a softmax layer. The final output of the network is a gaze heatmap that represents the predicted probability distribution of gaze locations for the final frame in the frame stack. The network architecture is shown in Figure 10.

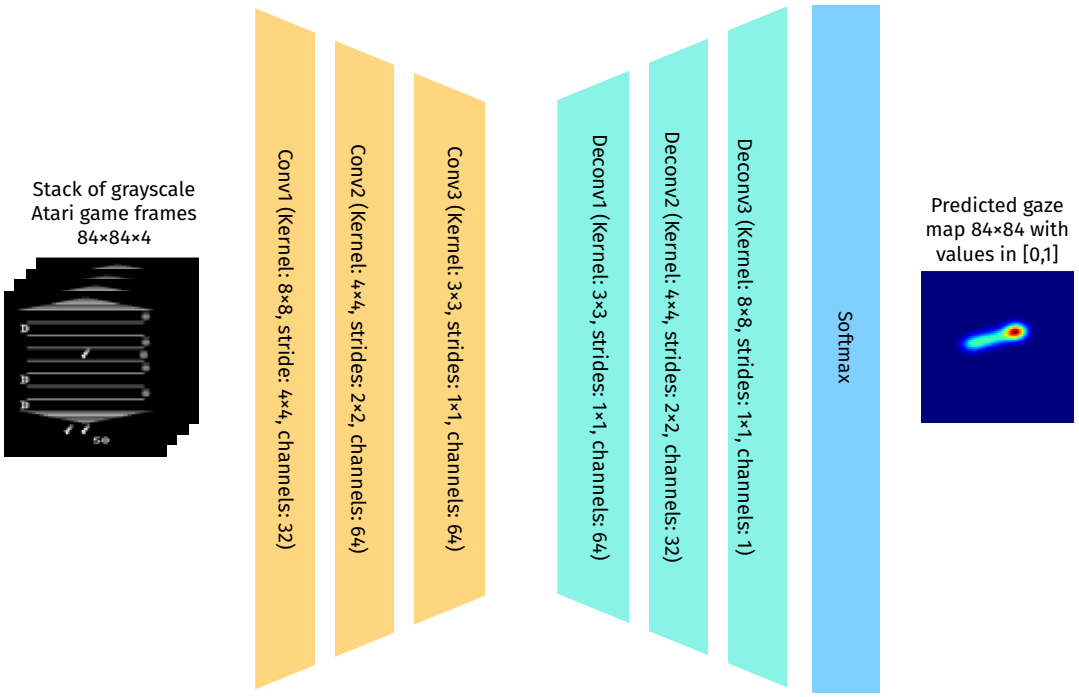

Figure 10: Gaze prediction network featuring the architecture introduced in Zhang et al. (2019).

[4] A detailed description of how AUC is calculated can be found in Section 3.1.5.

The gaze prediction network was trained on all Asterix game frames included in the Atari-HEAD dataset (Zhang et al. 2019), which amounts to 194,000 four-frame stacks in the training set, and 48,564 four-frame stacks in the validation set, and the corresponding gaze locations. The process of creating gaze heatmaps from the `.txt` files containing gaze coordinates was identical to that described in Section 3.1.2. In order to preserve gameplay continuity, frame stacking was done prior to shuffling the dataset. The gaze prediction network was trained for 250 epochs using the Kullback–Leibler divergence as the loss function, Adam optimiser (Kingma and Ba 2014) with a learning rate of 0.001, and a batch size of 256.

Gaze prediction network's performance was worse than the results reported by Zhang et al. (2019), as it only achieved an average AUC value of 0.6. It is unclear why Zhang et al.'s results were not reproducible.

Figure 11: Left: Gaze map predicted by the gaze prediction network overlaid on top of the corresponding game frame. Right: Masked game frame, i.e., the element-wise product of the predicted gaze map and the game frame. The masked frame is used as input to the "gaze" channel of the two-channel DQN (cf. Section 4.1.2)

4.1.2 Two-channel DQN

In order to augment the training of our baseline DQN agent with human gaze data, the baseline model was altered by adding a second channel. The architecture of this two-channel network follows the one introduced by Zhang et al. (2019). The input to the game frame channel is a stack of four grayscale game frame images. The input to the "gaze" channel is a masked image which is the element-wise product of the final image of the four-frame stack, and the output of the gaze prediction network described in Section 4.1.1. Each channel consists of three convolutional layers followed by max pooling and in the case of the final convolutional layer, global max pooling, followed by a fully connected layer. The output of the two channels is averaged before the state value and advantages are calculated (cf. Wang et al. 2015).

The hyperparameters for training the two-channel DQN were kept unchanged from the baseline described in Section 2. It was trained for 1,200 iterations (i.e., for a smaller number of iterations than the baseline).

4.2 Results

As seen in Figure 12, the gaze-augmented DQN reaches a higher reward value than the baseline. It also reaches lower values for the sample correction parameter, which means that the agent is able to play for longer before the game terminates. Overall, it appears that giving the agent access to game frames masked with human attention predictions improves its performance, even though it is not a dramatic increase. Given that the gaze prediction accuracy was limited, we can speculate that better gaze predictions could improve model's performance further.

