BIELEFELD UNIVERSITY
FACULTY OF TECHNOLOGY

Generalized Attention-Weighted Reinforcement Learning

.

Integrating Neuroscience Theory with Machine Learning Practice

Author:
Lennart Bramlage

Supervision:
Aurelio Cortese, Ph.D.
Prof. Barbara Hammer, Ph.D.

A thesis submitted in partial fulfilment of the requirements for the degree Master of Science

December 16, 2020

Abstract

Lennart Bramlage

Generalized Attention-Weighted Reinforcement Learning
Integrating Neuroscience Theory with Machine Learning Practice

Reinforcement Learning (RL) is a simple mechanism that allows biological organisms to learn rewarding behavior through trial-and-error. The framework has a long and successful history in machine learning applications as well. However, in most cases, it requires billions of interactions to perform adequately, begging the question of where artificial learners fall short. A popular theory from neuroscience posits that higher cognitive functions, such as memory, metacognition, and attention, join forces to create lower-dimensional task-state representations for much more efficient learning. Computational theoretical models of these functions are commonplace, but they are seldomly proven to work in an actual learning paradigm. Conversely, machine learning research rarely follows neuroscientific theory and instead progresses along the route of computational improvement. This thesis aims to present an integrated perspective of the two disciplines by translating a theoretical model of feature-based attentional learning, *Attention-Weighted Reinforcement Learning* (AWRL), into a functional neural network architecture. The viability, benefits, and shortcomings of this model are evaluated in a suite of canonical deep-RL benchmark tasks from the Atari framework. The proposed model uses the multi-head dot-product attention mechanism (i.e., self-attention) popularized by natural language processing applications. Self-attention introduces the exciting possibility of not only selecting task-relevant features but relating them to create compound representations of current observations, giving rise to a novel, generalized AWRL model. The first in a series of three experiments validates this extension by indicating that relating features in task-state representations is essential to solving the presented challenges in an RL paradigm. The second experiment shows that g-AWRL compares favorably in environments with increased levels of task-irrelevant features in observation space, the major issue in biological intelligence and sensory processing. Finally, the last experiment investigates how the model implements bottom-up (feature-driven) and top-down (task-driven) attention and their respective contributions to overall learning performance.

Acknowledgements

I want to thank my supervisor, Aurelio Cortese, for his enthusiasm, trust, and continued support throughout this thesis project and the preceding internship at his lab that inspired it. And Prof. Barbara Hammer, for her encouragement and spontaneity to take over the faculty-side of the supervision on such short notice.

My utmost thanks go out to the people I met during my time at ATR Institute, specifically Jessica Taylor, for her friendship and the many and much-needed after-work drinks, and Mieko Hirata, for her dedication, care, and incredible patience.

Additionally, I would like to thank the people in my life that kept me reasonably sane this year of a global pandemic. My great roommates and the beautiful people they introduced me to for many hours of social proximity and my oldest friends for their tireless capacity to deconstruct my fears and worries.

This thesis is dedicated to my mother and grandmother, who enabled my studies and so much more.

Contents

List of Figures

Chapter 1

Introduction

Biological intelligence is famously successful at adapting to and learning about novel situations that prompt decisions. Apart from inductive, physiological biases brought about by evolution, one central and well-understood mechanism is responsible for all manner of intelligent behavior: conditioning, or in computational terms, *Reinforcement Learning* (RL) (Sutton and Barto, 2018). The general idea is that rewarding experiences enforce behavior correlated with a particular set of observations (across all senses). Reinforcement can be negative and positive, discouraging, or promoting previously engaged in action, respectively. Even in the absence of all knowledge, guidance, or instruction, this general principle can implement a trial-and-error learning procedure that will lead to goal-seeking and harm-avoiding decision-making. The RL mechanism is not only biologically plausible (Niv, 2009) but also a practical algorithm for *Machine Learning* (ML) applications. Many ML methods rely on the existence of labeled data, i.e., data that provides both inputs and the corresponding outputs, such that the employed algorithm can learn a functional mapping. Instead, RL relies on an abstract objective function that commonly represents a maximization target of a reward signal in an environment (Bellman, 1966). The algorithm maximizes the target by exploring the environment and then exploiting its gathered knowledge of observation-action-reward relations by choosing actions that maximize the reward signal. While exploration-exploitation trade-off is one of the more challenging problems to solve in practice, the general approach allows the adoption of intelligent behavior even in complex environments with massive action- and observation-spaces. Artificial RL applications have successfully solved a plethora of board-, and video games (Mnih et al., 2013), been applied to numerous complex robotics tasks (Gu et al., 2017; Zhang et al., 2017), and even beat the world champion in the infinitely diverse game GO (Silver et al., 2016). However, a diminishing factor of all these success stories is the immense amount of trial-and-error interactions necessary to achieve even the smallest sliver of intelligent behavior in each of the tasks. Usually, the applied algorithm requires billions of interactions and, in some cases, equally as many predictions using a learned model of the environment to generate useful policies. If these RL implementations indeed are formalized versions of the primary learning principle in biological intelligence, then where do they fall short?

While ML research continually introduces novel approaches to making RL algo-

rithms faster, more robust, and general, the field of neuroscience produces psychological and computational theories that attempt to explain the efficiency of biological learning. Thus, a number of attractive candidate principles have emerged as possible and partial explanations of faster learning: Memory, for example, endows intelligent agents with the ability to recall and extrapolate from prior experience. This function enables both replay (continued learning after the fact) and reliving (ad-hoc decision making based on similar previous situations and choices) (Lewis et al., 2018; Lengyel and Dayan, 2009). Abstraction and rule learning relies on memory mechanisms as well (Ho et al., 2019; Konidaris, 2019), but instead of replaying or reliving prior experiences, this function breaks them down into logical parts to be reapplied to novel situations that differ only at a surface level (Santoro et al., 2016). Metacognition, or thinking about thinking, is a central element of consciousness and allows humans specifically to examine their own beliefs, choices, and emotions. Instead of relying on our first impression of a situation, we are able to frame them in the context of our experience, forgoing mental biases and choosing more enlightened courses of action (Pouget et al., 2016).

Notably, all these higher cognitive functions combine to achieve the singular goal of optimally representing our current perceptions, such that optimal actions can be chosen. Every natural environment (and every biological brain) is rich with irrelevant stimuli, intuitively commanding a being with limited time and energy to pick and choose the most relevant information and basing decision on a vastly reduced representation of relevant features (Cortese et al., 2019). This is the role of attention. From a sensory to an abstract level of high conceptual meaning, attention allows us to identify whatever is relevant to our choices. It serves as a gatekeeper of memory (Theeuwes et al., 2009), thus guiding the rules and concepts we deduce from experience. Trained attention too highlights which of our thoughts we examine closest, perhaps due to prior suggestions or repeated errors.

It seems to be no coincidence then, that attention has captivated the world of ML research in a renaissance initiated by applications in natural language processing. However, only recently has the concept entered the sphere of RL research and deep-RL applications in learning for decision-making. With sporadic interactions between the fields of ML and neuroscience research, definitions have remained vague. Both fields know multitudinous areas of application for attention mechanisms but evade a general nomenclature. Perhaps the most generally agreed upon definition of attention is the direction of computational resources towards relevant stimulus dimensions (Mackintosh, 1975) (spoken word/sounds, visual features, spatial location, and many more.) To reduce the sample inefficiency of deep-RL approaches, then, one should first examine attention as it both engages at the earliest sensory levels, and is a prerequisite of more complex cognitive functions.

1.1 Problem Statement

This work aims to exemplify an integrated perspective on intelligence by merging neuroscientific theory with ML application. Neuroscience all too often introduces computational models of complex processes in the human and animal brain with fitting of data from behavioural experiments as the only validation. There are usu-

ally only a few free parameters to underwrite the explanatory strength of these models. Conversely, RL research rarely takes cues from works on biological intelligence, despite obvious influences and relationships. *Most state-of-the-art approaches improve on computational issues*, which, while indispensable, forgoes the opportunity to take inspiration from plausible processes that facilitate intelligence in its only existing examples: biological organisms. As a result, what ML researchers deem attention may deviate vastly from a behavioral scientist's perspective.

