

Dynamically Choosing the Number of Heads in Multi-Head Attention

Fernando Fradique Duarte¹^a, Nuno Lau²^b, Artur Pereira²^c and Luís Paulo Reis³^d

¹*Institute of Electronics and Informatics Engineering of Aveiro, University of Aveiro, Aveiro, Portugal*

²*Department of Electronics, Telecommunications and Informatics, University of Aveiro, Aveiro, Portugal*

³*Faculty of Engineering, Department of Informatics Engineering, University of Porto, Porto, Portugal*

Keywords: Deep Reinforcement Learning, Multi-Head Attention, Advantage Actor-Critic.

Abstract: Deep Learning agents are known to be very sensitive to their parameterization values. Attention-based Deep Reinforcement Learning agents further complicate this issue due to the additional parameterization associated to the computation of their attention function. One example of this concerns the number of attention heads to use when dealing with multi-head attention-based agents. Usually, these hyperparameters are set manually, which may be neither optimal nor efficient. This work addresses the issue of choosing the appropriate number of attention heads dynamically, by endowing the agent with a policy π^h trained with policy gradient. At each timestep of agent-environment interaction, π^h is responsible for choosing the most suitable number of attention heads according to the contextual memory of the agent. This dynamic parameterization is compared to a static parameterization in terms of performance. The role of π^h is further assessed by providing additional analysis concerning the distribution of the number of attention heads throughout the training procedure and the course of the game. The Atari 2600 videogame benchmark was used to perform and validate all the experiments.


1 INTRODUCTION


With the advent of Deep Learning (DL), careful engineering and domain expertise began to be replaced by representation learning methods, whereby the representations (or features) are learned from the data by the learning procedure as opposed to being derived by domain experts (LeCun et al., 2015; Bengio et al., 2021). Some examples of this work include (Sermanet et al., 2012; Srivastava et al., 2015; Xu et al., 2015) in vision, (Vaswani et al., 2017; Graves et al., 2013; Bahdanau et al., 2015) in Natural Language Processing (NLP), (Humphrey et al., 2012) in signal processing and (Mnih et al., 2015; Zambaldi et al., 2019; Mott et al., 2019; Ha & Schmidhuber, 2018; Sorokin et al., 2015; Silver et al., 2016) in Deep Reinforcement Learning (DRL). However, DL models may be hard to train if not properly parameterized. This often involves setting


suitable values to a myriad of hyperparameters, which may not be trivial (Bengio, 2012).


Attention-based DRL agents, the focus of this work, further complicate this issue due to the additional parameterization associated to the computation of their attention function. The number of attention heads to use when dealing with multi-head attention-based agents such as the ones proposed in (Mott et al., 2019) and (Zambaldi et al., 2019) is an example of this. The values for these hyperparameters are usually set manually and their values remain fixed throughout the learning process. This may be undesirable for various reasons.

First, deriving a static value implies some kind of hyperparameter search, which may be costly (or even unfeasible) both in terms of computational time and resources. Furthermore, this cost cannot be amortized most of the time since the value derived is task specific and a new value must be derived each time. Second, the complexity of the task may vary throughout the learning process. As an example, in

^a <https://orcid.org/0000-0002-9503-9084>

^b <https://orcid.org/0000-0003-0513-158X>

^c <https://orcid.org/0000-0002-7099-1247>

^d <https://orcid.org/0000-0002-4709-1718>

most videogames the complexity of the task increases throughout the course of the game. In these cases, using non-adjustable values for these hyperparameters may be neither optimal (e.g., too many attention heads when the game is easy and too few when the game is hard) nor efficient (e.g., too many attention heads when in easier situations). In either case this may hinder the learning process.

This work addresses the issue of choosing the appropriate number of attention heads dynamically. More specifically, at each time step of agent-environment interaction, a policy π^h , trained with policy gradient, chooses the number of attention heads that should be used to derive the attention function, according to the contextual memory of the agent. This dynamic attention-based agent, enhanced with policy π^h , is compared to a similar agent parameterized statically with 4 attention heads in terms of performance. The role of π^h on the behavior of the agent is further assessed by providing additional analysis concerning the distribution of the number of attention heads throughout the training procedure and the course of the game. All the experiments were performed and validated on the Atari 2600 videogame benchmark.

It should be noted that this work does not present an efficient implementation to leverage the potential computational gains (in terms of time and resources) derived from using a dynamic approach. The focus is mainly on assessing the effects of such an approach on the learning process and the performance of the agent. The remainder of the paper is structured as follows. Section 2 presents the problem formulation, including a brief overview of the technical background and a motivational example. Section 3 discusses the experimental setup, which includes the presentation of the methods proposed and the training setup. Section 4 presents the experiments carried out and discusses the results obtained. Finally, section 5 presents the conclusions.

2 PROBLEM FORMULATION

This section starts by presenting a high-level view of the technical background. Next, a motivational example is discussed. Finally, the problem formulation is presented. This includes the rationale and the main research goals underlying this work.

