

Neuroevolution of Self-Attention Over Proto-Objects

Rafael C. Pinto

Federal Institute of Education, Science and Technology of
Rio Grande do Sul (IFRS)

Canoas, Brazil

Federal University of Rio Grande do Sul (UFRGS)

Porto Alegre, Brazil

rafael.pinto@canoas.ifrs.edu.br

Anderson R. Tavares

Federal University of Rio Grande do Sul (UFRGS)

Porto Alegre, Brazil

artavares@inf.ufrgs.br

Abstract

Proto-objects – image regions that share common visual properties – offer a promising alternative to traditional attention mechanisms based on rectangular-shaped image patches in neural networks. Although previous work demonstrated that evolving a patch-based hard-attention module alongside a controller network could achieve state-of-the-art performance in visual reinforcement learning tasks, our approach leverages image segmentation to work with higher-level features. By operating on proto-objects rather than fixed patches, we significantly reduce the representational complexity: each image decomposes into fewer proto-objects than regular patches, and each proto-object can be efficiently encoded as a compact feature vector. This enables a substantially smaller self-attention module that processes richer semantic information. Our experiments demonstrate that this proto-object-based approach matches or exceeds the state-of-the-art performance of patch-based implementations with 62% less parameters and 2.6 times less training time.

CCS Concepts

• **Computing methodologies** → **Genetic algorithms**; Continuous space search; Shape representations; *Sequential decision making*; **Neural networks**.

Keywords

Neuroevolution, Representation Learning

ACM Reference Format:

Rafael C. Pinto and Anderson R. Tavares. 2025. Neuroevolution of Self-Attention Over Proto-Objects. In *Genetic and Evolutionary Computation Conference (GECCO '25)*, July 14–18, 2025, Malaga, Spain. ACM, New York, NY, USA, 9 pages. <https://doi.org/10.1145/3712256.3726451>

1 Introduction

Visual attention mechanisms have emerged as a powerful solution for reducing computational complexity in high-dimensional perception tasks. By creating an information bottleneck between visual inputs and control networks, these mechanisms enable efficient processing of complex scenes [14]. Recent work has demonstrated that

Publication rights licensed to ACM. ACM acknowledges that this contribution was authored or co-authored by an employee, contractor or affiliate of a national government. As such, the Government retains a nonexclusive, royalty-free right to publish or reproduce this article, or to allow others to do so, for Government purposes only. *GECCO '25, July 14–18, 2025, Malaga, Spain*
© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM. ACM ISBN 979-8-4007-1465-8/2025/07
<https://doi.org/10.1145/3712256.3726451>

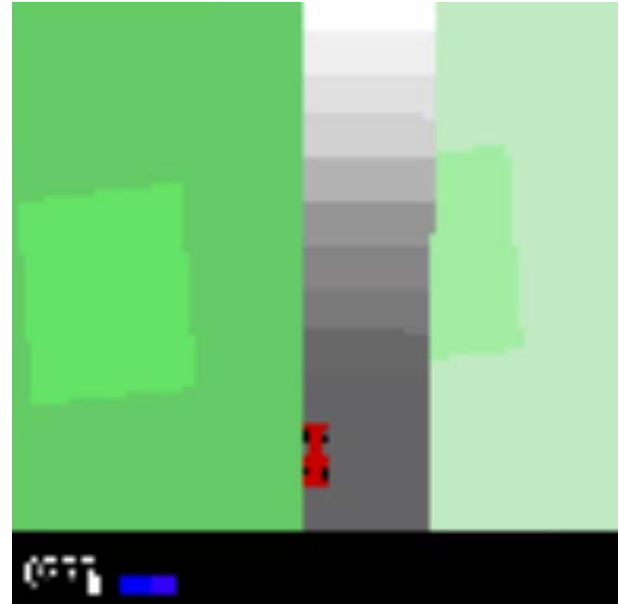


Figure 1: Our attentional agent is capable of focusing on entire uniform regions (e.g. large same-color regions on the image) instead of small fixed-size image patches.

evolving a hard-attention module jointly with an LSTM [10] controller can produce remarkably efficient agents that operate solely on small image patches [22]. This approach not only yielded neural networks orders of magnitude smaller than competing methods but also achieved state-of-the-art results in challenging Reinforcement Learning environments like Car Racing and Doom Take Cover [3]. The success stems from the attention layer's ability to filter irrelevant input regions, simplifying the controller's task while providing robust generalization and noise resistance.

We advance this line of research by replacing fixed-size, uniformly distributed patches with proto-objects – coherent regions of locally uniform visual features [6] – obtained through image segmentation [7] (Fig. 1). This shift in representation offers two key advantages. First, they provide a more compact representation, as most scenes decompose into fewer proto-objects than patches. Second, each proto-object encodes richer semantic information through a small descriptor vector capturing properties like shape, size, and color.

This proto-object approach enables a significantly streamlined architecture. The self-attention module becomes substantially smaller

while processing higher-level features, leading to improved selection and better-filtered information for the controller, which can also be simplified. Our results in the Car Racing and Doom Take Cover environments [3] demonstrate that this more efficient architecture matches or exceeds the performance of patch-based implementations while reducing the parameter count by 62% with 2.6 times faster training.

2 Background

Modeling human visual attention has been an active research area over the past 35 years. Many different models of attention were proposed, which in addition to providing theoretical contributions to neuroscience and psychology, have demonstrated successful applications in computer vision and robotics [2]. Early computational models focused primarily on bottom-up, saliency-based attention, while more recent approaches have incorporated top-down influences and object-based selection mechanisms.