Here, we take direct inspiration from neuroscience by translating a theoretical, computational model into an applied algorithm, using state-of-the-art ML methods. We examine various definitions of attention, which allows us to identify formal definitions of biological concepts and investigate their validity by having them learn and behave organically in a suite of canonical deep-RL benchmark tasks. This is a stark contrast to the above mentioned data-oriented fitting process of such theoretical models. We will then go on to examine whether these attention mechanisms truly harbor algorithmic benefits for convergence speed or generalization in deep-RL. Additionally, we will shine a light on the process of learning and point out similarities and differences towards theoretical models of attention acquisition.

In summary, we translate neuroscientific theory into applied ML, providing definitions of computational models geared towards applications in neural networks, specifically of attention. We then examine whether these models increase actual performance in deep-RL applications and whether their emergent behavior matches their hypothesized behavior.

1.2 Thesis Structure

In the following section, we will summarize prior work on attention in both disciplines, computational neuroscience and machine learning. We will draw out the most significant theoretical models and principles and the most prolific algorithmic approaches, comparing them in terms of their logical overlap and differences. Afterward, we will describe the general methods employed in the experiments, from artificial neural networks to the specific RL algorithms and their variants. The methods section is followed by our practical translation of the popular attention-weighted RL framework for application in deep-RL tasks. This section describes how we implement a generalized pair of forward-step and learning rule inspired by neuroscience with the established self-attention mechanism. In the succeeding experiments section, we will showcase the framework's versatile nature and its ML interpretation by applying it as the central mechanism in a range of attention-based neural network architectures. The thesis concludes with a discussion of the applied quality of the AWRL framework, the benefits of neuroscience-inspired ML research, and an outlook on future work.

Chapter 2

Background and Prior Work

2.1 Attention in Neuroscience

As suggested in the introductory paragraphs, even in neuroscience attention is a beast with many heads, rarely well-defined and ubiquitous at all levels of the cortical hierarchy. Here, we will give a short overview of attention research in natural intelligence, starting with sensory attention and moving on to more abstract definitions. We will then conclude this section with an investigation of the origins of attention in biological organisms, i.e., how is it acquired or learned? The final subsection then provides a preview of the interactions between RL and attention that we later expand upon in the chapter on generalized attention-weighted RL.

2.1.1 Sensory Attention

Perhaps the most intuitive concept of attention is that of focusing one's senses on a particular observable (hearing, taste, touch, smell) subset of stimulus dimensions. The necessity for this is a given, as biological organisms need to solve several problems like foraging for food, procreating, and avoiding danger, all on a budget of limited time and energy in noise-ridden environments. Even the most barren natural environments exert an abundance of perceivable signals, forcing animals to discard the majority of information that would excite their senses. For example, recent theories on human vision propose that as little as 0.00004% of the visually available data from a single glance goes on to influence our decision-making (Zhaoping, 2019) - the rest is discarded as early as the primary visual cortex (V1).

Visual attention makes up a large slice of attention research, in part because it is relatively easy to observe with non-invasive methods. Rapid movements of the eyes between fixation points, called saccades, are a facilitator of this so-called "overt" attention that is easily recorded by light-weight camera systems. The fixation point of an animal's gaze is a fair predictor of what they attend to. Research into the visual system suggests that peripheral vision serves as little more than a guidance mechanism for focal vision (Hooge and Erkelens, 1999), with peripheral feature detection prompting saccades towards visually salient stimuli. Examples of such

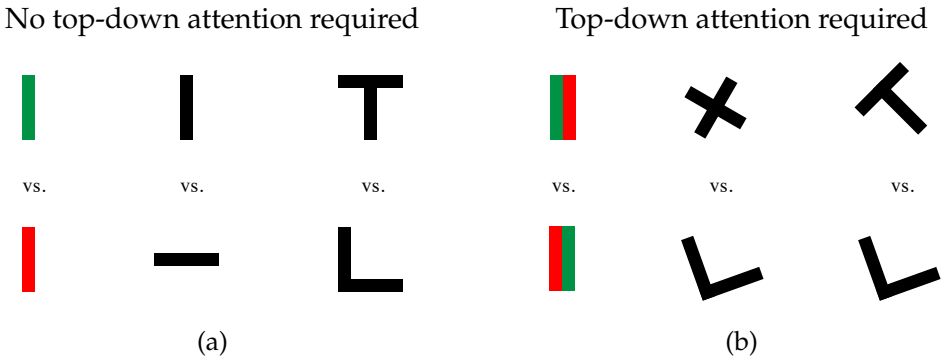


Figure 2.1.1: Common stimuli in visual psychophysics: top-down attention is not required to discern differences between visual features in (a) (color-contrast, simple orientation, and shapes). However, the stimuli in (b) demand top-down processes for accurate discrimination inside a field of similar distractors (Tsuchiya and Koch, 2015)

.

stimuli include motion, faces or shapes of predators or social peers, and many less complex stimuli such as specific color contrasts, spatial frequencies, or orientations (see Figure 2.1.1 a) (Li, 2002). These fixation processes can work in a "bottom-up" manner, that is, without the conscious decision to focus on a certain point in visual space (Kastner and Ungerleider, 2000). Saliency thus may be understood as an evolutionary attention mechanism, scoring detectable features by their statistical likelihood to impact one of our primary motivators (avoiding danger, procreation, and sustenance). Where bottom-up attention accounts for a reactionary or survivalist selection of relevant stimuli, top-down attention constitutes a selection mechanism guided by higher-order task settings (Connor et al., 2004). As such, stimuli that would otherwise not exhibit a "pop-out" effect can be examined with a higher focus of concentration (see Figure 2.1.1 b). Both routes of attention may work in concert to form a shared saliency map, i.e., which spatial areas of the visual input are most noteworthy (Duncan et al., 1997). However, they may also clash, as bottom-up saliency can overrule top-down attention and draw the eyes towards distracting stimuli or invoke visual illusions.

With this brief overview of overt visuospatial attention, we will now turn towards covert attention. Eye movements are not the only mechanism by which we focus on specific areas in the visual space. Instead, it is possible to heighten awareness in one particular peripheral locus without an overt shift of the fovea. Thus the overall input remains the same, but computational resources of further processing can be directed towards different areas either at will or through distractor stimuli. As suggested above, this ability might serve primarily as a guiding mechanism for foveal attention (Sheliga et al., 1994); in fact, various studies have found an increase in attentional processing directed at extrafoveal areas that were subsequently targeted as fixation points by saccadic movements (Rolfs and Carrasco, 2012).

Yet another dimension of attention is the dichotomy of spatial and feature-based attention (Posner et al., 1980; Carrasco, 2011). So far, we have examined selective attention in space. Conversely, intelligent organisms may attend to a selection of

task-relevant features in a covert manner. The major difference between these two modes is physiological, in that a subject that receives a cue to attend to a particular feature will do so globally, i.e., across the entire visual field. This implies that neurons tuned to represent the feature will be modulated to fire at higher rates if they detect the attended feature, regardless of its position (Saenz et al., 2002), which is much more plausible when top-down processes control them. That is, non-salient feature combinations may suddenly "pop out" when the right top-down task is engaging neurons that represent those features. This perspective follows from a popular model of physiological attention, the feature-similarity gain model. It prescribes that attention modulates the response strength of neurons representing specific features or spatial areas in, e.g., the visual field (Treue and Martínez Trujillo, 1999). Global feature attention, again, facilitates visual search in that it can guide focal vision. However, if the task is constructed in a way as to engage feature-attention, spatial attention will be superseded by feature attention. In simpler terms, global feature attention guides focal vision and selects fixation points in space, but focal vision will only confirm that the chosen location contains the sought-after feature instead of processing the full stimulus.