2.1 Multi-Head Attention

Attention-based agents have achieved a lot of success in many areas of Artificial Intelligence (AI), such as

NLP (Vaswani et al., 2017;Bahdanau et al., 2015) and DRL (Zambaldi et al., 2019;Mott et al., 2019;Sorokin et al., 2015). While many different variants of attention have been proposed in the literature, this work focuses on the attention formulation proposed in (Vaswani et al., 2017) and more specifically the scaled dot-product variant, computed as in Equation (1), where Q , K and V represent the queries, keys and values matrices, respectively and d_k denotes the dimension of the queries and keys vectors.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (1)$$

At a very high-level, an attention function can be described as mapping a query and a set of key-value pairs to an output, computed as a weighted sum of the values, where each weight is derived by a compatibility function between the query and the corresponding key. The queries, keys and values vectors composing the Q , K and V matrices can be derived from many different sources. In DRL for example, these vectors may be derived from the feature maps output by a Convolutional Neural Network (CNN) as in (Zambaldi et al., 2019;Sorokin et al., 2015) or from the hidden state of a Recurrent Neural Network (RNN), either a Long Short-Term Memory (LSTM) (Hochreiter & Schmidhuber, 1997) or a Convolutional LSTM (Shi et al., 2015) or both as in (Mott et al., 2019). Usually, several attention functions, also denoted as attention heads, are derived in parallel (to increase the expressive power of the model) and then aggregated together, resulting in multi-head attention.

2.2 Motivational Example

Figure 1 depicts the attention maps Λ_i , $i = \{1,2,3,4\}$ computed by *TDA (4H)*, an implementation of Soft Top-Down Attention (Mott et al., 2019) with 4 attention heads set statically.

As can be seen, the attention maps derived exhibit some interesting behaviors. One such behavior concerns redundancy and/or possibly complementarity between the attention maps. As examples of this, in Breakout Λ_3 and Λ_4 (left column) and Λ_1 and Λ_4 (right column) and in Seaquest Λ_1 and Λ_2 (right column), all exhibit some level of redundancy/complementarity between them.

The other behavior observed is a change of focus throughout the course of the game. As an example of this, in Breakout Λ_2 is mainly focused on the ball (left column), but as soon as the agent starts applying the flanking strategy to destroy the bricks from above, Λ_2 changes its focus mostly to the top area near the score (left column).

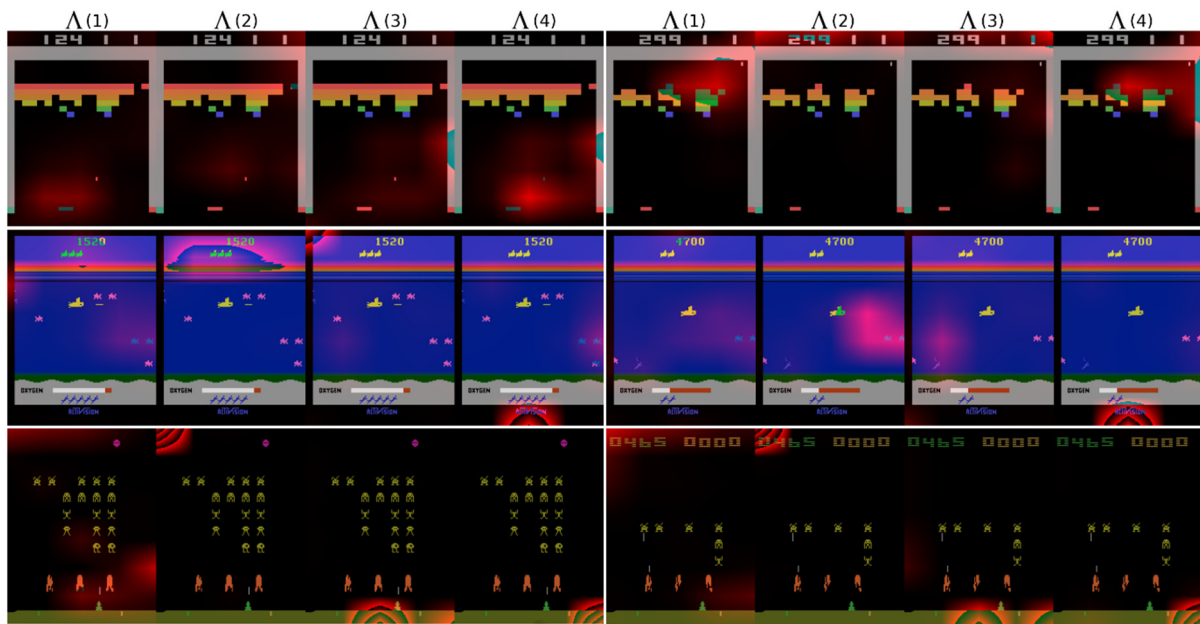


Figure 1: Attention maps (the red blobs) for the TDA (4H) agent for the Atari games Breakout (top row), Seaquest (middle row) and SpaceInvaders (bottom row).

As another example, in Seaquest Λ_3 is mostly focused on the top-left corner of the screen (left column), but as the game progresses, shifts its focus to the middle-left edge of the screen (right column). Finally, some attention maps remain focused on the same location throughout the entire course of the game, Λ_4 in Seaquest is an example of this, while others focus on apparently unimportant locations (Λ_4 in SpaceInvaders).