The biological visual system provides crucial insights for designing efficient artificial vision systems. A fundamental constraint is that neural resources are limited – Koch et al. [11] demonstrated that retinal ganglion cells balance metabolic costs against information transmission, achieving highly efficient coding despite using relatively low firing rates. This suggests an evolutionary pressure toward strategic information bottlenecks rather than attempting to process all input equally. Walther and Koch [28] showed that one such bottleneck occurs at the proto-object level, where coherent regions of the scene are selected for enhanced processing before full object recognition occurs. This allows the visual system to serialize complex scenes into manageable chunks while maintaining high coding efficiency.

2.1 Types of Visual Attention Mechanisms

Visual attention in biological systems operates through three primary mechanisms. Space-based attention operates on specific locations in the visual field, treating attention as a spotlight that enhances processing at selected spatial coordinates. Feature-based attention selectively enhances the processing of specific features (like color, orientation, or motion) across the entire visual field, regardless of spatial location. Object-based attention operates on perceptually grouped elements that form coherent objects, suggesting that attention selects entire object representations rather than just spatial locations or individual features [5, 25, 27].

2.2 Self-Attention

A key mechanism in modern computational attention is the self-attention layer. In its standard form [26], self-attention operates on a set of N input vectors, each of dimension d_{in} , linearly transforming them through learned weight matrices $\mathbf{W}_Q$ and $\mathbf{W}_K$ to obtain Query (Q) and Key (K) matrices:

$$S = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) \quad (1)$$

where d_k is the dimension of the key vectors. The attention scores S show how related the input elements are. S is further combined with a matrix V , which is also a linear transformation of the input, to

form the contextual representation $A = SV$, whose vectors contain the representation of each input considering the overall context.

2.3 Proto-Objects and Information Bottlenecks

Proto-objects are an intermediate representation between raw visual features and fully recognized objects [19, 28]. They are formed during pre-attentive processing and represent coherent regions of the visual field that share common visual properties. These structures serve as potential candidates for attention before full object recognition occurs [16], allowing the visual system to efficiently prioritize processing resources.

Information bottlenecks in visual processing serve to compress the high-dimensional visual input into more manageable representations while preserving task-relevant information [11, 23]. These bottlenecks can occur at various levels of processing, from early visual features to object recognition, and play a crucial role in managing the computational resources required for visual processing [29]. The formation of proto-objects itself represents a natural information bottleneck, as it reduces the complexity of the visual scene while maintaining behaviorally relevant information [28].

3 Related Work

Our approach builds upon and extends several lines of research in visual attention, reinforcement learning, and deep learning. Here we review the most relevant prior work that informs and contextualizes our contribution, focusing particularly on approaches that address the challenges of efficient visual processing and attention mechanisms.

3.1 Self-Interpretable Agents

Recent work by Tang et al. [22] explored the use of hard-attention mechanisms in reinforcement learning. Their approach divides the visual input into 7×7 patches and applies a modified self-attention mechanism to select only the top- k patches for processing (see Fig. 2). This creates an information bottleneck that forces the model to focus on the most relevant inputs while ignoring others. When applied to vision-based reinforcement learning tasks, this approach enabled agents to achieve better performance, interpretability and computational efficiency with far fewer parameters than conventional approaches. Our method differs mainly by using proto-objects instead of patches, reducing the number of tokens and their dimensionality while adding higher-level information to them.

3.2 Proto-Object Based Approaches

Orabona et al. [16] demonstrated the effectiveness of proto-objects as the basic units of visual attention. Their work showed that using proto-objects as an intermediate representation naturally corresponds to potential objects in the scene, allowing for efficient processing while maintaining meaningful visual features.

The idea of using higher-level information from blobs as input was further explored by Liang et al. [12] in their work on Blob-PROST (Pairwise Relative Offsets in Space and Time). Their approach demonstrated that visual features based on color blobs could achieve performance competitive with DQN [15] while using far fewer parameters. The core insight was representing game



Figure 2: The agent extracts patches from the current frame and selects the top- k ones (highlighted in white) by a hard attention mechanism. By visualizing the selected patches, it is possible to have a better idea of the learned strategy, as well as making it easier for debugging.

screens in terms of “blobs” - contiguous regions of same-color pixels that often correspond to game objects. This is a special case of our use of proto-objects, that are not restricted to same-color pixels. This representation was combined with position and relative offset features between blobs, capturing both spatial and temporal relationships. However, their representation consisted of more than 100 million binary features, while ours is based on compact representations of individual proto-objects that can be attended individually.

3.3 Object-Based Reinforcement Learning

While proto-object approaches focus on intermediate representations, some researchers have pushed further toward working with fully-fledged objects. Woof and Chen [30] experimented with representations of high-level objects (e.g. identifiable entities in a game screen) rather than end-to-end video game playing. However, they assume the objects are previously given, which requires access to the game engine or specific object detectors.

A more robust approach [1] demonstrated that operating directly on object-level representations can dramatically improve sample efficiency in reinforcement learning tasks. Their approach learns a succinct object representation from pixels without supervision by first oversegmenting images into primitive segments, modeling their dynamics, and then combining segments when they have similar dynamics. Their method leverages the insight that important events tend to occur when objects interact, using this to guide exploration toward states involving object contacts. While their approach achieves impressive results – learning up to 10,000 times

faster than conventional deep RL methods and even outpacing human learning in some cases – it requires significant computational overhead for object detection and tracking. This motivates the exploration of proto-object approaches, which provide a balance between processing efficiency and representational power.