In summary, sensory attention is best explained as a combination of bottom-up (stimulus-driven) and top-down (task-driven) selection of stimulus dimensions or features. This selection can be overt or covert (specifically in vision), meaning that either a sensory organ is directed at a specific region or that neurons representing this region are modulated to respond at higher frequencies. Additionally, attention can be globally feature-oriented through the same mechanism, i.e. through the modulation of neurons representing said features. Lastly, attention can be applied to highlight one sensory modality over others, incurring a certain computational cost and adding yet another definition to the list.

2.1.2 Attention and Memory

A critical, enabling factor of intelligent behavior is the ability to integrate various sensory measurements over time, such as in reading a sentence, listening to speech, or watching a movie. At the smallest timescale, this skill is implemented by working memory, which, in its essence, constitutes the sustained activity of initial sensory excitement (Courtney et al., 1997; Cohen et al., 1994). However, due to its limited capacity, working memory needs a gate-keeping mechanism to selectively represent those stimuli that will affect future decisions (Conway and Engle, 1996). In this way, top-down- or executive attention determines which perceivable features enter working memory to inform current goal-directed behavior or compel further processing in long-term memory systems. Again, executive attention competes with bottom-up saliency for space in working memory, making the system susceptible to distractor stimuli (Knudsen, 2007). These interactions culminate in one three-way model of control, attention, and memory, whereby control directs attention to sample the current observation space across all sensory modalities. Those feature subsets that earn access to working memory will then influence control, promoting shifts in attention (Ahissar and Hochstein, 2000). It is easy to identify a simple interaction here that we will come across multiple times: attention limits perception, perception directs attention. In these ways, attention is one

of the primary mechanisms that bias memory formation (encoding), as exemplified by various studies. For example, Gardiner and Parkin asked subjects to memorize a list of words while performing a secondary tone-monitoring task. Subjects that needed to complete the secondary task performed significantly worse when recalling the list of words, suggesting a shared budget of attention across modalities.

However, attention is not only curating memory contents but likely acts as a selective retrieval mechanism. The same experiment design mentioned above can highlight that retrieval of memory is equally impaired by a secondary task assignment (during recall of the list of words, instead of during learning). While this is no definitive proof that goal-directed attention draws from the same pool of cognitive resources as memory retrieval, there must be a shared bottleneck, with attention or some alternative selecting process being plausible candidates.

In conclusion, (working) memory and attention share significant theoretical overlap and numerous integrative models of their interactions. The case for attention as a precursor or gatekeeper of short-, and consequentially mid- and long-term memory is well-studied and supported by behavioral and physiological findings (Aly and Turk-Browne, 2017; Córdova et al., 2019). While researched to a lesser extent, the role of attention in memory retrieval is somewhat plausible and backed by initial behavioral studies.

2.1.3 Acquisition and Learning of Attention

The ways in which intelligent organisms acquire efficient attention mechanisms are as multi-faceted as the mechanisms themselves. As alluded to in the previous sections, evolutionary development is a plausible explanation for various bottom-up- and saliency-related processes. Still, most animals attain flexible attention on much smaller timescales. Humans are especially adept at modulating their attention within small timespans, attending relevant stimulus dimensions after only a few seconds of engagement with a novel task.

Most computational models that aim to explain these processes emphasize the relevance of value signals during learning (Doya, 2007). Behavioral studies have repeatedly indicated bidirectional interactions between value-based learning and attention (Anderson et al., 2011; Chelazzi et al., 2013; Leong et al., 2017). Attentional focus reportedly has a massive impact on value computation and, as a result, reward prediction errors, a signal that indicates whether or not currently held beliefs about the value of a stimulus are correct. While other paradigms may be possible, RL has been the most well-established and physiologically plausible candidate process to implement attentional learning. The two earliest computational models merging value-oriented RL and attention are the Mackintosh- and the Pearce-Hall models. The Mackintosh model proposes that attention is directed at stimulus-dimensions most predictive of reward (Mackintosh, 1975), thereby underwriting the view of attention as an information-seeking mechanism (Gottlieb et al., 2014). Pearce and Hall put forward a different perspective. Their model states that attention selects those stimulus dimensions that the agent is most uncertain about, i.e., those that caused more significant value prediction errors in the past. This suggested mechanism finds support in several studies that illustrate how reward

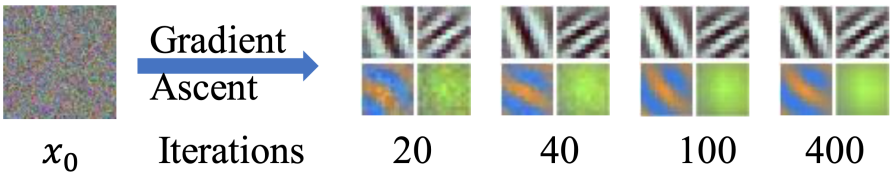
prediction errors indeed enhance subjects' ability to learn about corresponding stimuli. Both models have promoted large bodies of research, sparking a variety of attempts to integrate them into a singular framework. Dayan et al. suggests that intelligent organisms employ both models at different phases of learning and acting. During learning, agents pay attention to stimuli and stimulus-dimensions with high uncertainty, i.e., sizeable preceding prediction errors (Pearce-Hall model). At choice of action, attention highlights those stimulus dimensions that hold the most predictive power about value outcomes (Mackintosh). Theoretically, this dichotomous perspective on attention would implement an optimal exploration-exploitation strategy, provided that learning and choice episodes were sufficiently distinct, and the attention regimes readily deployed. In practice, it means that learning about value shapes attention implicitly, regardless of the applied model.

2.2 Attention in Machine Learning

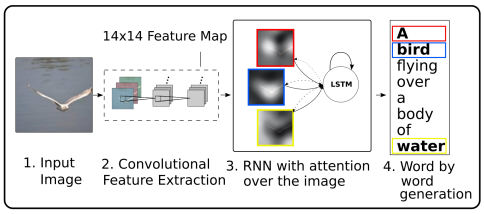
Like neuroscience, machine learning research commands a storied research history into attention mechanisms, both explicit and implicit. Much of this research focuses on applications of artificial neural networks. Similar to biological brains, these models require additional measures to become more flexible in tasks with changing goals, variably sized inputs, and shifting data. This research's primary purpose is to improve model performance and convergence rates, i.e., learning speed. Secondarily, but no less valuable is the ability to visualize or otherwise capture what the neural network is attending to when making judgments. In this subsection, we will investigate the most impactful applications in image and natural language processing, as well as decision-making.

2.2.1 Image Processing

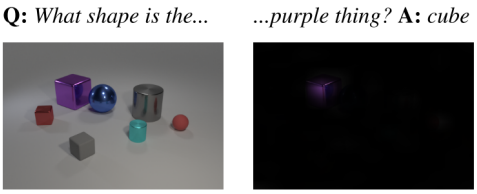
Attention for image processing constitutes a relatively faithful translation of biological concepts to machine learning. Before examining some explicit models, it is essential to note that attention might be considered a passive phenomenon from a naïve, computational perspective. Neural networks and similar gradient-based learning paradigms implicitly comply with the directive to identify the most relevant data segments to optimize an objective function in line with the Mackintosh model of biological attention. For example, in classification paradigms, one can generate an image that would effectively excite a specific neuron in the image classifier network, using the Jacobian (Nguyen et al., 2016). In other words, one would optimize the pixel values of an input image through gradient methods, where the objective function is the activation of some neuron or filter (for convolutional neural networks) (see Figure 2.2.1 a). The result will be the image that the particular optimized-for neuron would get excited by most, which will be a visual feature that speaks for a specific image class, provided the classifier is fully trained. This process is called "activation maximization" - a synthesis of preferred stimuli. Through minimizing prediction errors for the class of a particular image, i.e., the objective function, the network has implicitly changed to prefer (or attend to) those features



(a) Activation Maximization: An image of random pixels is optimized to activate four separate convolutional filters via gradient ascent. The results are images that the filters would respond to most. (Qin et al., 2018)



(b) Show, Attend, and Tell network: The feature maps of sequential glimpses, corresponding to extracted patches from a larger image, are integrated to generate a natural language caption. (Xu et al., 2015)



(c) FiLM-Layer: The FiLM architecture linearly modulates feature channels referred to in natural language queries. As a result areas containing these feature channels are highlighted while processing the answer. (Perez et al., 2018)

Figure 2.2.1: Types of implicit and explicit attention in image processing applications.

most predictive of class[1]. The same is true for most machine learning models. However, this type of "attention" is only visible after the fact and will not aid in training the model, prompting the emergence of more integrated and explicit mechanisms.