2.3 Dynamically Choosing the Number of Attention Heads

While some of the observations discussed in the previous section may be partially due to lack of training or the underlying architecture of the agents, it seems to be the case that statically setting the number of attention heads to a non-adjustable value is not the optimal solution. The redundancies observed and the focus on unimportant locations may be indications of this. Furthermore, choosing this value may not always be trivial. In the case of Atari, for example, training an agent may take anywhere from several hours to several days, making hyperparameter search an expensive exercise.

Dynamically choosing this value, avoids this issue and may present additional advantages, including: a faster learning process, better finetuned to the specificities of the task, a more efficient training procedure (resource and timewise) and more

performant agents, better optimized to the nuances that may occur throughout the course of the task. Motivated by this, the present work proposes the following research goals:

- **G1.** Some tasks were designed to be progressively harder (e.g., Atari games). In this case it is reasonable to assume that initially the agent would need to compute less attention maps, thus easing the learning process. Moreover, the attention maps derived could potentially be more focused and present less redundancy, with new maps being derived only when a new aspect of the task must be attended to. These assumptions are assessed by analyzing the distribution of the number of attention heads throughout the learning process and the course of the task. The quality and the behavior of the attention maps derived are also assessed via visual inspection;
- **G2.** The Atari videogame benchmark offers a suite of games featuring different challenges. Intuitively, harder games may need more attention heads when compared to easier games. Tailoring the number of attention heads according to the characteristics of the game and throughout its course, according to its interaction history, may potentially improve the performance of the agent. The discussion of the results provides a performance comparison between the agents enhanced with policy π^h and a baseline agent parameterized statically.

It should be noted that the implementation used does not leverage the potential computational gains (time and resource-wise) derived from using a dynamic approach. Although such an implementation is currently being worked on, it is slow to train and needs further finetuning. Therefore, this work focuses on assessing the effects of the proposed approach on the learning process and the performance of the agents.

3 EXPERIMENTAL SETUP

This section presents the agents implemented and tested. The training setup is presented at the end of the section and includes the testing and training protocols used and the parameterization of the agents.

3.1 Baseline Agent

The baseline agent, *TDA (4H)*, consists of an implementation of (Mott et al., 2019). Architecturally, the agent is composed of 4 main modules, namely: vision encoder, query network, memory module and the policy. More specifically, the vision encoder consists of 4 convolutional layers configured with (1, 32, 64, 64) input and (32, 64, 64, 256) output channels, kernel sizes (8, 4, 4, 4), strides (4, 2, 2, 1) and no padding, respectively. Each layer is followed by batch normalization (Ioffe & Szegedy, 2015) and a ReLU nonlinearity. The Convolutional LSTM rnn_{down} sitting on top of this CNN was configured with 64 input/output channels, kernel size 3 and stride 1 with padding 1.

The query network is composed of 3 linear layers with sizes (256, 128, 1280), respectively, each followed by layer normalization (Ba et al., 2016) and a ReLU. The attention function computation is similar to (Mott et al., 2019). The LSTM rnn_{top} comprising the memory module was configured with size 256. Finally, the policy module π consists of a linear layer of size 128, followed by layer normalization and a ReLU. This layer feeds two other linear layers: the actor, which chooses the action a_t to take at each timestep and the critic, which computes the value of each state $V(o_t)$. Figure 2 presents a more pictorial depiction of this. The agent was statically parameterized with 4 attention heads.

3.2 Dynamic Multi-Head Attention Agent

The dynamic multi-head attention agent (*DTDA*) is enhanced with policy π^h , responsible for choosing the number of attention heads $n_t = \pi^h$ to use at each timestep t . π^h is implemented similarly to the policy module and shares its input. In practice and for simplicity, the agent computes a fixed maximum number of attention heads, and the excess heads are zeroed out, i.e., $q_i = ans_i = \Lambda_i = 0$ for $i > \pi^h$, where 0 denotes a vector of zeros. The remaining architecture is similar to Figure 2.

3.3 Training Setup

All agents were trained for a minimum of 16,800,000 frames, similarly to (Machado et al., 2018), using the Advantage Actor-Critic (A2C) algorithm (Mnih et al., 2016). Adam (Kingma & Ba, 2015) was used as the optimizer, the learning rate was set to $1e-4$ and the loss was computed using Generalized Advantage Estimation with $\lambda=1.0$ (Schulman et al., 2016). The input image is converted to grayscale and cropped to 206 by 158 pixels with no rescaling and the internal state of the memory module is never reset during training.

The training results were computed at every 240,000th frame over a window of size $w=50$ and correspond to the return scores (averaged over all the agents) obtained during training in the last w episodes. Each trained agent played 100 games to derive the test returns per episode. The results include the overall median and the average return and standard deviation obtained by the best agent. Two agents were used to perform each experiment using the Atari 2600 videogame platform, available via the OpenAI Gym toolkit (Brockman et al., 2016). The one-way ANOVA and the Kruskal-Wallis H-test were used as the statistical significance tests ($\alpha=0.05$). H_0 considers that all the agents have the same return mean results. Table 1 presents the remaining parameterization.