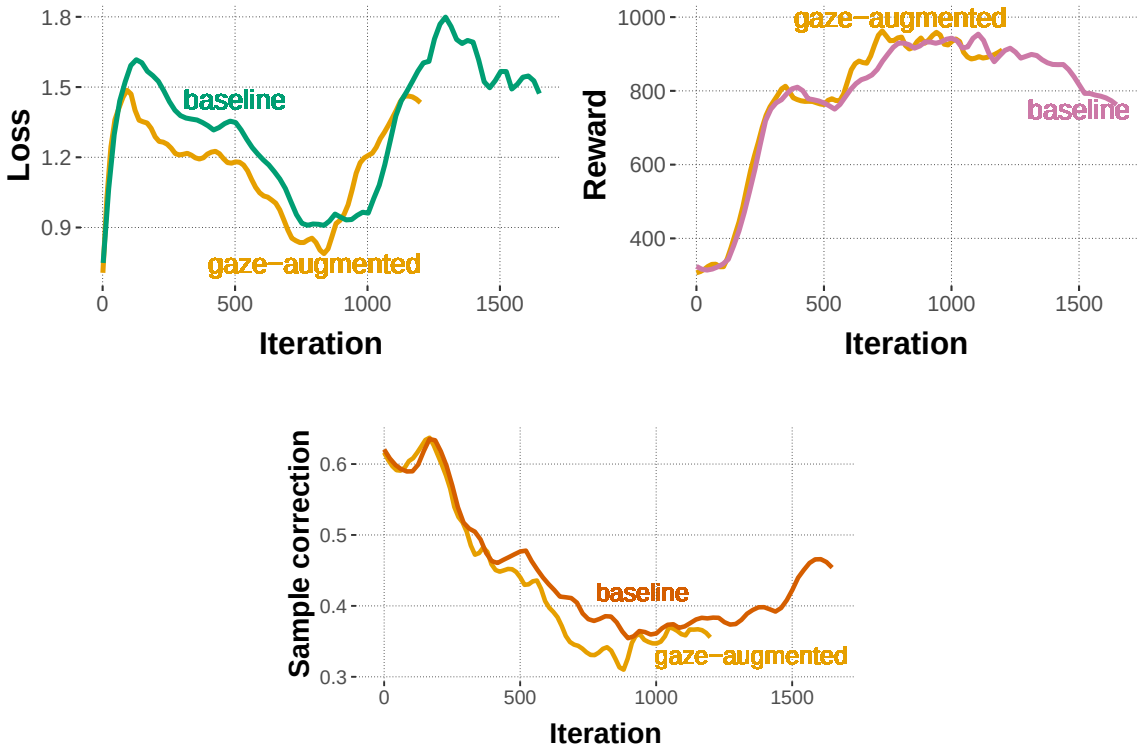

Figure 12: Training loss, reward summed over all new samples, and the sample correction value of the baseline DQN, and the two-channel gaze-augmented DQN that was trained on predictions of the gaze network in addition to stacks of four greyscale game frames. Smoothed results from two runs. (Since training these models from scratch was computationally expensive, this prevented us from running more experiments.)

5 Using foveated game frame images to train the DQN

5.1 Methods

5.1.1 Differentiable blur

In order to force our DQN to focus its attention on a specific part of the input image, we created a gaze mask that is similar to human vision, i.e. high-resolution vision at the point of focus, and blurred vision around the center. However, this new output image has to be differentiable with respect to the input image as well as the gaze positions. Let s be our image, s^{gauss} be the same image distorted with a Gaussian blur, and x and y the respective gaze positions. Then the new image is calculated as:

$$blur_{\lambda,r} : \mathbb{R}^n \times \mathbb{R}^m \times \mathbb{R} \times \mathbb{R} \to \mathbb{R} \times \mathbb{R}, (s, x, y) \mapsto [s_{i,j}^{blurred}]_{n \times m}$$

$$s_{i,j}^{blurred} = \frac{(1 - m'_{i,j}) \cdot s_{i,j} + \lambda * (m'_{i,j}) \cdot s_{i,j}^{gauss}}{(1 - m'_{i,j}) + (\lambda \cdot m'_{i,j})}$$

$$m'_{i,j} = \frac{m_{i,j} - \min_{k,l}(m_{k,l})}{\max_{k,l}(m_{k,l}) - \min_{k,l}(m_{k,l})}$$

$$m_{i,j} = \max(0, \sqrt{(x-i)^2 + (y-j)^2} - r)$$

r determines the radius of the sharp area. λ determines how much the image is blurred outside of the sharp area. n and m are the width and height of the image respectively.

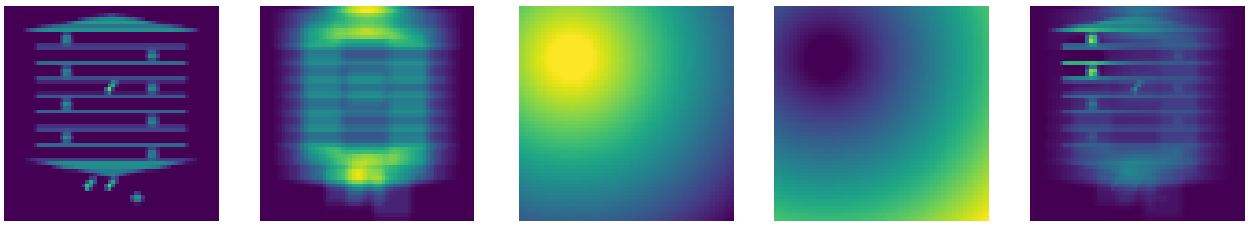

Figure 13: From left to right: A random state from the Asterix game. The same state with a Gaussian blur applied to it. $(1 - m')$. m'. Blurred state with gaze at $x = 20$ and $y = 20$