4 Proto-Object Attentional Agent

Our work builds upon and connects several research directions in computer vision, deep learning, and evolutionary computation. We combine insights from biological models of visual attention, efficient neural architectures, and classical computer vision techniques to create a hybrid system that leverages the strengths of each approach. We apply hard-attention mechanisms to proto-objects rather than raw pixels or arbitrary patches. This approach implements an information bottleneck similar to what Koch et al. [11] observed in biological systems, while operating on the semantically meaningful proto-objects described by [16, 19]. By selecting only the most relevant proto-objects for processing, we create an information bottleneck at a more semantically meaningful level than previous approaches.

This combination is particularly well-suited for neuroevolution, as the discrete selection of top- k proto-objects and the transfer of coordinates to the controller create nondifferentiable operations that are challenging for gradient-based methods but natural for evolutionary approaches. Additionally, by forcing the model to be explicitly selective about which parts of the visual input it processes, we gain direct interpretability – we can visualize exactly which proto-objects the model considers important for its decisions, providing insights into its decision-making process that are often lacking in traditional deep learning approaches.

4.1 Implementation

Our method consists of 5 main stages: convolution, quantization, segmentation, attention and control, described next.

4.1.1 Convolution. The convolution stage aims to shift, rescale, filter and/or mix the original image channels, providing a pre-processed representation for the next stages. Particularly, in our experiments, we use a single convolutional layer with 3 1×1 filters. The choice of 3 filters is necessary for compatibility with the residual connection. It is possible to add more convolutional layers as long as they keep the same image size. In this case, adding a final layer with 3 filters is enough to bring down the number of dimensions to the same number of channels in the image. After that, we add the original image to the convolution output, forming a residual connection [9], whose function will become clear in the next stage.

4.1.2 Quantization. Quantization aims to reduce the amount of information to be processed in the next stages. In our experiments, we perform simple uniform quantization of the convolution output using 1 bit per channel (could be more for more complex tasks, and could even be evolved). As a result, we obtain an image with at most 8 distinct colors, each representing a different kind of segment. Note that, besides it being a simple fixed quantization, its combination with the convolutional layer before it results in an adaptive segmentation and quantization mechanism.

This stage has synergy with the convolutions: the shift, rescale, and mix of the original image channels may put them into different quantization bins. But since there are discontinuous jumps in the evolution fitness surface necessary to find an appropriate segmentation, which can take some time for the evolutionary algorithm to figure out, we use the residual connection in the previous stage as a means to kickstart evolution from a trivial segmentation over the original image colors. Thus, the purpose of convolution is to change the segmentation away from the trivial one (if necessary).

4.1.3 Segmentation. Segmentation aims to create the proto-objects, i.e. descriptors for regions of semantically similar pixels received from the previous stages. In this work, we apply image labeling by color-connected regions [7, 20]. From each extracted region, a set of attributes can be obtained, resulting in d_{in} features (regions 1 pixel wide or tall are treated as noise and ignored).

After thorough experimentation, we ended up with a set of $d_{in} = 11$ features, namely: quantized segment color (R, G, B), center of mass (X,Y), total area in pixels, bounding box width, bounding box height, bounding box area, aspect ratio and extent (bounding box area divided by region area). All of those can be easily and efficiently computed from the obtained regions, and will help the next stage to make more informed decisions. Orientation (correlation among pixel coordinates) could also be useful, but it added too much runtime overhead to our model and was left out. All values are normalized between -1 and 1, and aspect-ratio is also log-transformed such that 1 and -1 correspond to extreme ratios, while 0 means equal sides:

$$NormAspectRatio = 2 \frac{\log(aspectRatio)}{\log(\max(imageWidth, imageHeight))} - 1$$

4.1.4 Attention. The attention module aims to model relations among the proto-objects identified in the segmentation stage. The features for N proto-objects are fed to our model's attention layer as a set of N d_{in} -dimensional tokens, in the attention jargon [26]. The attention layer embeds these tokens into two d_q -dimensional vectors Q and K . There is, however, an additional touch in our implementation: we add a Parametric Rectified Linear Unit (PReLU) layer before and after the linear transformations. PReLU is a generalization of ReLU activation, where the slope of the negative part is adaptive ($PReLU(x) = \max(ax, x)$) for each layer or neuron (the latter in our case). For only 15 additional parameters, this enables our attention layer to model more complex relations, as a single PReLU neuron was shown to solve the XOR problem [17]. In our case, it enables the selection of midrange values (like gray colors) when a is negative (making the function nonmonotonic), which is not possible with pure linear layers. More layers of traditional self-attention could instead be used, but we opted for the simpler PReLU solution in this work to keep the number of parameters and runtime low.

Proceeding with the usual self-attention procedure, an attention matrix is computed by Eq. 1, and then an importance vector is obtained by row-wise summation. Instead of the usual mixing of tokens with a V matrix performed in traditional self-attention, we simply perform top- k proto-object selection over the resulting row-wise sum.

We remark that the attention computation has quadratic asymptotic time complexity on the number of tokens N . Drastically reducing the number of tokens by using proto-objects instead of patches makes our attention module much faster.

In our experiments, we went to the extreme and set $k = 1$ (coordinates of a single proto-object is passed to the controller, described next). This is possible because our attention module is more expressive than the original and proto-objects contain higher-level information, making a single well-selected proto-object enough for the controller to make its decisions (better selection means less work to the controller). It is also more biologically plausible, as we focus on a single visual item at time [4].

4.1.5 Control. Finally, the control stage selects an action to perform in the environment. In our implementation, a transfer function $f(n)$ is applied to each feature vector from the selected proto-objects and the results are concatenated and fed as input to an LSTM [10] controller, which is responsible for learning temporal associations and producing the control output.