Table 1: Hyperparameters. Values annotated with * denote a scaling factor. γ denotes the discounting factor.

| Entropy | Critic | Reward clipping | γ |
|----------|---------|-----------------|----------|
| $1e-2^*$ | 0.5^* | [-1, 1] | .99 |

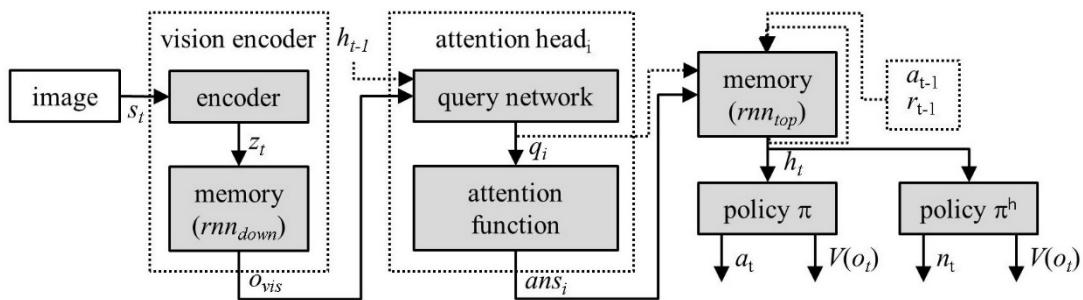


Figure 2: Overall architecture. h_{t-1} , a_{t-1} and r_{t-1} denote the previous state of rnn_{top} , the previous action performed, and the previous reward obtained, respectively. The number of attention heads is $i = \{1,2,3,4\}$ for the static agent and $i = \{1,2,\dots,n_t\}$ for the dynamic agent. q_i denotes the i^{th} query and ans_i denotes the output from the i^{th} attention head.

4 EXPERIMENTAL RESULTS

This section first presents the performance results obtained by the agents. Next, π^h is further assessed by providing additional analysis concerning the distribution of the number of attention heads throughout the training procedure and the course of the game. An analysis of the attention maps derived is also presented, followed by the discussion of the results obtained.

4.1 Performance Results

The agents tested were: *TDA (4H)*, the baseline agent statically parameterized with 4 attention heads and *DTDA (4H)* and *DTDA (8H)*, the dynamically parameterized agents with a maximum of 4 and 8 attention heads, respectively. Figure 3 presents the training and test results obtained.

Concerning the training results, dynamically choosing the number of attention heads does not seem to provide a significant improvement to the learning process. In terms of the test results, the performance of the agents is similar for Breakout and SpaceInvaders, whereas for Seaquest *DTDA (8H)* performed the best. *DTDA (4H)* on the other hand, suffered from some instability during training which resulted in 2 agents of very different quality with mean average return 47,469 and 24,144, respectively.

4.2 Distribution of the Number of Attention Heads

The distribution of the number of attention heads throughout the training process is depicted in Figure 4. As depicted, the distribution does not seem to converge to any stable configuration. As training progresses different values of n_t gain or lose prominence. For example, in Breakout (left), $n_t = 3$

seems to be losing preference consistently, but at the end of training begins to gain some preference again. A similar observation can be made for $n_t = 5$ in SpaceInvaders (middle) and $n_t = 2$ and $n_t = 5$ in Seaquest (right). This instability may be due to lack of training. The distribution of the number of attention heads throughout the course of the game is depicted in Figure 5. As can be seen, the strategies derived by the agents are very different, even when using the same model (trained with different initialization seeds).

While most of the strategies derived use several different values of n_t throughout the course of the game, some of these strategies approach a static parameterization. *DTDA (4H)* in SpaceInvaders with $n_t = 3$ and $n_t = 4$ is an example of this. Also, some of these strategies seem to agree (to some extent) with the intuition that the values of n_t should increase as the game becomes harder. For example, one of the *DTDA (8H)* agents in Seaquest favors $n_t = 2$ initially and later switches to $n_t = 3$. Contrary to this, the other *DTDA (8H)* agent for Seaquest favors $n_t = 5$ initially and as the game progresses switches to $n_t = 2$. Also, one of the *DTDA (4H)* agents for Seaquest favors $n_t = 3$ initially and later switches to $n_t = 1$.

In the case of Breakout and SpaceInvaders assessing the degree of difficulty of the game is harder and such analysis is not as straightforward. For example, in SpaceInvaders as the game progresses, the enemies get closer to the agent, making the game more challenging, but on the other hand there may be less enemies to attend to. The strategies derived are also sometimes contradictory, concerning the difficulty of the game. For example, most of the strategies derived for Breakout and Seaquest favor values of $n_t \leq 4$, although Seaquest may be considered more challenging than Breakout.