5.1.2 Sampling

Assume that we have a DQN $Q : S \times A \to \mathbb{R}$, a blurrer $B : S \to \mathbb{R}^4 \times \mathbb{R}^4$, an environment E, a starting state s_0, initial gaze positions x, y. Note that the blurrer outputs 4 x and 4 y coordinates, because four stacked images count as one state. These four positions, however, do not necessarily have to be the same.

Algorithm 1 Sampling

Initialize $Q, B, E, s_0, x, y, t = 0$
while E is not finished **do**
  $s_t^{blurred} \leftarrow blur_{\lambda,r}(s_t, x, y)$
  $a \leftarrow argmax_a(Q(s_t^{blurred}, a))$
  $s_{t+1} \leftarrow E(a)$ ▷ Sample new state from environment
  $x, y \leftarrow B(s)$
  $x, y \leftarrow$ random position with probability ϵ
  $t \leftarrow t + 1$
end while

5.1.3 Training

The overall training procedure is almost the same as for double deep Q-learning. The only difference is that after each update for the Q-network the blurrer is also updated using gradient descent. For a rough sketch see Algorithm 2.

Algorithm 2 Training

Initialize Q, Q_{target}, B, replay buffer
for i **do**
  for j **do**
    Sample $s_t^{blurred}, a, r, s_{t+1}^{blurred}$ from the replay buffer
    Perform a training step for the DQN using $s_t^{blurred}, a, r, s_{t+1}^{blurred}$
  end for
  for k **do**
    Sample $s_t^{blurred}, a, r, s_{t+1}$ from the replay buffer
    $x, y \leftarrow B(s_t^{blurred})$
    $s_{t+1}^{blurred} \leftarrow blur_{\lambda,r}(s_{t+1}, x, y)$
    $q \leftarrow Q(s_{t+1}^{blurred})$
    Minimize loss w.r.t. B: $-q^2 \cdot \frac{q}{|q|} + \text{clip}(x, 0, 84)^2 + \text{clip}(y, 0, 84)^2$
  end for
  Sample from environment and add new data to replay buffer
  Perform Polyak averaging
end for

5.2 Verifying the methods

In order to see if this method works, we trained it on a toy problem. The agent controls a small ball at the bottom of the screen on the x-axis and has to collect falling balls, and avoid falling rectangles. Results show that the agent did learn the basic game, and that the gaze point goes to where the agent needs information about the falling object. A clip of gameplay is available at https://github.com/egrund/DQN_vs_human_gaze/blob/main/toy_problem.gif

Figure 14: Four consecutive frames from a test game run (cf. Section 5.2) by a trained agent

5.3 Results

The agent was trained on roughly 6.4 million frames, and we decreased the model size in contrast to the original run. Regarding the blur, we chose $r = 12$ and $\lambda = 50$. As one can see, the average reward per episode increased. However, if one looks at a run https://github.com/egrund/DQN_vs_human_gaze/blob/main/asterix_with_blurr/normal_run.gif, one can see two things. Firstly, the gaze positions stay the same for every frame; secondly, the agent still uses information in the blurred area in a way that would be hard for humans. It can accurately track the positions of certain objects, and also knows what that object (collectable or enemy) is. Another interesting thing is that all the gaze positions are in the top left corner of the game frames. If one looks at the average x and y gaze positions, one notices that x and y are close to 0. So in this first few iterations, although the value of the exploration parameter ϵ is still high, it might be that the agent is trained on more images which have a gaze positions in the top left corner, further increasing this effect.

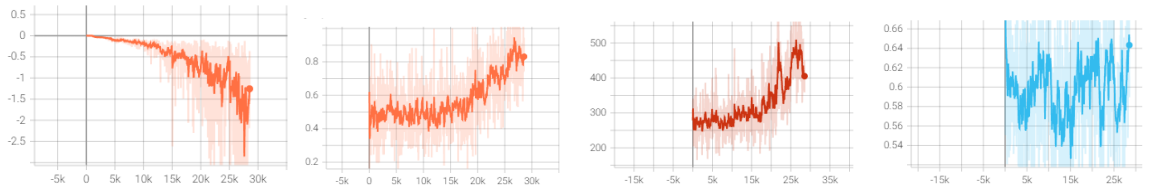

Figure 15: From left to right: Training loss of the blurrer. Training loss of the DQN. Average reward of one episode. Sample correction value, which can be used as a measure of how long the agent was able to play before the game terminated (the lower the value, the longer the game).