In our case, $f(n)$ just returns the coordinates of the proto-object center of mass. More elaborate transfer functions could be used to feed the controller with more properties of each selected proto-object, but the center of mass was enough for our problems. This is possible because the joint evolution of the attention end control modules results in an implicit "agreement": by always selecting the same kind of proto-object (grass, track, etc...), there is no need for the controller to guess which is it. If attention focused on different kinds of proto-objects each time, it would not be possible to distinguish them solely by their coordinates, unless they consistently appeared on specific regions of the screen, being distinguishable by position (like the head-up display always at the bottom of the screen). They could also be distinguished by the controller if the attention module consistently puts the same kinds of proto-objects into the same ranking spots (e.g., grass first, track second), but this is an additional piece of complexity to be learned.

4.2 Overview

A summary of the differences between our hyperparameter choices and the previous work based on image patches [22] is shown in Table 1, as well as the resulting number of learnable parameters in each model, showing that our model is significantly (62%) smaller in total, due to its compact attention layer and smaller bottleneck with $k = 1$. The complete process can be seen in Fig. 3. Although this model is non-differentiable, it is learnable via derivative-free optimization methods such as CMA-ES [8].

5 Experiments and Results

In order to compare our approach to the patch-based one of [22], we test it on the same environments of [22]: CarRacing and Doom-TakeCover [3]. For both, we run CMA-ES with a population of 128 solutions for 1000 generations and evaluate models on 8 seeds every generation. Seeds are a function of generation and repetition numbers. We test models every 100 generations on 400 new seeds and extract means and variances to produce 95% confidence intervals. Statistical significance is obtained from two-sided Mann-Whitney U tests [13]. Note that the original experiments in [22] ran for 2000 generations with 16 seeds each and a population of 256 solutions,

Table 1: Comparison of hyperparameters and number of learnable parameters in patch-based model [22] and proto-object-based model (ours). The latter uses 62% less learnable parameters.

	Patches [22]	Proto-Objects (Ours)
Model Hyperparameters		
Attention Input Size (d_{in})	147	11
Embedding Size (d)	4	2
K	10	1
$f(n)$ Dimension	2	2
LSTM Input Size	20	2
# of LSTM Neurons	16	16
Number of Learnable Parameters		
Convolution	0	12
Attention	1184	63
LSTM	2432	1280
Output	51	51
Total	3667	1406

so they are not directly comparable. We halved each of those hyperparameters due to hardware limitations, and performed the original experiments again in this new setup for fair comparison.

Our experiments ran on the following hardware setup: AMD Ryzen 5950X CPU, 128GB DDR4 3200 RAM and Nvidia RTX 3090 GPU. Training was parallellized over 32 threads, limiting each evaluation to a single thread. The patch-based solution took advantage of the GPU, but our method was optimized for CPU, as multi-label connected-components analysis and labeling did not fit well with the GPU.

5.1 Car Racing

This is a top-down racing environment with randomly generated tracks (as seen in Figs. 1 and 2). It is visually simple enough to skip the convolution and quantization stages of our approach, but we perform them anyway in order to verify the generality of the method. The reward is -0.1 every frame, -100 for going far off-track (which also causes termination), and $+1000/N$ for every track tile visited, where N is the total number of tiles visited in the track (tiles are visible as slightly distinct shades of gray), and it is considered solved above 900 points. This incentivizes the controller to be fast and accurate. There are 3 continuous actions: steering (-1 is full left, $+1$ is full right), gas and braking. There is a version V2 of this environment available¹, but it uses Pygame², which is slow. We use V0, which is twice as fast by using OpenGL, and implement our own optimizations that adds a further 2x speedup. There are no significant differences between both versions except for compatibility with the new API [24] and better hardware and software compatibility.

Our method was more sample-efficient, with superior average score throughout the training, and achieved a significantly better

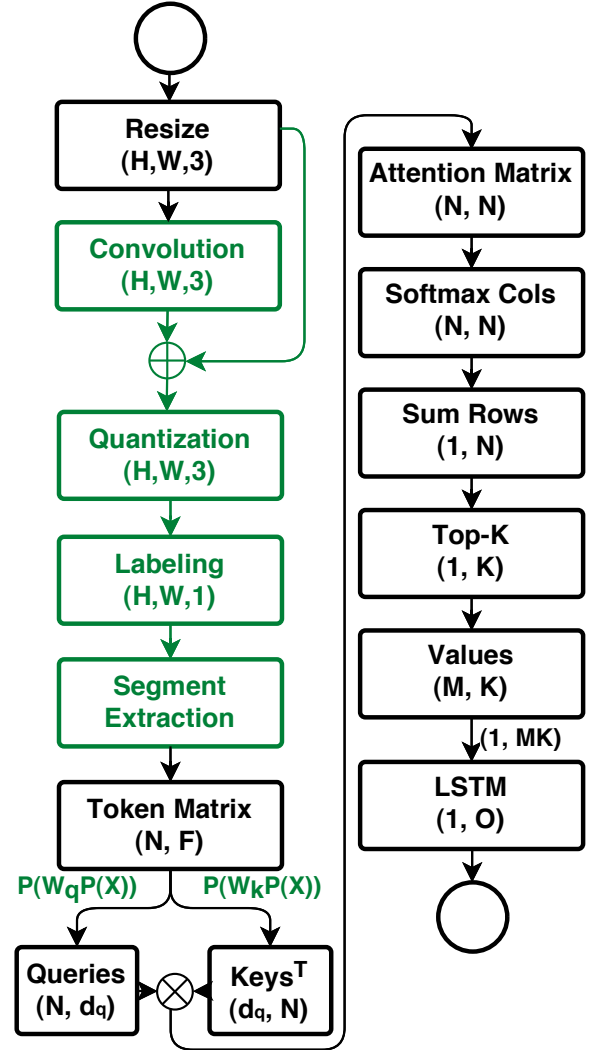


Figure 3: Flowchart of our complete process. In our experiments, $H = W = 96$, $F = 11$ (number of segment features), $d_q = 2$, $M = 2$ (we use only the x, y coordinates from each token), $k = 1$ and $O = 3$ (number of outputs for both environments). Any number of convolutional layers can be used, as long as they preserve image size and the last one has 3 filters to match the residual connection. We use one layer of 3 1×1 filters. Our quantization is set to 1 bit per channel (8 colors). P is the *PReLU* activation function. Green elements are new in relation to [22].