Mnih et al. propose a neural network model that highlights specific image regions for further processing. Like saccades in the human eye, a location-network proposes a sequence of coordinates at which to extract higher resolution samples in a larger, low-resolution image. A recurrent neural network integrates these sequential "glimpses" to arrive at a class prediction. Glimpses, although of higher resolution, comprise much fewer pixels than the whole image. While learning speed may be somewhat diminished compared to traditional convolutional networks because of the use of a recurrent model, this approach reduces complexity significantly when scaling up input data. That is, regular image processing has linear complexity with respect to the size of the input image, where this approach provides better optimal case ratings. Unlike the introductory example, this network is not fully differentiable due to the "hard" selection of processing positions and therefore needs to be trained via RL instead of supervised learning. Additionally, the choice of position does not have an immediate or differentiable impact on the objective (classifying the contents of the image). Ba et al. extend this approach by replacing the RL objective with a free energy formulation. The fully differentiable model maximizes the joint probability of predicting the correct label at a particular chosen position. This model can classify more than a single object in an image by scanning the entire visual space sequentially and outputting a label prediction at

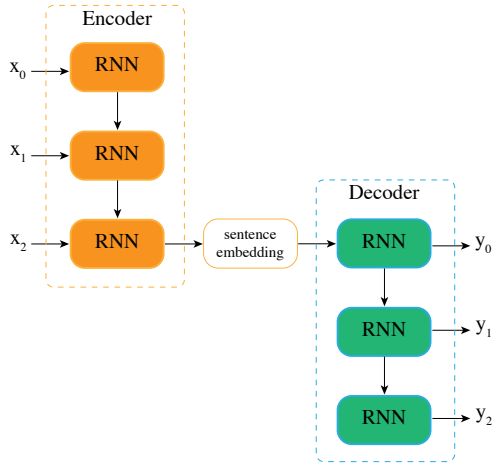
[1] A prerequisite for this is, of course, that the training data and model allow for an optimization process that is not grounded in spurious features (such as image backgrounds, leaked data, etc.)

every step. Such sequential models of visual processing went on to applications in image and video captioning (see Figure 2.2.1 b). Xu et al. propose a similar model comprised of a location-network and a recurrent integration model that, instead of labels, outputs full-sentence descriptions of the image contents. They also depart from a purely hard-selection process for the attentional spotlight by introducing a process of "soft" weighing and summing annotation vectors. Each annotation vector represents a different image region; thus, the model selects some regions over others in the final representation vector. The process was initially popularized by applications in natural language processing, specifically neural language translation, and will be described in more detail in the following section.

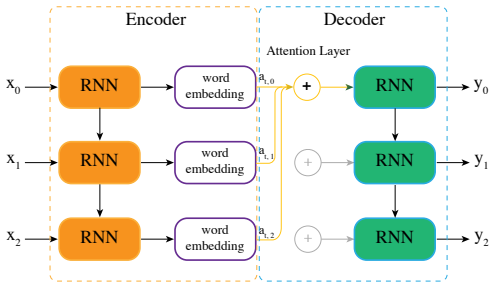
Apart from this variety of spatial attention models, there is a large number of feature-based methods. Stollenga et al. proposed a method that would allow pretrained convolutional neural networks to highlight individual feature maps in a recurrent process of reweighing. Classifying a single image would initiate a recurrent multistep process in the network that eventually culminated in a label prediction. However, feature-based attention in biological organisms is frequently task-conditioned or cued and used for visual search. De Vries et al. demonstrated a system that would dynamically adapt its parameters for batch normalization per convolutional channel, based on a question posed in natural language. The idea of task-conditioned parameters was carried on by various other approaches, such as the FiLM-layer (Feature-wise Linear Modulation), a convolutional layer that selectively highlights some feature maps over others, based on multi-faceted questions in natural language (Perez et al., 2018) (see Figure 2.2.1 c). The method combines multiplicative and additive attention, i.e., an affine feature transform, to identify feature maps corresponding to word tokens in natural language. It thus incorporates a variety of approaches that employ either one or the other (Hu et al., 2020) and predominantly condition the selection function on other elements in the input, resulting in bottom-up processing as opposed to top-down in the FiLM-architecture.

2.2.2 Natural Language

A common problem in machine translation is alignment, i.e., finding the correct source for the next word in a translated sentence when the two languages differ significantly in syntax or the source language uses multiple tokens for a single word in the target language and vice versa. This issue and similar problems have inspired the introduction of neural machine translation methods; complex neural networks trained end-to-end with backpropagation. Such models often comprise encoder-decoder architectures. The encoder generates representations of sentence elements by processing its entirety, thus capturing not only a word at a time but the overall meaning. The decoder then queries encoded sentence elements to translate and generate the sentence in the target language. The necessity for the whole sentence to be processed before translation starts is intuitive, as a word for word translation is unlikely to match the actual expression. Bahdanau et al. kicked off a massive trend in all sub-fields of machine learning by proposing a neural network model that selects sentence elements for translation through explicit attention weights (see Figure 2.2.2). Like previous works, they employed a bidirectional recurrent



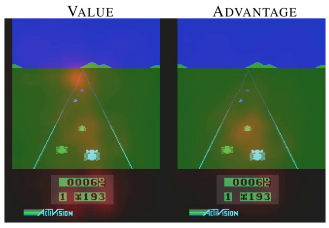
(a) The traditional encoder-decoder architecture for natural language translation. The whole sentence is represented as a single vector and subsequently translated.



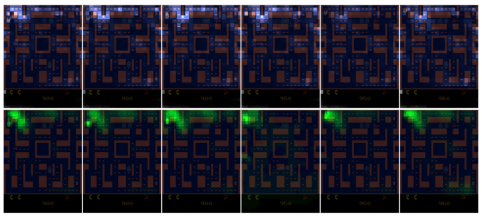
(b) Bahdanau et al. propose an encoder-decoder that considers attention-weighted temporal embeddings of every sentence token per word translation (including previous history and future tokens in the case of bidirectional encoders.)

Figure 2.2.2: Encoder-Decoder architectures for natural language translation using recurrent neural networks.

network that generates "annotations" by processing a sentence front-to-back and back-to-front. Each annotation is a vector representation of a word token enriched with the context of the entire sentence. Instead of merely using another recurrent network as the decoder, Bahdanau et al. added an attention weighting step. Where the decoder would traditionally query the next annotation vector, it would consider the entire sequence instead. This process was implemented by an "alignment model," conditioned on the recurrent decoder's previous hidden state and the current annotation. The output of the alignment model was a single scalar weight for each sequence element, i.e., the annotations. Thus every single translated word was represented by a mixture of all words in the source sentence before entering the decoding step. Despite its relative simplicity, the approach achieved impressive results in benchmark tasks, triggering the development of the self-attention (also known as dot-product attention) mechanism (Vaswani et al., 2017). This approach of generating weighted sums of input sequences constitutes one of the principal mechanisms of this thesis work and is thus treated comprehensively in the main chapter (chapter 4). Briefly, this work's significant contribution was the complete replacement of recurrent sequence models in encoder-decoder architectures with the equivalent of several, hierarchical alignment models, as introduced by (Bahdanau et al., 2015). Ever since, the self-attention mechanism has found application in all manner of language-based machine learning, from classification and translation to full-text generation.



(a) Jacobian of the separate value- and advantage streams in the dueling q-network architecture (Wang et al., 2016).



(b) The lower sequence of images shows the Jacobian of the value function estimate, demonstrating significant overlap with the predictive attention mechanism in the top row (Mott et al., 2019).