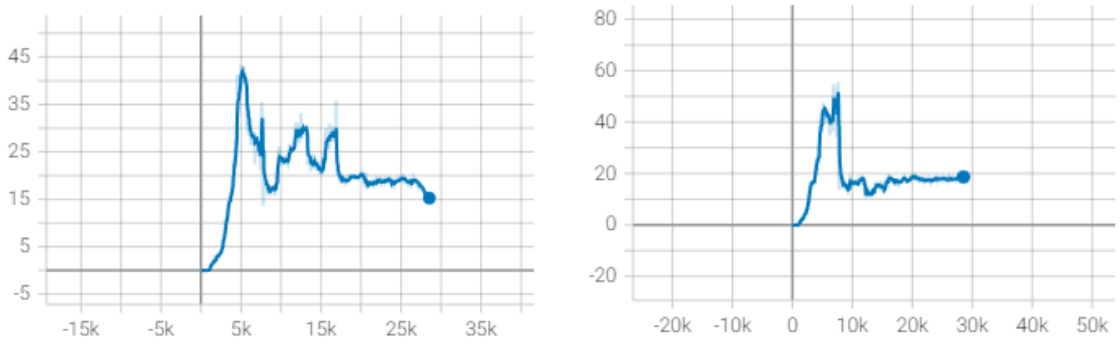

Figure 16: **Left:** average x coordinate, **Right:** average y coordinate

5.4 Improvements

One problem with applying the Gaussian blur to the game frames is that most parts of the images are black. This means that information far away from the gaze point does not get destroyed, because there are no other pixels around the game objects. To counter this, we add the same noise to each game frame. This results in a blur where information far away from the gaze positions is completely destroyed. To counter the effect of the first gaze positions being in the top left corner, we have to ensure that the output values of the blurrer are not distributed around 0. This is done via initializing all weights of the final layer with a constant (0.0018125), and the bias with 0. However, this only solved the problem of the DQN acting in the blurred area. In the improved version, the agent prefers to be within the sharp area.[5] However, the gaze positions still remain the same in each frame. This could be due to the following. The chance that during sampling the gaze positions will be on the "point of interest" is lower than the gaze positions being anywhere else. Thus, the model learns to work with blurred data. An increase of Q-values now results in the images being "stable", meaning that the gaze positions are in the same place in every frame. Furthermore, the algorithm probably needs a lot more training and a much slower exploration parameter decay than in the normal DQN training. So it could be that we simply did not train enough or that the agent did not have enough exploration.

[5] A video clip is available at https://github.com/egrund/DQN_vs_human_gaze/blob/main/asterix_with_blurr/improved_run.gif

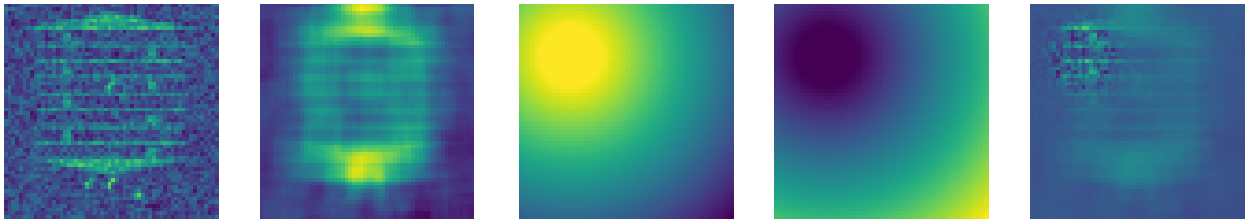

Figure 17: From left to right: A random state from the Asterix game with noise applied to it. The same state with a Gaussian blur applied to it. $(1 - m')$. m'. Blurred state with gaze at $x = 20$ and $y = 20$

6 Discussion and conclusion

While deep reinforcement learning agents have been shown to achieve human-level performance on complex tasks such as playing Atari games (Mnih et al. 2015), they require large amounts of training data to achieve these results. The goal of our project was to explore how human gaze data can be used to make DQN training faster and more sample-efficient. Atari-HEAD, a recently published dataset that comprises 117 hours of human gameplay data (Zhang et al. 2019)) makes it possible to pursue this goal.

Firstly, we compare gaze heatmaps of human players who play Asterix to the saliency maps obtained from a DQN trained to play the same game. The human eyetracking data are taken from the Atari-HEAD dataset, and preprocessed to create discrete fixation maps and continuous gaze probability distributions for every game frame. The saliency maps of the trained DQN agent are created by using a combination of a perturbation-based method (Greydanus et al. 2018) and SARFA (Puri et al. 2020). We find that the saliency maps are significantly more similar to the human gaze heatmaps for the corresponding game frames than for randomly selected frames. However, the similarity values are still rather low (comparing AUC, TP TN, IG and Pearson's correlation coefficient). One of the factors contributing to moderate average similarity may be the fact that trained DQN agent usually attend to more regions of input images than human players (cf. Guo et al. 2021). Only the True Positive metric is not affected by this, as it does not penalize the agent for attending to more areas than human players. With these findings we should bear in mind that saliency maps cannot be used as an explanation for the algorithm's behaviour directly, because they are more exploratory than explanatory (Atrey et al. 2019). But in our case this finding fits well with the limited success of our baseline DQN, as our baseline model does not get into the game score range where human players are, and there is a strong positive correlation between saliency map similarities and game scores (Guo et al. 2021).