($p = 1.1e-22$) score of 910.39 after training (Fig. 4). Moreover, as Table 2 shows, it did so by using only 2% the number of tokens per frame as the patch-based solution and 62% less adjustable parameters. And despite running on CPU, it trained 2.7 times faster with respect to the patch-based method, which ran on GPU.

An interesting aspect of this experiment is to observe the evolution of segmentation and attention, as Fig 5 shows. The solution

¹https://gymnasium.farama.org/environments/box2d/car_racing/

²<https://www.pygame.org>

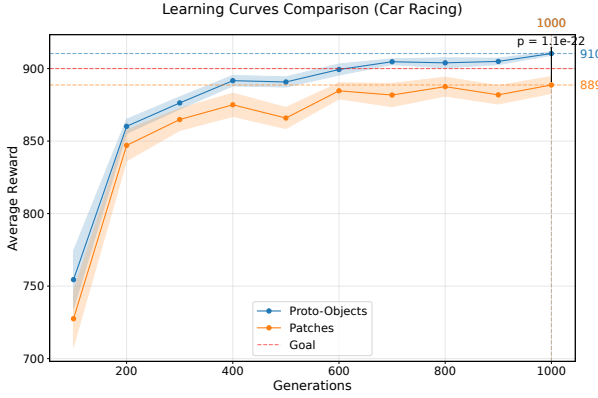


Figure 4: Learning curve comparison in the Car Racing environment over 400 test runs out of training sample tracks for each shown generation. Our proto-object method achieves significantly better results after 1000 generations in relation to the patch-based of [22]. Both peaked at 1000 generations.

Table 2: Comparison of results. The number of tokens per frame is fixed for the regular patch-based model, but variable for our proto-object-based model. Our method is currently optimized for CPU. In the number of tokens section, n refers to number of frames, while in scores it equates to number of runs.

	Patches [22]	Proto-Objects (Ours)
Number of Tokens and 95% CI of Best Solution (n=800)		
Car Racing	529	12.6 ± 0.26
Doom Take Cover	529	10.7 ± 0.73
Best Score and 95% CI After 1000 Iterations (n=400)		
Car Racing	888.69 ± 5.84	910.39 ± 1.28
Doom Take Cover	959.27 ± 58.85	930.68 ± 57.19 ($k = 1$) 1192.82 ± 75.26 ($k = 10$)
Training Time		
Car Racing	97h (GPU)	36.5h (CPU)
Doom Take Cover	85.5h (GPU)	33h ($k = 1$, CPU) 55h ($k = 10$, CPU)

starts with the trivial quantization over the task’s original colors, but since the track is dark, it gets merged with the black head-up display (HUD) at the bottom of the screen. Nevertheless, it already knows how to focus on the smaller grass region, as it usually points at the direction the car must turn. At 300 generations it learns to separate the track from the HUD, while at 800 generations it separates the car and the red corner markings from the track. While the car is useless (it is always at the same place and was even merged with the track in other experiments), the red markings can reinforce the correct turning side by “voting” (as queries) on their adjacent grass region. At 900 generations, it learns to segment the track tiles, but discards it in the final solution.

5.2 Doom Take Cover

This task is based on the game Doom, which is visually more complex and features much more colors than the previous task (see Fig. 6), making the convolution and quantization steps strictly necessary to prevent a huge number of segments. It takes place in a rectangular room. The agent is spawned along the wall, and monsters are constantly and randomly spawned along the opposite wall. They keep shooting fireballs at the agent, which must avoid them to survive. The agent gets 1 reward point for every tic alive and has 3 discrete actions: move left, right or stand still.

Learning curves are shown in Fig. 7. We observe that our approach had slightly lower sample-efficiency, needing more generations to match the patch-based model performance ($p = 0.414$). We also experimented with $d_q = 4$, $k = 10$ (same as the patch-based setup) and 3x3 convolutions (2671 parameters) and this solution had better sample-efficiency and achieved significantly ($p = 2.8e-5$) higher performance at a 1193 score in 55h of training. We hypothesize that the performance degradation was due to $k = 1$, meaning that the LSTM is under much harder work to keep up with multiple interest proto-objects on screen, or even missing some of them entirely, while also learning to discard the wall proto-objects that activate when there are no projectiles on screen.