Figure 2.2.3: Visualizations of implicit and explicit saliency maps based on value- and advantage-estimation in reinforcement learning.

2.2.3 Reinforcement Learning and Control

Similar to the expression of implicit attention in the image processing section, neural networks trained via RL can exhibit selective properties with respect to task-relevant features after training. The dueling deep Q-network computes both state- and state-action-values separately to maintain estimates about the value of an observation and the specific advantages of each possible action (Wang et al., 2016). After training, the Jacobian of these value- and advantage-streams can highlights which parts of the input observation the network is basing its judgments on (see Figure 2.2.3 a). Again, this "attention" is implicit, and after the fact, hence it does not aid in learning. However, it does reaffirm the hypothesized need and benefit of attention.

Before the renaissance of attention in machine learning, many deep-RL approaches were already implementing partial solutions to the problems solved by biological attention. The Importance-Weighted Actor-Learner Architecture (IMPALA), among various other contributions, introduces a task-conditioning which allows external natural language queries to affect both actions and value judgments of the RL agent (Espeholt et al., 2018). These textual queries enter a recurrent subnetwork that processes sequential observations and generates responsive actions. Due to these dynamics, the agent can adapt rapidly to new queries and effectively implement policies in multi-task environments, i.e., switch its attentional focus on the fly and in a top-down manner. While the general idea is similar to task-conditioned parameters (De Vries et al., 2017; Perez et al., 2018), there is no change in parameters necessary after the agent has trained to perform in all tasks. Other approaches, such as the method of prioritized experience replay (Schaul et al., 2016), venture into more abstract areas of attention over time. Most value-function based RL methods can learn from a large volume of prior reward-observation samples where the agent samples a batch of experiences uniformly at every learning step. With extensive and uncurated memory buffers, this can lead to the resampling of mostly insignificant experiences. Prioritized experience replay tackles this issue by sampling those experiences more readily that the agent struggled to evaluate earlier - an idea in line with the theoretical Pearce-Hall model of attending those input

elements that the agent is most uncertain about.

Only recently have deep-RL approaches picked up on more explicit mechanisms of attention. These mechanisms are narrowly focused on the concept of spatial attention in the visual domain, an unsurprising fact given the success and popularity of deep-RL in tasks with RGB-image observations (Silver et al., 2016; Mnih et al., 2013). Mott et al. proposes a recurrent neural network model that sequentially queries an input volume, specifically a stack of RGB-images in the classic Atari 2600 benchmark (see Figure 2.2.3 b). Their approach is able to highlight image patches, significant for value prediction and decision-making. Notably, they implicitly integrate the bidirectional interaction of attention and learning. Only those image regions enter the decision-making and querying part of the model that have already been attended to. Thus, the agent is constrained in learning by its own selection regime, while this attentional focus is shaped by learning. Tang et al. equip their agent with similar capabilities. However, they apply a "hard" attention mechanism, reducing the size of the input used for further processing significantly. The model can not be trained via RL but instead learns via evolutionary algorithms, making it the sole competitor to build on evolved bottom-up attention theories. Because of the hard selection process, every fixed-sized image patch can be rated according to its importance for the fitness function, providing a catalog of the most salient stimuli.

Other approaches tread closer to an integrated, or multi-modal, view of attention along several layers of the processing hierarchy. Zambaldi et al. demonstrate a model with significant similarities to the one proposed by Mott et al.. However, their approach is entirely content-based and more heavily focused on the interactions of image regions, i.e., a relational inductive bias of visual processing. It is easy to see that such an approach can theoretically extend beyond the visual domain and integrate information from several sensory and memory sources, provided that a sufficiently abstract embedding space can be found during learning. Image regions are treated as so-called entities and, theoretically, more abstract than the highlighted image patches of the previous two methods. This paper's crucial contribution is the research into the benefits of non-local computation, one of the limitations of regular convolutional neural network models. The ability to relate image regions arbitrarily harbors significant benefits for visual reasoning, as demonstrated in their experiments on real-time strategy games.

2.3 Summary

In this chapter, we have introduced a multitude of dimensions along which attention affects both biological and artificial systems. Most notable are the following definitions: Bottom-up vs. top-down, in short, describes the triggers of attention. Where bottom-up attention is generally caused by the immediate perception of a salient stimulus, and top-down attention is either cued or conditioned on a task and thus a higher instance of control. It is difficult to draw this distinction in most machine learning applications since all models serve a rather specific purpose. Thus virtually all applied ML models are task-conditioned from the very first learning steps. However, the difference becomes much clearer in approaches that deal with

multi-task environments (Espeholt et al., 2018; Perez-Liebana et al., 2019), where the entire model is cued to perform various tasks through time. This paradigm is most pronounced in models that query different sets of parameters based on current tasks or cues (Perez et al., 2018; Espeholt et al., 2018). Spatial vs. feature-based describes a dichotomy of the centers of attention. Spatial selection is a dominant factor in visual processing, brought about by either bottom-up or top-down attention processes. It highlights specific image regions for further processing and either discounts or discards the remainder. The same may be true for different modalities, where space is defined in other dimensions, such as time in auditory tasks. While the ML examples presented in the above paragraphs seem to lean on spatial attention, the distinction is not that clear. For example, the model proposed by Mnih et al., while selecting sequential foci for spatial visual processing, is noticeably influenced by salient features in the input image during the classification of the MNIST data-set. This phenomenon may well support the theory of feature-based attention guiding spatial attention. On the other hand, explicit feature-based attention has remained relatively rare in ML methods and is mostly found in natural language processing and the rare case of content-based attention in control and RL. Attention to memory can be regarded as an extension of feature-based attention in that different, and more abstract representations are selectively stored and retrieved by the mechanism. Naturally, there are different modalities of sensory experience as well, and their integration with prior knowledge factors into most decisions that intelligent beings make. Biological organisms integrate several streams of information with apparent ease. However, neuroscience literature suggests that there is a shared conceptual budget of attention between these venues (Gardiner and Parkin, 1990). Some ML approaches integrate both natural language and visual experience to implement cued task solving, but due to the very specialized tasks in most approaches, this is the rare outlier. There is no example of an implemented, limited focus over multi-sensory and memory integration. There are, however, standalone models that implement an attentional selection and retrieval process similar to the hypothesized gatekeeper in working memory models. Both the neural Turing machine (Graves et al., 2014) and the differentiable neural computer (Graves et al., 2016) are neural network models comprising several parallel read and write heads to work as a trainable computer — one element of this two-way process has made its way into RL, in several approaches that selectively sample prior experiences to train an agent effectively (Schaul et al., 2016; Andrychowicz et al.). While all experiences enter these memories indiscriminately, they will be re-sampled for learning based on prior reward prediction errors. These mechanisms arose before the resurgence of attention research in ML circles, and thus we call them implicit in the previous chapters. However, the principles reigning them tend to be faithful to common neuroscience theories on attention mechanisms, perhaps even more so than the variety of models mentioned in the latter half of the RL and control subsection. Artificial attention will bend towards those stimulus or memory elements most relevant for minimizing the objective function, no matter the mechanism implementing it. Prioritized replay, however, implements an inductive bias - we already know that past reward prediction errors are excellent indicators of valuable experience, and their prioritization will lead to faster learning about relevant prior observations (Mattar and Daw, 2018). This specific bias corresponds to an immutable interpretation of the Pearce-Hall model, while all approaches that differentiate attention patterns with respect to some objective function will implement

the Mackintosh framework through emergent dynamics in the respective model. This dichotomy becomes particularly evident in value-based RL-methods, where attention is implicitly shaped by learning about the environment's value function. Ultimately, this line of thought leads us to the crucial two-way-road perspective, introduced by Niv et al.; Leong et al.; Radulescu et al., in which attention constrains what we perceive and therefore learn about while learning about things (especially value association) shapes our attention. We will investigate the implications of this view more thoroughly following the methods section.