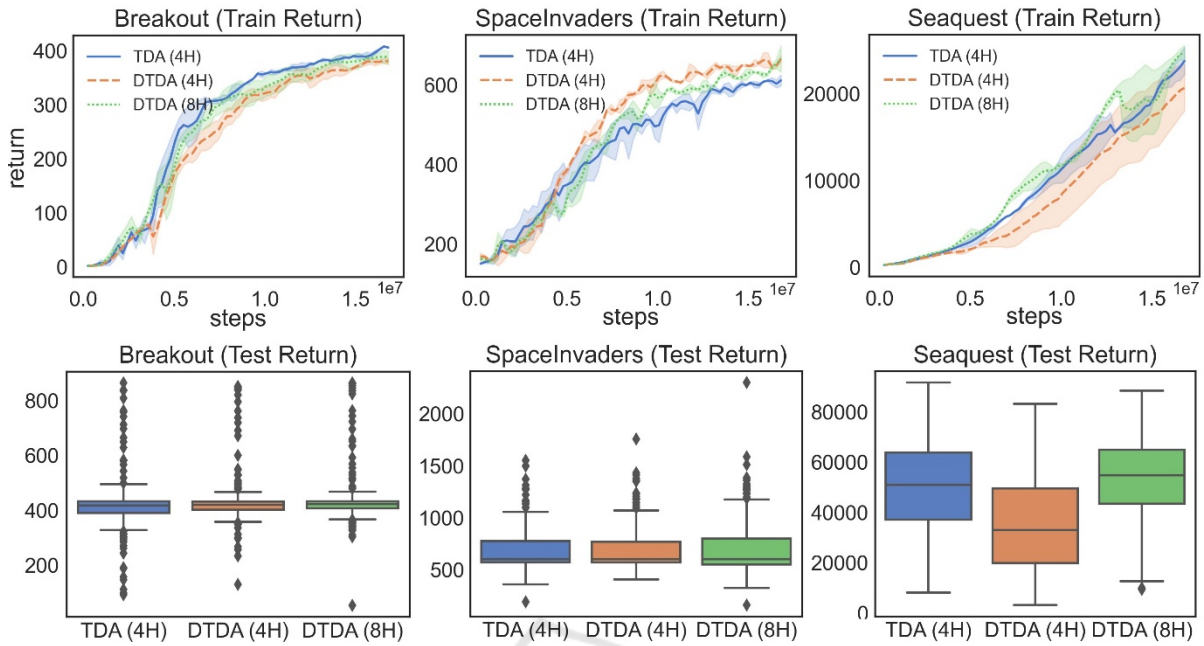


Figure 3: Test return: (Breakout) TDA (4H) 417 with best (429/124), DTDA (4H) 420 with best (457/112) and DTDA (8H) 423 with best (444/123), (SpaceInvaders) TDA (4H) 600 with best (706/225), DTDA (4H) 600 with best (692/234) and DTDA (8H) 600 with best (758/309), (Seaquest) TDA (4H) 50,850 with best (51,459/17,817), DTDA (4H) 32,885 with best (47,469/16,499) and DTDA (8H) 54,735 with best (57,971/14,456). The results for Breakout and SpaceInvaders are not statistically significant (p-value 0.10 and 0.87, respectively).

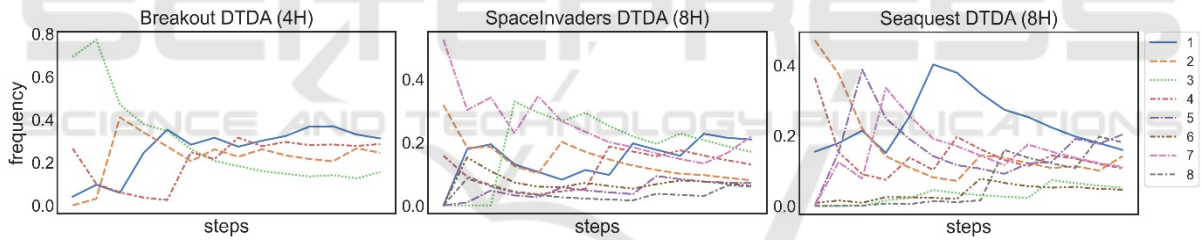


Figure 4: Distribution of the number of attention heads throughout the training process. The results were computed at every 1,200,000 frames and averaged over 10 games.

4.3 Visualization of the Attention Maps

Figure 6 depicts the visualizations of the attention maps derived by the *DTDA (4H)* agents. In *SpaceInvaders*, both strategies produced attention maps with varying degrees of redundancy/complementarity. This is most prevalent in the 3 heads strategy (top row, left column). The other strategy (last 2 rows, left column) also suffers from this issue but was able to produce some specialized attention maps. Λ_2 focuses mainly on the agent, whereas Λ_3 focuses mostly on the enemy ships (also mildly on the agent). These (redundancy and or complementarity) issues prevail in *Seaquest*. In this case both strategies were able to produce specialized attention maps.

In the $n_t = \{1, 2, 4\}$ strategy (top row, right column), Λ_2 focuses almost exclusively on a delimited rectangular-like area in the middle of the screen, whereas Λ_4 focuses mostly on the borders of the screen. In the $n_t = \{1, 3\}$ strategy (middle row, right column), Λ_2 focuses mostly on the oxygen meter (and the top left corner). Examples of situations where the $n_t = \{1, 3\}$ strategy uses a single attention head, are depicted on the last row, right column. As can be seen these attention maps are more generalist and focus on several elements simultaneously.