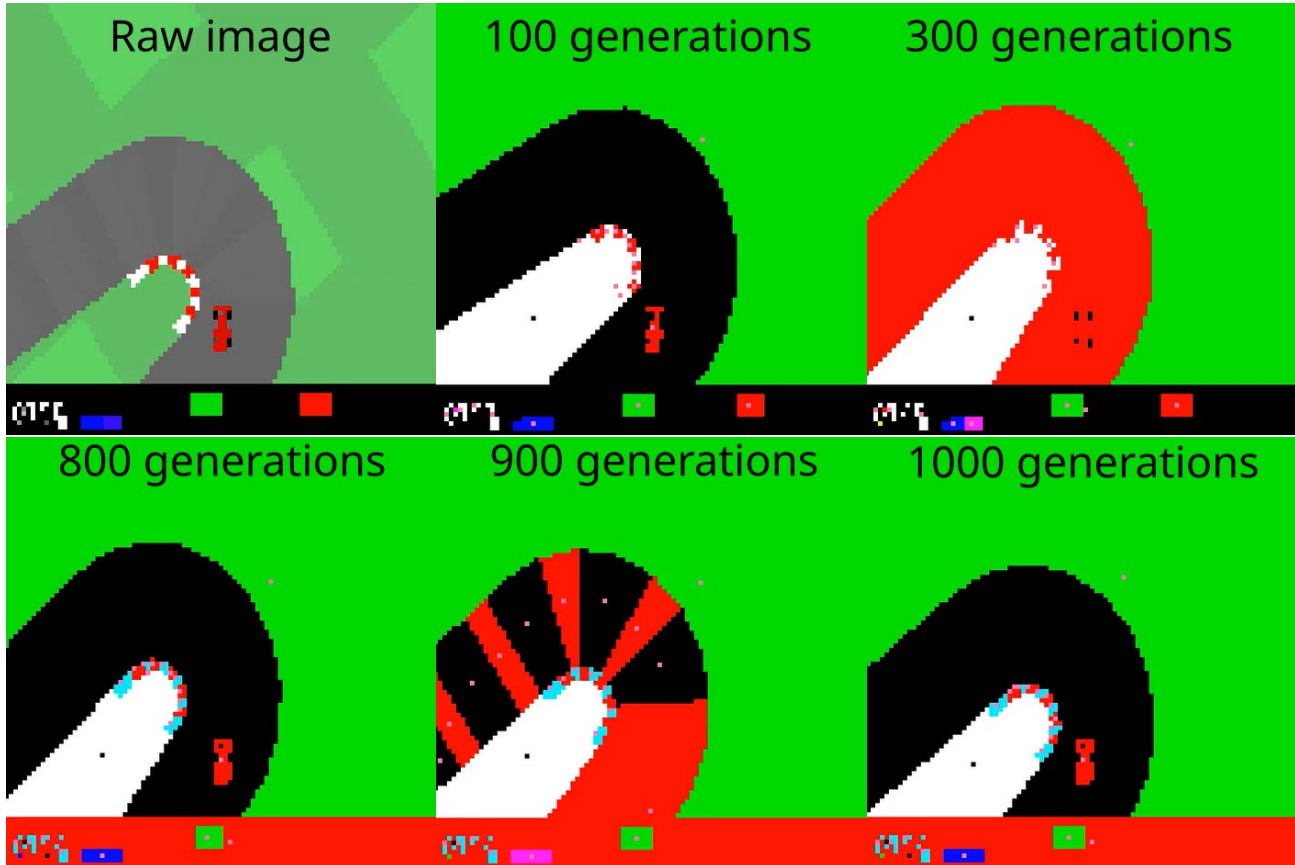
Table 2 also shows that the number of extracted proto-objects was low for this environment as well, demonstrating that our pre-processing steps are effective in reducing and uniformizing the visual complexity of different domains, while keeping necessary information for decision making. The training time was 2.6 times faster for $k = 1$ and 1.6 times faster for $k = 10$.

The key processing stages in the Doom environment are illustrated in Fig. 8: image resizing, 1x1 convolution, color quantization, and attention ($k = 1$). Remarkably, the evolved agent adopts a surprisingly minimalist strategy, ignoring seemingly critical elements like incoming fireballs. Instead, it focuses exclusively on the right-most monster on screen while executing a rhythmic left-to-right movement pattern. This strategy matches the performance of the patch-based model, despite the latter attending to both fireballs and walls. The equivalence to our simple approach suggests that the patch-based model’s LSTM might also be relying primarily on periodic movement and ignoring projectile coordinates. This strategy proves effective because the monsters’ projectiles target the agent’s current position – continuous movement therefore serves as a robust avoidance technique, regardless of the specific locations of incoming fire. However, our $k = 10$ agent seems more reactive to fireballs.

6 Conclusion and Future Works

We have presented a novel representation for bottleneck-attention-based agents in visual tasks that operates on proto-objects rather than raw pixels or image patches. By working with these pre-attentional primitive objects, obtained through classical computer vision methods, we achieved comparable or superior performance while dramatically reducing the number of tokens to attend and their dimensionality, as well as training time compared to previous solutions. The success of this hybrid approach highlights one of the key advantages of evolutionary methods for training such models: the freedom to combine differentiable and non-differentiable

Figure 5: Evolution of segmentation at 5 relevant points in time. The basic attention strategy (focus on the smaller grass region, highlighted in white with centroid in black; other centroids in pink) is learned early in the process, while segmentation keeps evolving until the end. Top-Left: Raw image after resizing and before segmentation. Top-Center (100 generations): The trivial segmentation from original colors is kept for the first couple hundred generations. Top-Right (300 generations): It learns to separate the track from the head-up display (HUD) below and differentiates some of the ABS sensors. Bottom-Left (800 generations): It swaps the track and HUD segmentations, making the car visible, differentiating the red corner markers and hiding the gyroscope indicators. It also differentiates the corner white markings from the score and speed indicator, and merges the ABS sensors again. Bottom-Center (900 generations): It differentiates the track segments. Bottom-Right (1000 generations): It gives up on the fine-grained segmentation of the track and goes back to the previous segmentation strategy, with very small and irrelevant changes in some pixels in the score.



components without being constrained by the requirements of gradient-based optimization. Nevertheless, developing a fully differentiable version of our solution remains an attractive direction for future work, as it could substantially improve sample efficiency.