Chapter 3

Methods

3.1 Artificial Neural Networks

Artificial Neural Networks (ANNs, or NNs) are adaptive, nonlinear functions that were originally inspired by biological neural networks, found in animal brains. The adaptivity of these systems stems from a number of trainable parameters which constitute connection weights between parallel and successive network nodes called neurons. These weights roughly correspond to synapses in animal brains, which too have adaptive strengths and regulate the magnitude of transmitted signals. In the same way, neurons collect signals transmitted through connected synapses and emit a signal when a certain threshold is surpassed thus allowing both artificial and biological networks to implement complex and nonlinear logical operations. This computational model was conceived as early as 1943 and termed the "threshold logic" model (McCulloch and Pitts, 1943). Several learning rules emerged to train the elements of such networks, with Hebbian Learning (Hebb, 1949) and Backpropagation (Rumelhart et al., 1986) being the most prevalent in contemporary research. Due to limitations in computational capacities, however, ANNs fell by the wayside until they experienced a major surge in popularity in the last decade. An increase in available computational resources facilitated the breakthrough success of NNs in the late 2000s, allowing applications to surpass numerous state-of-the-art machine learning methods in image classification, natural language processing and ultimately RL. Specifically, these performance gains were possible because of the large number of trainable parameters and successive network layers in the presented models, giving rise to the deep learning paradigm.

The underlying architecture and learning rules in this work are implemented using artificial neural network models, which provide a general function layout and learning paradigm. This section will lay out the basic mechanism of information processing in ANNs, discuss how neural network weights are adapted, that is "trained", using backpropagation, and what special architectures harbour which benefits.

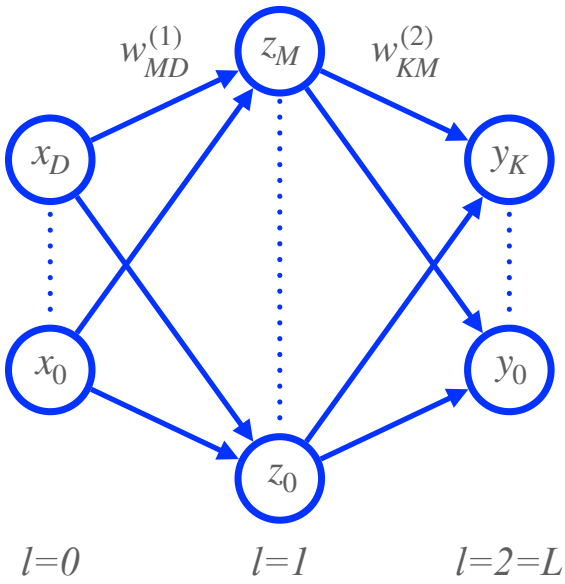


Figure 3.1.1: A multi-layer perceptron with a single hidden layer $l = 1$.

3.1.1 General Mechanism

Formally, ANNs are models represented by computation graphs and thus comprised of nodes (neurons) and edges (weights). Multiple nodes constitute a layer, with subsequent layers of nodes mapping an input vector to an output vector. In the simplest case, termed *Multi-Layer Perceptron* (MLP), every node in layer (l) has a weight, connecting it to every node in layer $(l+1)$ (illustrated in Figure 3.1.1). The concept of excitation in biological neurons translates roughly to that of the nodes in artificial neural networks. The input vector provides a set of scalar values that can be interpreted as the output of an initial set of neural network nodes. Each subsequent node then evaluates whether it emits a signal based on the weighted input signals passing along the connected network edges as such:

$$a_j = w^{(1)}_{j0} + \sum_{i=1}^{D} w^{(1)}_{ji} x_i \tag{3.1}$$

where $w^{(1)}_{ji}$ is a single weight scalar out of the weight matrix $\mathbf{W}$, connecting each of $i \in \{1, 2, ..., D\}$ input values x_i to the jth neuron in the (1)st hidden layer. $w^{(1)}_{j0}$ is an additional bias term, comparable to the same term in linear models for, e.g., regression analysis. The term can be absorbed by appending a constant value $x_0 = 1$ to the input vector (and correspondingly $a_0 = 1$ for intermediary hidden layers), such that the formula reduces to:

$$\begin{aligned} a_j &= \sum_{i=0}^{D} w_{ji}^{(1)} x_i \\ a_j &= \mathbf{W}_{j,:}^{(l)\top} \boldsymbol{x} \end{aligned} \tag{3.2}$$

which is the dot product between the input vector $\boldsymbol{x}$ and the weight vector of node j. a_j can then be understood as the formal representation of a membrane potential. Whether or not the artificial neuron is excited by this potential depends on the interaction of the potential's magnitude and a nonlinear activation function $h(a_j)$. These functions govern the input, output relationships of each individual neuron through both activation thresholds and corresponding return values. The computation of the activation value a_j is a simple affine transformation, while the nonlinear activation function endows the model with the ability to find nonlinear decision boundaries or approximate nonlinear functions. Common examples for such functions are the logistic function (see Equation 3.3), or the hyperbolic tangent function.

$$\begin{aligned} z_j &= h(a_j) = \sigma(a_j) \\ \text{where } \sigma(a_j) &= \frac{1}{1 + e^{-a_j}} \end{aligned} \tag{3.3}$$

There exist a variety of different activation functions each with their specific qualities. The principal requirement for an arbitrary mapping to be an activation function is the possibility for it to be differentiated. In recent years, the *Rectified Linear Unit* (ReLU) function has gradually deposed the classic sigmoidal activations in hidden layers because of its convergence speed and general simplicity:

$$ReLU(a_j) = \max(0, a_j) \tag{3.4}$$

For the final layer in each neural network, the adoption of sigmoidal activations is still common, as they clamp outputs to the $[0, 1]$ or $[-1, 1]$ intervals. This allows an intuitive interpretation of neural network outputs as either labels or mutually non-exclusive label probabilities. If mutual exclusivity is a desired property of the neural network output, the softmax function is employed at the final layer:

$$\begin{aligned} y_j &= \frac{e^{a_j}}{\sum_{i=1}^{M} e^{a_i}} \\ \text{such that } &\sum_{j=1}^{M} y_j = 1 \end{aligned} \tag{3.5}$$

Thus the outputs of the final layer will sum to one and may readily be interpreted as a probability distribution over output nodes. Once the number of layers L, the numbers of nodes per layer, and the activation functions have been decided upon, the network can be formally expressed as a deeply nested function with interchanging calls to dot-product and activation functions. For the concrete example of a single hidden layer MLP, that is $L = 2$, and anonymous activation functions $h(\cdot)$ the entirety of the input-output mapping can be defined as:

$$y_k(\boldsymbol{x}, \mathbf{W}) = h\left(\sum_{j=0}^{M} w_{kj}^{(2)} h\left(\sum_{i=0}^{D} w_{ji}^{(1)} x_i\right)\right) \tag{3.6}$$

3.1.2 Optimization

In their earliest application, MLPs were used with manually crafted weights to implement nonlinear function mappings. However, the reason for their revival was the feasibility of automated training procedures in deeper neural networks with far more parameters - most prominently gradient descent with error backpropagation. In traditional supervised learning paradigms, the network processes a set of input data samples $\mathbf{X} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \boldsymbol{x}_3, ..., \boldsymbol{x}_N\}$ and attempts to predict associated true labels $\boldsymbol{y} = \{y_1, y_2, y_3, ..., y_N\}$. A predefined loss function determines how well the network is doing. One of the most intuitive examples of a loss function is the *Mean-Squared Error* (MSE):

$$\mathcal{L} = \frac{1}{n}\sum_{i}(y_i - \hat{y}_i)^2 \tag{3.7}$$

where $\hat{y}_i$ denotes the network-predicted label, and n is the number of processed input data samples which may be the full set with $n = N$ or any smaller subset of $\mathbf{X}$, termed a mini-batch. The network weights are initialized semi-randomly, depending on the initialization scheme; thus, performance will be sup-bar as training commences. Weights will then be updated iteratively using gradient descent, where the size of gradient descent steps is regulated by a learning rate hyperparameter η:

$$\mathbf{W}^{\tau+1} = \mathbf{W}^{\tau} - \eta\nabla\mathcal{L}(\mathbf{X}_{1:n}, \mathbf{W}^{\tau}) \tag{3.8}$$

In the case of updates using smaller mini-batches, we speak of stochastic gradient descent, as opposed to batch-gradient descent. Note that while the evaluation of the loss function depends on the input data, the optimization process only manipulates the actual network parameters. The gradient of the loss function with respect to each individual weight is computed using the chain rule of partial derivatives, as follows:

$$\nabla\mathcal{L}(w_{ji}) = \frac{\partial\mathcal{L}}{\partial w_{ji}} = \frac{\partial\mathcal{L}}{\partial a_j}\frac{\partial a_j}{\partial w_{ji}} \tag{3.9}$$

Simply put, we differentiate the loss function with respect to the network activations, which are, in turn, differentiable with respect to the network weights. Computing the gradient then allows us to shift the network weights to reduce the overall loss function. Performing this gradient descent procedure iteratively allows for a more careful exploration of the high-dimensional loss landscape and facilitates convergence to global minima.