Figure 7 depicts the visualizations of the attention maps derived by the *DTDA (8H)* agents. The $n_t = \{2, 3, 4\}$ strategy (top row) presents an interesting behavior. When the game seems to be easier, the agent relies mostly on Λ_1 and Λ_2 (left column). Λ_1 is more generalist and provides the surfacing cue to the agent (right column), whereas Λ_2 focuses on a very specific rectangular-like area near the bottom right corner of the screen.

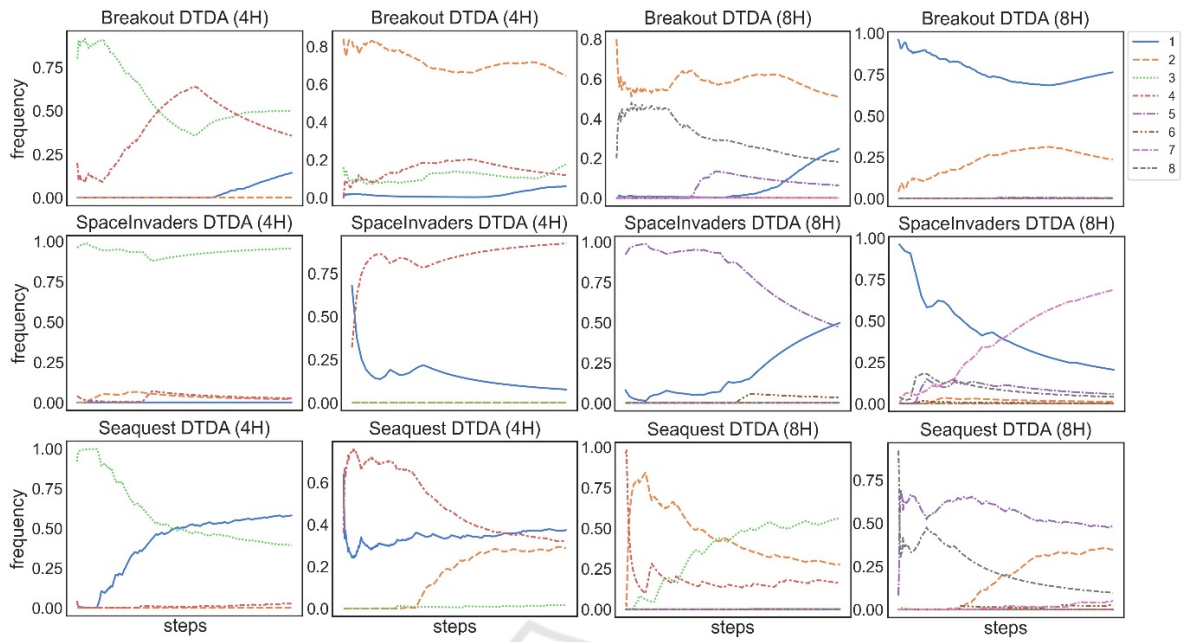


Figure 5: Distribution of the number of attention heads throughout the course of the game. The results were computed at every 240,000 frames over a single game. Best strategies derived for the number of heads n_t : (Breakout) DTDA (4H) approaching a static parameterization with $n_t = 2$, (SpaceInvaders) DTDA (8H) with $n_t = 1$ initially, gradually favoring $n_t = 7$, (Seaquest) DTDA (8H) with $n_t = 2$ initially, gradually favoring $n_t = 3$.