Next, we used the Atari-HEAD data (Zhang et al. 2019) to train a convolutional-deconvolutional neural network to predict which regions of each game frame a human

player would attend to. The output of the trained gaze prediction network was used to mask game frames with attention information. A two-channel DQN was trained on stacks of four game frames and frames masked with attention. The two-channel model showed a moderate improvement in performance, achieving a higher maximum reward value and a smaller sample correction value than our baseline. These results are consistent with what Zhang et al. (2019) report. However, this approach would be difficult to apply to real-life problems where a similar amount of eyetracking data as in the Atari-HEAD dataset is not readily available, or is prohibitively expensive to obtain. Thus, it is desirable to have a human attention-like mechanism as part of artificial agents, as attention is very likely to lead to performance improvements, but there is currently no simple way of achieving this.

In the final part of our project, we have experimented with peripherally blurring game frame images in order to imitate the perceptual input of humans. This is a possible way of letting an agent develop an attention mechanism that is similar to human overt visual attention. We have shown how foveating input images can be integrated into the training of a DQN. We demonstrate that our foveating approach works well on a toy problem. In the case of an agent playing Asterix, our approach leads to a different behaviour than we expected; however, further tweaks and experiments may improve our results further.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., ... Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems [Software available from tensorflow.org]. https://www.tensorflow.org/

Atrey, A., Clary, K., & Jensen, D. (2019). Exploratory not explanatory: Counterfactual analysis of saliency maps for deep reinforcement learning. *arXiv preprint arXiv:1912.05743*.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). OpenAI Gym.

Bylinskii, Z., Judd, T., Oliva, A., Torralba, A., & Durand, F. (2019). What do different evaluation metrics tell us about saliency models? *IEEE transactions on pattern analysis and machine intelligence*, *41*(3), 740–757.

Das, A., Agrawal, H., Zitnick, L., Parikh, D., & Batra, D. (2017). Human attention in visual question answering: Do humans and deep networks look at the same regions? *Computer Vision and Image Understanding*, *163*, 90–100.

De la Cruz, G. V., Du, Y., & Taylor, M. E. (2019). Pre-training with non-expert human demonstration for deep reinforcement learning. *The Knowledge Engineering Review*, *34*, e10.

Deng, T., Yan, H., Qin, L., Ngo, T., & Manjunath, B. S. (2020). How do drivers allocate their potential attention? Driving fixation prediction via convolutional neural networks. *IEEE Transactions on Intelligent Transportation Systems*, *21*(5), 2146–2154. https://doi.org/10.1109/TITS.2019.2915540

Greydanus, S., Koul, A., Dodge, J., & Fern, A. (2018). Visualizing and understanding Atari agents. In J. Dy & A. Krause (Eds.), *Proceedings of the 35th international conference on machine learning* (pp. 1792–1801). PMLR. https://proceedings.mlr.press/v80/greydanus18a.html

Guo, S. S., Zhang, R., Liu, B., Zhu, Y., Ballard, D., Hayhoe, M., & Stone, P. (2021). Machine versus human attention in deep reinforcement learning tasks. *Advances in Neural Information Processing Systems*, *34*, 25370–25385.

Hester, T., Vecerik, M., Pietquin, O., Lanctot, M., Schaul, T., Piot, B., Horgan, D., Quan, J., Sendonaris, A., Osband, I., Dulac-Arnold, G., Agapiou, J., Leibo, J., & Gruslys, A. (2018). Deep Q-learning from demonstrations. *Proceedings of the AAAI Conference on Artificial Intelligence*, *32*(1).

Itti, L., & Koch, C. (2000). A saliency-based search mechanism for overt and covert shifts of visual attention. *Vision Research*, *40*(10), 1489–1506.

Judd, T., Ehinger, K., Durand, F., & Torralba, A. (2009). Learning to predict where humans look. *2009 IEEE 12th international conference on computer vision*, 2106–2113.

Kempka, M., Wydmuch, M., Runc, G., Toczek, J., & Jaundefinedkowski, W. (2016). ViZDoom: A Doom-based AI research platform for visual reinforcement learning. *2016 IEEE Conference on Computational Intelligence and Games (CIG)*, 1–8. https://doi.org/10.1109/CIG.2016.7860433

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. https://doi.org/10.48550/ARXIV.1412.6980

Kümmerer, M., Wallis, T. S., & Bethge, M. (2015). Information-theoretic model comparison unifies saliency metrics. *Proceedings of the National Academy of Sciences*, *112*(52), 16054–16059.

Le Meur, O., & Baccino, T. (2013). Methods for comparing scanpaths and saliency maps: Strengths and weaknesses. *Behavior Research Methods*, *45*(1), 251–266.

Li, Y., Sycara, K., & Iyer, R. (2018). Object-sensitive deep reinforcement learning. *arXiv preprint arXiv:1809.06064*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533.

Palazzi, A., Abati, D., Calderara, S., Solera, F., & Cucchiara, R. (2019). Predicting the driver's focus of attention: The DR(eye)VE Project. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *41*(7), 1720–1733. https://doi.org/10.1109/TPAMI.2018.2845370

Pramod, R. T., Katti, H., & Arun, S. P. (2022). Human peripheral blur is optimal for object recognition. *Vision Research*, *200*, 108083.

Puri, N., Verma, S., Gupta, P., Kayastha, D., Deshmukh, S., Krishnamurthy, B., & Singh, S. (2020). Explain your move: Understanding agent actions using specific and relevant feature attribution. *International Conference on Learning Representations (ICLR)*.