Our experiments revealed that bottleneck attentional models are susceptible to local maxima during evolution. The dual-module architecture (attention and control) makes it challenging to discover new attention strategies once an approach becomes established, as the controller adapts specifically to the current attention mechanism. Any significant changes to the attention module risk disrupting this delicate balance. We hypothesize that CMA-ES may be too greedy for this architecture, and alternatives like differential evolution [21] might be more suitable by allowing multiple attention strategies to evolve in parallel.

We demonstrated that by enhancing the attention layer, sending coordinates of a single proto-object to the controller is sufficient to produce effective policies. This works because the LSTM can maintain and update an internal state representation across frames, deciding what information to preserve or discard. This approach aligns well with biological eye movements, where focus necessarily shifts between individual locations or objects [4]. However, this simplified information flow comes at the cost of longer learning times when there are multiple relevant entities on screen, as the attention module must attend to them all and the controller must develop sophisticated memory management strategies. One potential solution is to decouple memory and control, possibly by implementing attention mechanisms over recently attended coordinates to generate fixed-size embeddings [18] for the controller. This



Figure 6: The Doom Take Cover environment. The agent has to avoid the fireballs shot by the enemies in the back of the room by moving left and right. The higher number of colors requires our convolution and quantization components to prevent a huge number of segments.

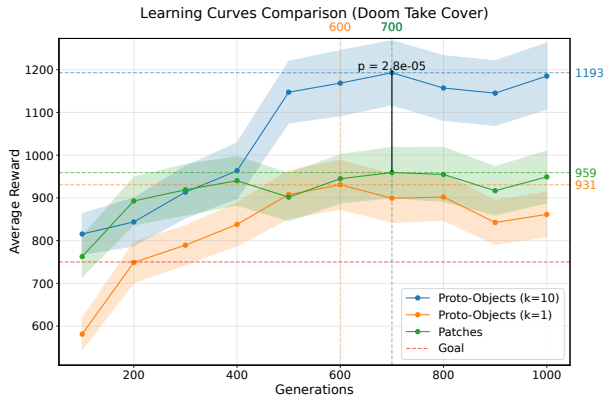


Figure 7: Learning curve comparison in the Doom Take Cover environment over 400 test runs out of training sample seeds for each shown generation. Our method with $k = 1$ showed lower sample-efficiency but achieved similar performance after 1000 generations in relation to [22], which achieved peak performance at 700 generations ($p = 0.414$ between best solutions). However, with $k = 10$ (same as the patch-based setup) and further adjustments detailed in the text, our solution had better sample-efficiency and achieved significantly higher ($p = 2.8e-5$) performance (1193 score at 700 generations).

could be further extended to include adaptive storage and retrieval from vector databases.

Several promising directions for future research emerge from this work. Feedback signals from the controller could modulate attention, enabling active top-down strategies. This would require enriching the information flow from the attention module to help the controller interpret incoming signals. Multi-head attention represents another natural extension. The approach could potentially

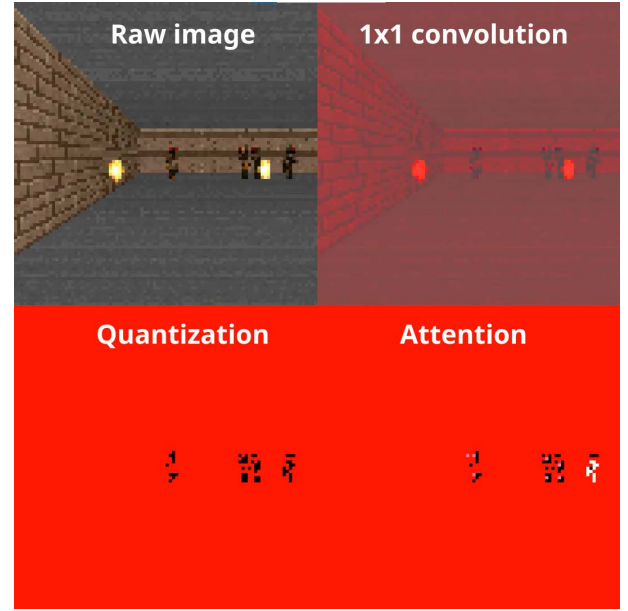


Figure 8: Processing stages in the Doom Take Cover environment. Top-left: raw image resized to 96x96. Top-right: 1x1 convolution + residual. Bottom-left: quantization. Bottom-right: attention. Note that this solution completely ignores the fireballs since the quantization stage.

scale to full object recognition by incorporating additional convolutional layers and processing depth (when available) and motion information. Self-attention mechanisms might enable autonomous grouping of regions into higher-level entities, while cross-attention could facilitate object tracking across frames.

Finally, a crucial next step is validating our approach on real-world images and determining whether increased complexity in the convolution and quantization stages is necessary, or if patch-based approaches prove more effective in such scenarios. Success in this domain could lead to more efficient robot and self-driving car systems, reducing computational requirements while enabling more sophisticated intelligence per processing unit.

Acknowledgments

The authors would like to acknowledge FAPERGS (Notice 10/2021 – ARD/ARC) for the financial support. This research was partially supported by Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq (Grant 445238/2024-0). This study was also supported by the Federal Institute of Education, Science and Technology of Rio Grande do Sul (IFRS).