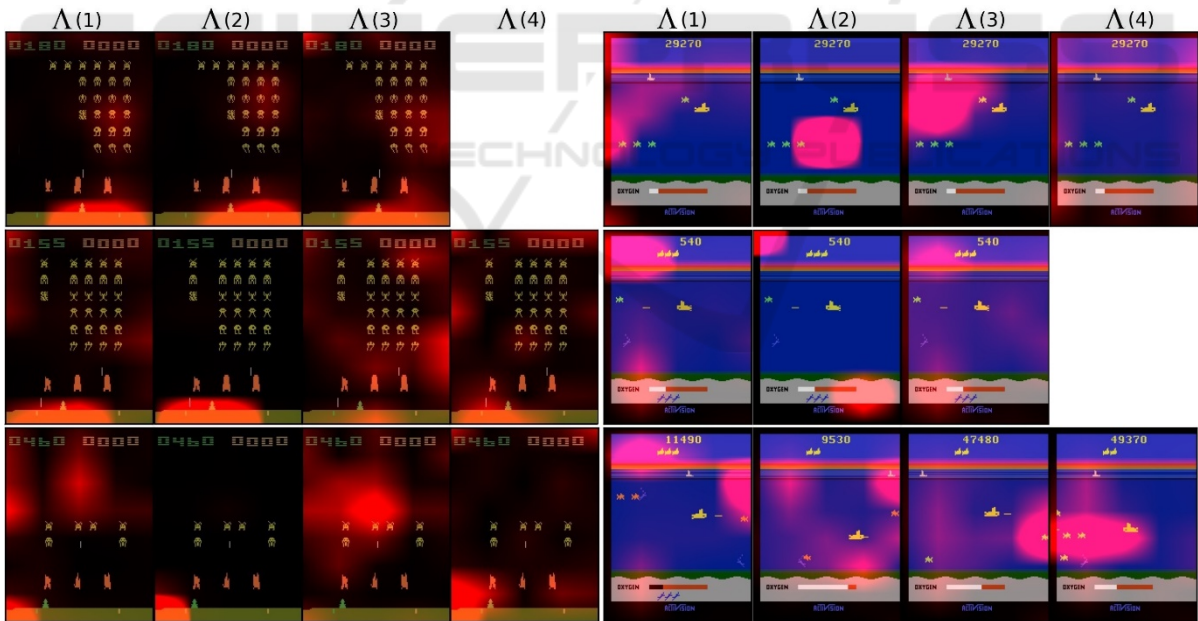


Figure 6: (left) Attention maps derived by DTDA (4H) for SpaceInvaders. (right) Attention maps derived by DTDA (4H) for Seaquest.

As the game progresses the agent starts computing Λ_3 more often. Λ_3 seems to provide mostly redundancy and complementarity to Λ_t (middle column). Λ_4 is rarely computed and focuses on the top left corner of the screen (not shown). The other strategy $n_t = \{2, 5, 8\}$ (bottom row)

is not as interesting and is harder to interpret given the high degree of redundancy and complementarity between the various attention maps.

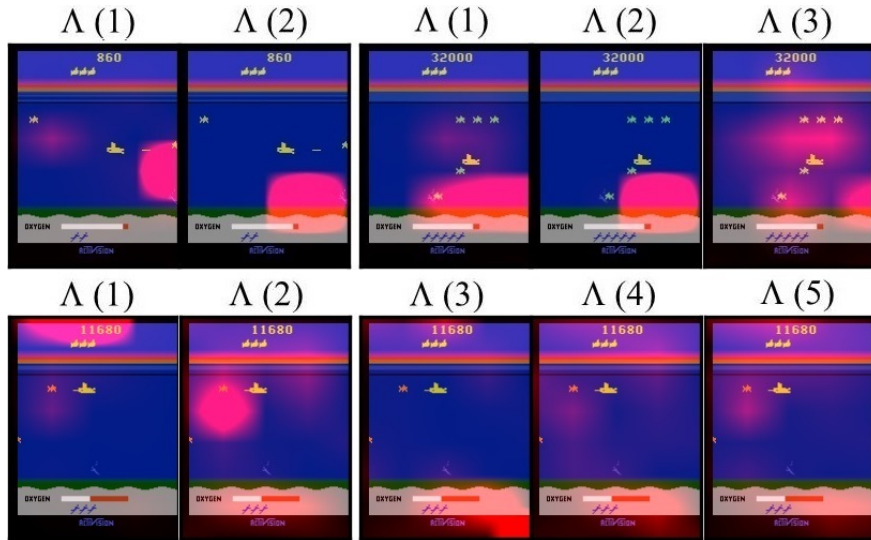


Figure 7: Attention maps derived by the DTDA (8H) agents for Seaquest.

4.4 Discussion

Overall, π^h did not improve (or eased) the learning process. On one hand, π^h is another module that must be optimized, which in turn may pose more difficulties to the learning process. On the other hand, dynamically changing the value of n_t may also introduce some instability, since the optimization process must switch between different ‘operating modes’ introduced by the number of attention heads being used at each timestep. At a high-level, π^h can be thought of as a kind of tradeoff between parameterization complexity and optimization load. It is a question of whether easing the parameterization burden by introducing more parameters to optimize provides a real benefit in practice.

Nevertheless, as shown by the test results, π^h was able to improve the performance of the agent in some cases, such as in Seaquest, while at the same time obtaining similar performance results to those obtained by a statically parameterized agent, without the need to perform hyperparameter search. Ultimately, the benefits of the approach proposed can become even more noticeable if an efficient implementation can take advantage of its potential computational gains. However, such an implementation was not presented in this work.

Finally, both the strategies derived for n_t as well as the quality of the resulting attention maps seem to be very dependent on the optimization process. As shown by the results, different agents as well as agents sharing the same model, discovered different strategies to exploit the number of attention heads, with varying performance results. In some cases, different strategies obtained very similar results,

while in other cases the strategies performed very differently.

This work proposed a simple policy π^h which does not consider the number or the quality of the attention maps nor the redundancy or relationships between them. A more sophisticated implementation may take these aspects into account to derive better results. Finally, such an implementation may also tackle the possible tradeoff between using less attention heads, therefore obtaining attention maps that are more generalist, or using more attention heads in an attempt to obtain attention maps that are more focused and specialized but that may also present more redundancy and or complementarity between them.