Saran, A., Zhang, R., Short, E. S., & Niekum, S. (2020). Efficiently guiding imitation learning algorithms with human gaze. *CoRR*, *abs/2002.12500*. https://arxiv.org/abs/2002.12500

Schaul, T., Quan, J., Antonoglou, I., & Silver, D. (2015). Prioritized experience replay. https://doi.org/10.48550/ARXIV.1511.05952

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., & Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature*, *529*(7587), 484–489.

Sundararajan, M., Taly, A., & Yan, Q. (2017). Axiomatic attribution for deep networks. *Proceedings of the 34th International Conference on Machine Learning*, *70*, 3319–3328.

van Hasselt, H., Guez, A., & Silver, D. (2015). Deep reinforcement learning with double q-learning. https://doi.org/10.48550/ARXIV.1509.06461

Varadarajan, M., Lamberta, B., Chargin, W., Walden, R. J., Sugiyama, A., Daoust, M., Aradwad, M., Bileschi, S., & zac-hopkinson. (2022). https://www.tensorflow.org/tensorboard/get_started

Wang, Z., Schaul, T., Hessel, M., van Hasselt, H., Lanctot, M., & de Freitas, N. (2015). Dueling network architectures for deep reinforcement learning. https://doi.org/10.48550/ARXIV.1511.06581

Zhang, R., Liu, Z., Zhang, L., Whritner, J. A., Muller, K. S., Hayhoe, M. M., & Ballard, D. H. (2018). AGIL: learning attention from human for visuomotor tasks. *CoRR*, *abs/1806.03960*. http://arxiv.org/abs/1806.03960

Zhang, R., Walshe, C., Liu, Z., Guan, L., Muller, K. S., Whritner, J. A., Zhang, L., Hayhoe, M., & Ballard, D. (2019). *Atari-HEAD: Atari Human Eye-Tracking and Demonstration Dataset* (Version 4). Zenodo. https://doi.org/10.5281/zenodo.3451402

Zhaoping, L. (2014). *Understanding vision: Theory, models, and data*. OUP Oxford.

Appendix

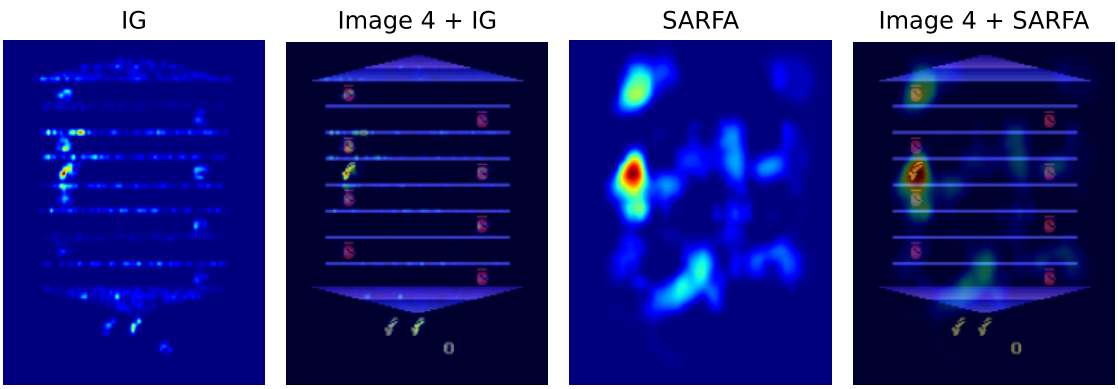

Figure A1: Comparison of an Integrated Gradients (IG) attribution mask (Sundararajan et al. 2017) and a perturbation-based SARFA saliency map (Puri et al. 2020) for the same input frame stack.
Outer left: Integrated Gradients attribution mask. Left: IG mask overlaid on top of the final frame in the four-frame stack. Right: SARFA saliency map (calculated by perturbing one fourth of all pixels in the game frame image, and then scaling the mask up to the full frame resolution). Outer right: SARFA saliency map overlaid on top of the final frame.