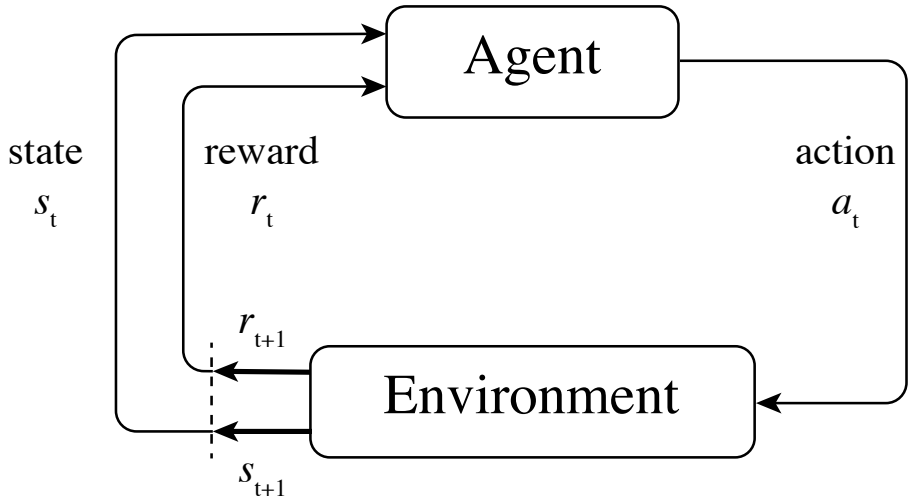


Figure 3.2.1: Agent-environment interaction in a Markov Decision Process.

Throughout this work, we will employ ANNs in the RL paradigm, widely regarded as the third principal learning paradigm beyond supervised and unsupervised learning. The above sections describe the application of NNs in supervised learning, which provides a clear definition of optimal solutions - finding a global optimum in the loss function. However, these loss functions tend to be far more intricate in RL problems and rarely allow for stationary solutions because of shifts in both the structure of input $\mathbf{X}$ and output data $\boldsymbol{y}$ distributions. The following sections will discuss these challenges in detail.

3.2 Reinforcement Learning

Reinforcement Learning (RL) is a semi-supervised learning paradigm that allows an agent to learn optimal behavior by interacting with an environment. In this chapter, we will discuss the formalisms of both entities and their interactions - starting with the environment, followed by methods that describe how to optimize behavior.

3.2.1 Generalities

The RL framework is adept in formulating a vast array of control problems, in part, due to its formalization of the environment as *Finite Markov Decision-Processes* (MDP). Interactions take place at each of a sequence of discrete time steps $t = \{0, 1, 2, 3, ..., T\}$. At each step, the agent either receives an observation emitted by the environment or its actual state, based on which it grounds its selection of the next appropriate action. The agent receives a scalar reward based on both the previous state and the selected action in the following time step. In this way, the interacting entities give rise to a trajectory $\tau = \{s_0, a_0, r_0, s_1, a_1, r_1, ..., s_T, a_T, r_T\}$ (see Figure 3.2.1). The agent attempts to pick actions in such a way as to maximize the overall reward of the trajectory. Definitions of MDPs have varied over the

decades since their original conception. Here and throughout the remainder of this thesis, we will use an intuitive formulation of the MDP as a four-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{R}, p)$, based on Sutton and Barto (2018). Where $\mathcal{S}$ is a finite set of states, $\mathcal{A}$ is a set of possible actions, $\mathcal{R}$ is a set of possible scalar rewards, and p is a stochastic dynamics function, such that $p : \mathcal{S} \times \mathcal{R} \times \mathcal{S} \times \mathcal{A} \mapsto [0, 1]$. This tuple describes a time-discrete MDP, meaning that there is a countable set of states and that each agential action prompts a single state-transition. The MDP is entirely defined by these sets, and the accompanying dynamics function p:

$$p(s_{t+1}, r_t | s_t, a_t) = \Pr\{S_{t+1} = s_{t+1}, R_t = r_t | S_t = s_t, A_t = a_t\} \tag{3.10}$$

Where $S_t, S_{t+1} \in \mathcal{S}, R_t \in \mathcal{R}, A_t \in \mathcal{A}$ denote random variables that take on specific values s_t, r_t, a_t from their respective finite sets. While s_t and r_t are produced by environment dynamics, actions a_t are generated by an agent's policy, which is given by a parametric mapping of states to actions $\pi_\theta : \mathcal{S} \mapsto \mathcal{A}$.The dynamics function in Equation 3.10 factorizes the stochastic elements of the environment. It describes its associated dynamics in full, namely, defining a distribution over successive states and rewards, conditioned on previous states and agential action. Thus the distribution prescribes a factorization of the probability of observing a specific trajectory, given a certain policy:

$$p(\tau) = \rho(s_0) \prod \pi_\theta(a_t | s_t) p(s_{t+1} | s_t, a_t) \tag{3.11}$$

As the name suggests, MDPs underly the Markov property; in other words, the dynamics function is conditioned on the previous state alone and disregards any information about the previous trajectory. This strong assumption of independence drastically reduces the complexity of the environment and the associated graph structure. Importantly, this distribution may take the form of a deterministic function, meaning that it is a simple mapping from state-action pairs to successive state-reward pairs in an environment without stochastic transitions.

The problem statement, then, is for the agent to maximize expected scalar return by induction of a trajectory of states under a selected sequence of actions and the environment's dynamics function p. This objective is grounded in a repurposing of the dynamics function, such that $r : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$:

$$\begin{aligned} r(s, a) &= \mathbb{E}[R_t | S_{t-1} = s, A_{t-1} = a] \\ &= \sum_{r \in \mathcal{R}} r \sum_{s_{t+1} \in \mathcal{S}} \pi_\theta(a_t | s_t) p(s_{t+1}, r_t | s_t, a_t) \end{aligned} \tag{3.12}$$

Where $r(s, a)$ now describes the scalar reward earned by executing action a_t in state s_t. Intuitively, some states in the environment or choices of action are more desirable than others, and it is the goal of the agent to encounter these frequently. Additionally, some states may require a specific action selection to emit a reward. Actions, on the other hand, may incur costs. For example, movement may consume energy when it is the agent's goal to maintain high energy levels by foraging food sources. The environment is considered solved when the optimal policy pa-

rameters θ^* are found, that maximize returns across the length T of the trajectory τ:

$$\begin{aligned} J(\theta) &= \mathop{\mathbb{E}}_{\tau \sim p(\tau)} \left[\sum_{t=1}^{T} r(s,a) \right] \\ \theta^* &= \mathop{\text{argmax}}_{\theta} J(\theta) \end{aligned} \tag{3.13}$$

Maximizing rewards across the entirety of a trajectory with T time-steps gives rise to the definition of return G_t, that is, the rewards received after time-step t:

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + ... + R_T \tag{3.14}$$

Many MDPs include terminal states, which signify the end of an episode. Given the accompanying objective of maximizing the expected return, these terminal states limit the objective function's time horizon T. These environments are episodic, meaning that there is a natural limitation between trajectories. The introduction of a discount parameter is useful in MDPs that forego terminal states,i.e., are non-episodic. The restated objective will not go towards infinity and favor returns in the near future, depending on the choice of a discount parameter $0 < \gamma < 1$. The discounted return is defined as follows:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1} \tag{3.15}$$

Intuitively, this introduces the notion of recursion that is ubiquitous throughout the study of RL. The return can be expressed in terms of the relationship between successive time-steps which is paramount for the theory behind the algorithms in the remainder of this chapter:

$$\begin{aligned} G_t &= R_{t+1} + \gamma(R_{t+2} + \gamma R_{t+3} + ...) \\ G_t &= R_{t+1} + G_{t+1} \end{aligned} \tag{3.16}$$

With the environment and the notion of rewards defined, we will investigate the intricacies of agent behavior in the following section.