5 CONCLUSIONS

This work proposed a dynamic approach to choose the number of attention heads to use at each timestep of agent-environment interaction, based solely on the contextual memory of the agent and without the need to perform hyperparameter search. When compared to a statically parameterized agent, the approach proposed was able to improve the performance of the agent in Seaquest while obtaining similar results in Breakout and SpaceInvaders. These initial results are very promising and can be leveraged to derive better implementations more suited to tackling the limitations discussed. Furthermore, the benefits of this approach can become more noticeable if an efficient implementation can take advantage of its potential computational gains.

ACKNOWLEDGEMENTS

This research was funded by Fundação para a Ciência e a Tecnologia, grant number SFRH/BD/145723/2019 - UID/CEC/00127/2019.

REFERENCES

- Ba, L. J., Kiros, J. R., & Hinton, G. E. (2016). Layer Normalization. *CoRR*, *abs/1607.06450*.
- Bahdanau, D., Cho, K., & Bengio, Y. (2015). Neural Machine Translation by Jointly Learning to Align and Translate. In *3rd International Conference on Learning Representations, ICLR 2015*.
- Bengio, Y. (2012). Practical recommendations for gradient-based training of deep architectures. *CoRR*, *abs/1206.5533*.
- Bengio, Y., LeCun, Y., & Hinton, G. E. (2021). Deep learning for AI. *Commun. ACM*, *64*(7), 58–65.
- Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). OpenAI Gym. *CoRR*, *abs/1606.01540*.
- Graves, A., Mohamed, A., & Hinton, G. E. (2013). Speech recognition with deep recurrent neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2013*.
- Ha, D., & Schmidhuber, J. (2018). World Models. *CoRR*, *abs/1803.10122*.
- Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735–1780.
- Humphrey, E. J., Bello, J. P., & LeCun, Y. (2012). Moving Beyond Feature Design: Deep Architectures and Automatic Feature Learning in Music Informatics. In *Proceedings of the 13th International Society for Music Information Retrieval Conference, ISMIR 2012*. (pp. 403–408).
- Ioffe, S., & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015*. (Vol. 37, pp. 448–456). JMLR.org.
- Kingma, D. P., & Ba, J. (2015). Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015*.
- LeCun, Y., Bengio, Y., & Hinton, G. E. (2015). Deep learning. *Nat.*, *521*(7553), 436–444.
- Machado, M. C., Bellemare, M. G., Talvitie, E., Veness, J., Hausknecht, M. J., & Bowling, M. (2018). Revisiting the Arcade Learning Environment: Evaluation Protocols and Open Problems for General Agents. *J. Artif. Intell. Res.*, *61*, 523–562.
- Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T. P., Harley, T., Silver, D., & Kavukcuoglu, K. (2016). Asynchronous Methods for Deep Reinforcement Learning. In *Proceedings of the 33rd International Conference on Machine Learning, ICML 2016*. (Vol. 48, pp. 1928–1937). JMLR.org.
- Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M. A., Fidjeland, A., 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. *Nat.*, *518*(7540), 529–533.
- Mott, A., Zoran, D., Chrzanowski, M., Wierstra, D., & Rezende, D. J. (2019). Towards Interpretable Reinforcement Learning Using Attention Augmented Agents. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems, NeurIPS 2019*. (pp. 12329–12338).
- Schulman, J., Moritz, P., Levine, S., Jordan, M. I., & Abbeel, P. (2016). High-Dimensional Continuous Control Using Generalized Advantage Estimation. In *4th International Conference on Learning Representations, ICLR 2016*.
- Sermanet, P., Chintala, S., & LeCun, Y. (2012). Convolutional neural networks applied to house numbers digit classification. In *Proceedings of the 21st International Conference on Pattern Recognition, ICPR 2012*. (pp. 3288–3291).
- Shi, X., Chen, Z., Wang, H., Yeung, D.-Y., Wong, W.-K., & Woo, W. (2015). Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015*. (pp. 802–810).
- 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. P., Leach, M., Kavukcuoglu, K., Graepel, T., & Hassabis, D. (2016). Mastering the game of Go with deep neural networks and tree search. *Nat.*, *529*(7587), 484–489.
- Sorokin, I., Seleznev, A., Pavlov, M., Fedorov, A., & Ignateva, A. (2015). Deep Attention Recurrent Q-Network. *CoRR*, *abs/1512.01693*.
- Srivastava, N., Mansimov, E., & Salakhutdinov, R. (2015). Unsupervised Learning of Video Representations using LSTMs. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015*. (Vol. 37, pp. 843–852). JMLR.org.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is All you Need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*. (pp. 5998–6008).
- Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A. C., Salakhutdinov, R., Zemel, R. S., & Bengio, Y. (2015). Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015*. (Vol. 37, pp. 2048–2057). JMLR.org.
- Zambaldi, V. F., Raposo, D., Santoro, A., Bapst, V., Li, Y., Babuschkin, I., Tuyls, K., Reichert, D. P., Lillicrap, T.

P., Lockhart, E., Shanahan, M., Langston, V., Pascanu, R., Botvinick, M. M., Vinyals, O., & Battaglia, P. W. (2019). Deep reinforcement learning with relational inductive biases. In *7th International Conference on Learning Representations, ICLR 2019*.