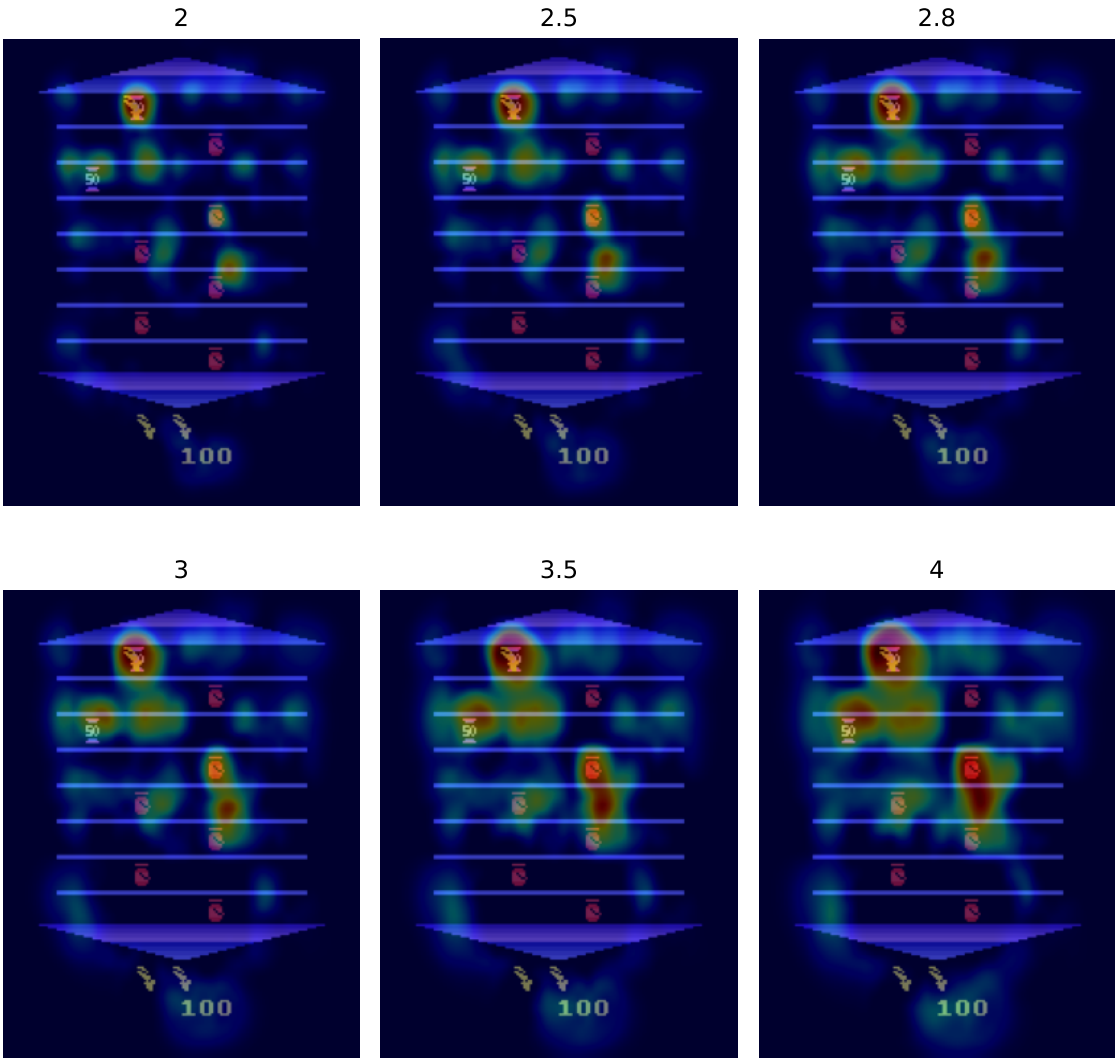

Figure A2: The effect of different values of σ on creating the mask. All resulting SARFA saliency heatmaps are shown overlaid on top of the final frame in the four-frame stack.
Above each image is the value of σ used for the Gaussian mask creation.