References

- [1] William Agnew and Pedro Domingos. 2018. Unsupervised Object-Level Deep Reinforcement Learning. Deep Reinforcement Learning Workshop (NIPS 2018).
- [2] Ali Borji and Laurent Itti. 2012. State-of-the-art in visual attention modeling. *IEEE transactions on pattern analysis and machine intelligence* 35, 1 (2012), 185–207.
- [3] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. OpenAI Gym. arXiv:1606.01540 [cs.LG]

- [4] Marisa Carrasco. 2011. Visual attention: The past 25 years. *Vision Research* 51, 13 (2011), 1484–1525. doi:10.1016/j.visres.2011.04.012 Vision Research 50th Anniversary Issue: Part 2.
- [5] Zhe Chen. 2012. Object-based attention: A tutorial review. *Attention, Perception, & Psychophysics* 74 (2012), 784 – 802. doi:10.3758/s13414-012-0322-z
- [6] Leif H. Finkel and Paul Sajda. 1992. Proto-objects: an intermediate-level visual representation, In *Optical Society of America Annual Meeting. Optical Society of America Annual Meeting* -, -, FO1. doi:10.1364/OAM.1992.FO1
- [7] Christophe Fiorio and Jens Gustedt. 1996. Two linear time union-find strategies for image processing. *Theoretical Computer Science* 154, 2 (1996), 165–181.
- [8] Nikolaus Hansen. 2006. *The CMA Evolution Strategy: A Comparing Review*. Springer Berlin Heidelberg, Berlin, Heidelberg, 75–102. doi:10.1007/3-540-32494-1_4
- [9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, Los Alamitos, CA, USA, 770–778. doi:10.1109/CVPR.2016.90
- [10] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.
- [11] Kristin Koch, Judith McLean, Ronen Segev, Michael A. Freed, Michael J. Berry, Vijay Balasubramanian, and Peter Sterling. 2006. How Much the Eye Tells the Brain. *Current Biology* 16, 14 (2006), 1428–1434. doi:10.1016/j.cub.2006.05.056
- [12] Yitao Liang, Marlos C. Machado, Erik Talvitie, and Michael Bowling. 2016. State of the Art Control of Atari Games Using Shallow Reinforcement Learning. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems (Singapore, Singapore) (AAMAS '16)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 485–493.
- [13] H. B. Mann and D. R. Whitney. 1947. On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other. *The Annals of Mathematical Statistics* 18, 1 (1947), 50 – 60. doi:10.1214/aoms/1177730491
- [14] Volodymyr Mnih, Nicolas Heess, Alex Graves, and Koray Kavukcuoglu. 2014. Recurrent models of visual attention. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2 (Montreal, Canada) (NIPS'14)*. MIT Press, Cambridge, MA, USA, 2204–2212.
- [15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing Atari with Deep Reinforcement Learning. arXiv:1312.5602 [cs.LG]
- [16] Francesco Orabona, Giorgio Metta, and Giulio Sandini. 2007. A Proto-object Based Visual Attention Model. In *Attention in Cognitive Systems. Theories and Systems from an Interdisciplinary Viewpoint*, Lucas Paletta and Erich Rome (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 198–215.
- [17] Rafael C. Pinto and Anderson R. Tavares. 2024. PreLU: Yet Another Single-Layer Solution to the XOR Problem. arXiv:2409.10821 [cs.NE]
- [18] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. arXiv:1908.10084 [cs.CL]
- [19] Ronald A Rensink. 2000. The dynamic representation of scenes. *Visual cognition* 7, 1-3 (2000), 17–42.
- [20] William Silversmith. 2025. *connected-components-3d: Connected Components on Discrete and Continuous Multilabel 3D & 2D Images*. GitHub. <https://github.com/seung-lab/connected-components-3d>
- [21] Rainer Storn and Kenneth V. Price. 1997. Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. *Journal of Global Optimization* 11 (1997), 341–359. <https://api.semanticscholar.org/CorpusID:5297867>
- [22] Yujin Tang, Duong Nguyen, and David Ha. 2020. Neuroevolution of self-interpretable agents. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference (Cancún, Mexico) (GECCO '20)*. Association for Computing Machinery, New York, NY, USA, 414–424. doi:10.1145/3377930.3389847
- [23] Naftali Tishby, Fernando C. Pereira, and William Bialek. 2000. The information bottleneck method. arXiv:physics/0004057 [physics.data-an]
- [24] Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U. Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, Rodrigo Perez-Vicente, Andrea Pierré, Sander Schulhoff, Jun Jet Tai, Hannah Tan, and Omar G. Younis. 2024. Gymnasium: A Standard Interface for Reinforcement Learning Environments. arXiv:2407.17032 [cs.LG]
- [25] Stefan Treue and Julio C. Martínez Trujillo. 1999. Feature-based attention influences motion processing gain in macaque visual cortex. *Nature* 399 (1999), 575–579. <https://api.semanticscholar.org/CorpusID:4424973>
- [26] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems (Long Beach, California, USA) (NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 6000–6010.
- [27] Shaun P. Vecera and Martha J. Farah. 1994. Does visual attention select objects or locations? *Journal of experimental psychology. General* 123 2 (1994), 146–60. doi:10.1037//0096-3445.123.2.146
- [28] Dirk Walther and Christof Koch. 2006. Modeling attention to salient proto-objects. *Neural Networks* 19, 9 (2006), 1395–1407. doi:10.1016/j.neunet.2006.10.001 Brain and Attention.
- [29] Jeremy M Wolfe. 1994. Guided search 2.0 a revised model of visual search. *Psychonomic bulletin & review* 1 (1994), 202–238.
- [30] William Woof and Ke Chen. 2018. Learning to Play General Video-Games via an Object Embedding Network. In *2018 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, Maastricht, Netherlands, 1–8. doi:10.1109/CIG.2018.8490438