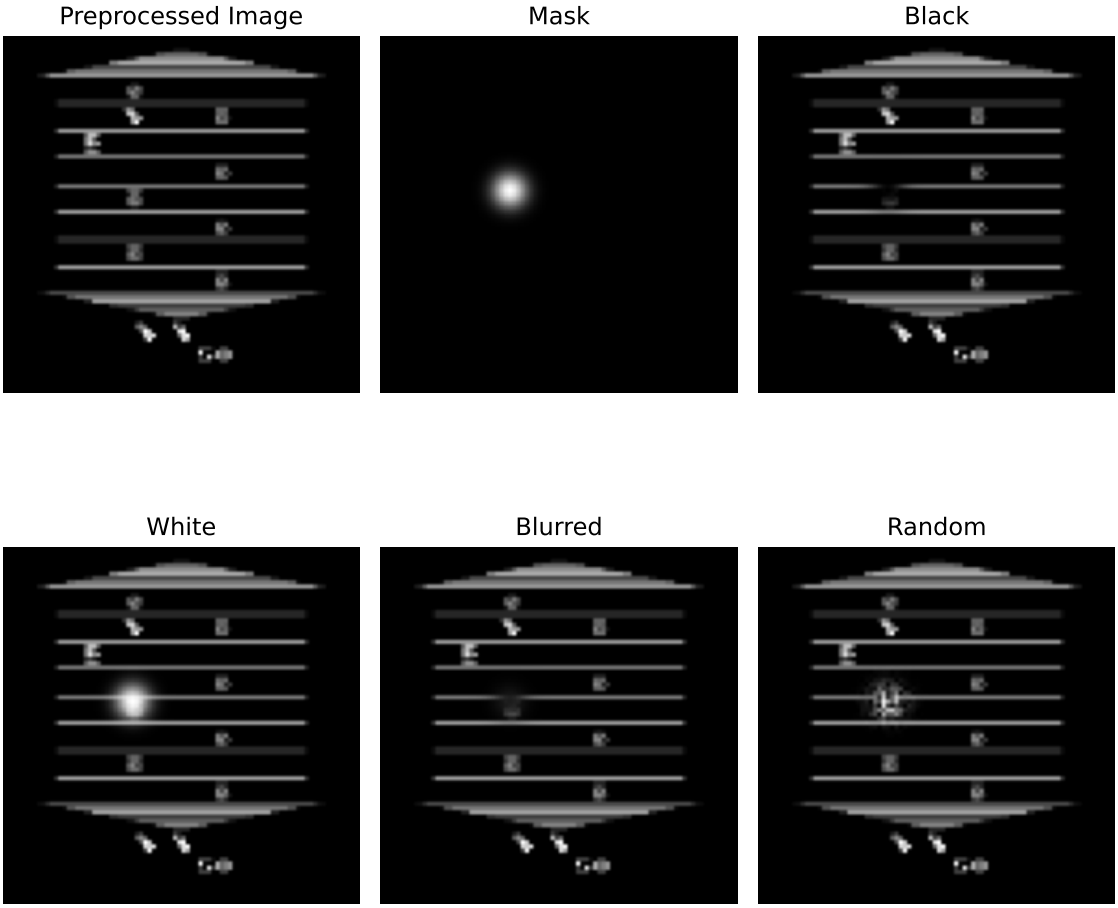

Figure A3: Comparing different perturbation modes by examples for each perturbation. Top left: The resized grayscale frame image that will be perturbed. Top middle: The mask used for the perturbation. Top right: The image perturbed with a black perturbation. Bottom left: The image perturbed with a withe perturbation (maximum pixel value in this image). Bottom middle: The image perturbed with a blurred perturbation (of the image itself). Bottom right: The image perturbed with a random perturbation in the range of image's values.

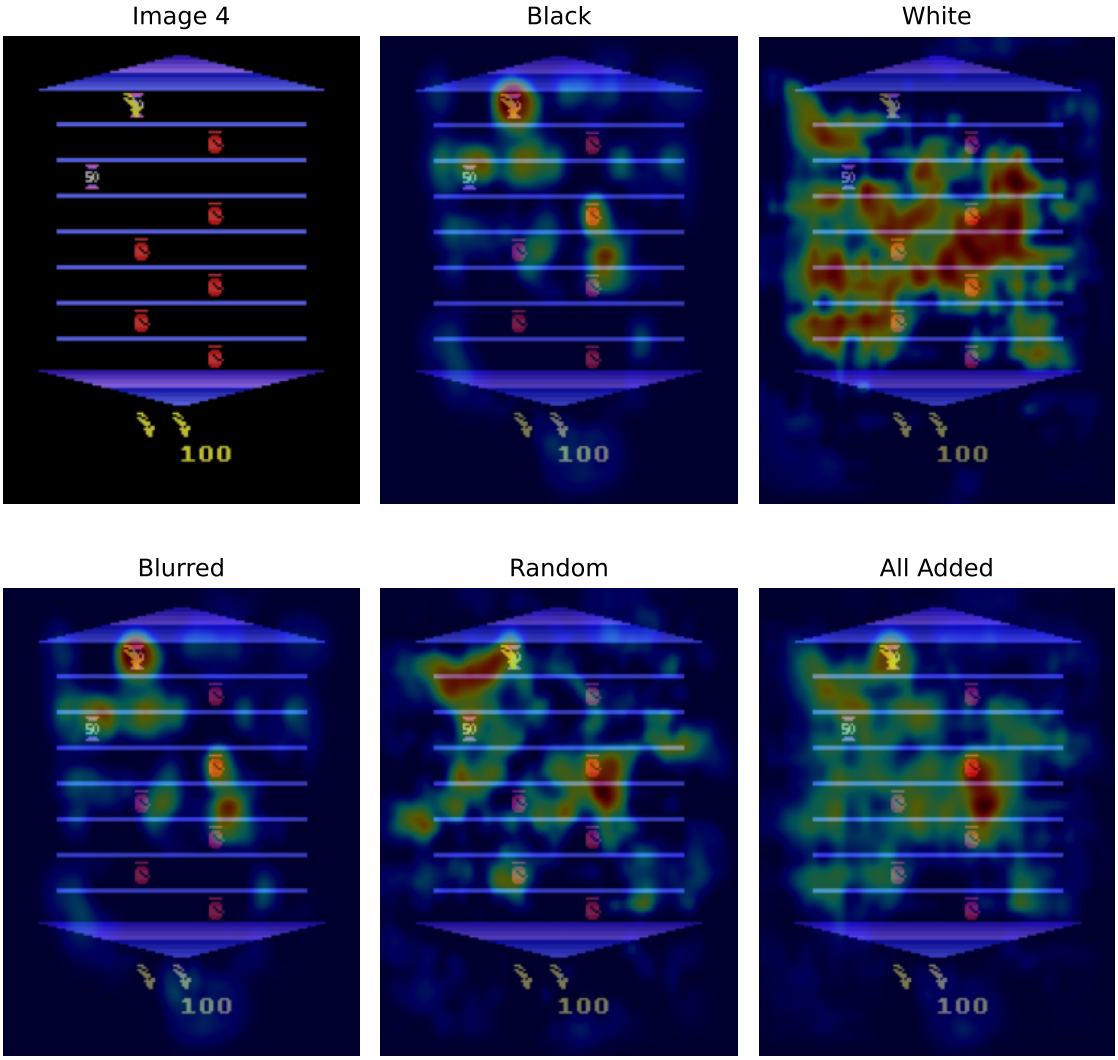

Figure A4: Showing example saliency maps created by using different perturbation modes, as used for our decision to use black. All resulting SARFA saliency maps are shown overlaid over the final input image.
From top left to bottom right: Original final frame image; black perturbation saliency map; white perturbation saliency map; original image blurred perturbation saliency map; random perturbation saliency map; all previously mentioned saliency maps added up.