3.2.2 Policies and Value Functions

As mentioned previously, the agent selects from a choice of actions to perform at each time step. This choice is a function of the current observation and is commonly described as a policy π. The design of this function depends entirely on the chosen approach to solve a given RL problem. The simplest example is a tabular solution, which records tuples of observations, actions taken, and rewards received. After a random exploratory period, the agent can base its choice of action on previous experiences. Given a before-seen observation, the agent will pick the action with the highest associated return. It is easy to see that this approach has severe limitations

in assuming that the observation space is small and easily comparable. The more common procedure, is thus, to define the policy as a function of state under some parameters θ. Depending on the makeup of that function, its parameters may then be optimized by gradient methods. The general objective, i.e. the optimal policy, then is to maximize the expected return with respect to the parameters of the policy distribution, posing an unconstrained optimization problem as follows:

$$\pi_\theta^* = \underset{\theta}{\operatorname{argmax}}\, \mathbb{E}\left[\sum_{t=1}^{T} r(s_t, a_t)\right] \tag{3.17}$$

RL relies heavily on the notion of expected returns. Therefore, most RL algorithms involve estimating a value function of the environment $V^\pi(s_t)$, which describes the value of the agent occupying a specific state s, given a policy π. In these terms, value is defined by the future returns the agent can acquire if they follow the policy starting at this particular state of the environment. Unlike the above-mentioned tabular method, this approach inherently captures the causality of an environment, because a state's value is defined by both its immediate return and its potential to lead to future returns. The necessity of conditioning on the current policy is easily explained in that realizing high returns from states with high potential requires the execution of practical actions. An important preliminary to the value function is the Q-value function, which describes not the value of a state, but the expected value of a state-action pair. In simpler terms, what is the expected benefit of choosing action a in state s, such that $Q : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$? The Q-value function is given by:

$$Q(s_t, a_t) = \sum_{t=1}^{T} \mathbb{E}[R_t | s_t, a_t] = \sum_{t=1}^{T} \mathbb{E}[r(s_t, a_t)] \tag{3.18}$$

and the corresponding update rule:

$$Q^{new}(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha R_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \tag{3.19}$$

Importantly, this expression illustrates the recursive relationship of values, where a state's value is given by a combination of immediate reward and the γ-discounted state-value function of the successor states under a policy. In this case, the policy is to choose whatever action maximizies the state-action value expression. Marginalizing over all possible actions gives the value as a function of the state alone, formally defined as follows:

$$V(s_t) = \mathop{\mathbb{E}}_{a \sim \pi_\theta}[Q(s_t, a)] \tag{3.20}$$

By integrating the expectation term, we find that this is the Bellman Equation, thus establishing the necessary condition for optimality with respect to action selection in dynamic programming:

$$V(s_t) = \max_{a_t} \left(r(s_0, a_0) + \gamma \sum_{t=1}^{T} p(s_{t+1}|s_t, a_t) V(s_{t+1}) \right) \tag{3.21}$$

3.2.3 REINFORCE

The REINFORCE algorithm describes our first complete instruction set of how an agent learns to act, by computing the above quantities through sampling an environment. To evaluate the measure of performance described in Equation 3.16, and thus compute the approximate gradient proportional to the real gradient of the objective function 3.22, the agent requires empirical samples of state-action pairs (see section B in the supplementary materials for details on the policy gradient theorem).

$$\nabla J(\theta) \propto \sum_{s} \mu(s) \sum_{a} q_\pi(s, a) \nabla \pi_\theta(a|s, \theta) \tag{3.22}$$

By gathering samples of state-action trajectories and their associated returns, we can employ gradient ascent to search the parameter space for the optimal policy-mapping. By restating the expression above, we can derive the gradient of the objective expectation over returns with respect to the policy:

$$\nabla J(\theta) \propto \mathbb{E}_{a\sim\pi} \left[\sum_{a} q_\pi(s_t, a) \nabla \pi_\theta(a|s_t) \right] \tag{3.23}$$

The stochasticity results from the empirical sample of the environment, represented by $s = s_t$, as we can only cover states within reach of the current policy. Additionally, the policy itself may be stochastic, as might the environment's transition function. This limitation requires repeat roll-outs, i.e., re-sampling of the ever-adapting policy. REINFORCE computes the complete returns at time t, meaning that it computes the value, including all future returns until the end of the episode, and is thus termed the Monte-Carlo policy gradient.

Replacing the sum over random variables in the policy gradient theorem with empirical samples $s = s_t$ and actions $a = a_t$ gives an applicable gradient of the parameters. Additionally, the sample of the returns is a sample of the true state-action value function of the environment and thus q_π can be replaced with G_t in the gradient as follows:

$$\nabla J(\theta) = \mathbb{E}_\pi \left[G_t \frac{\nabla \pi_\theta(a_t|s_t)}{\pi_\theta(a_t|s_t)} \right] \tag{3.24}$$

The newly introduced ratio of the policy gradient ensures that likelier actions, which will generally be updated more frequently due to the stochastic sampling of the policy distribution, are updated in smaller steps than less probable actions. Consequently, the REINFORCE update is as follows:

$$\theta_{t+1} \leftarrow \theta_t + \alpha G_t \frac{\nabla \pi_\theta(a_t|s_t)}{\pi_\theta(a_t|s_t)} \tag{3.25}$$

Intuitively, this update rule changes the probability of choosing action a_t in state s_t, by the amount of the received returns G_t through manipulating the state-to-action mapping parameterized by θ. The vector of derivatives supplies the direction in parameter space that maximizes this probability if the return is positive. The update size is then proportional to the received return and inversely proportional to the action probability. This formulation ensures that the update maximizes action probabilities associated with high returns but mitigates the previously mentioned bias towards highly probable actions. For example, if a highly likely action continuously generates moderate returns, it would reach an exceptionally high probability precluding any further exploration within only a few updates. This case, however, will be prevented by the ratio term in the update rule. Generally, observing a high, positive return will significantly increase the probability of a given action, while a large negative return decreases it. As mentioned previously, the policy is a smooth function, allowing the exploitation of learned mappings in states that the agent has not yet seen, but that resemble previously experienced states.

REINFORCE has good theoretical convergence properties because of its stochastic gradient approach. The expected update over an episode is sure to follow the same direction as the true gradient of the performance measure. Under this property, REINFORCE will converge in close proximity to the optimal solution of a finite MDP, provided the step size of the update α is sufficiently small. However, the learning rate is not the only influence on the magnitude of the update step. As a Monte-Carlo method, REINFORCE is quite susceptible to high variance across updates, possibly leading to slow learning or premature convergence in sub-optimal parameter configurations. A simple way to handle these issues is the introduction of a baseline function.

3.2.4 Value Function Estimation

We can re-frame the previously derived stochastic gradient update in a more general expression by including a baseline function $b(s)$. This additional term allows our update steps to compare the received returns in a relative sense, instead of taking them at face value. The baseline can be any function, even a simple constant. Using a constant baseline will be useful in an environment where the agent receives a positive return at every state. If the constant value of the baseline is 0, the algorithm reduces to REINFORCE's base case. For most MDPs, the baseline should be a function of the state or the observation (given the partially observable case). The state has a significant impact on the viability of actions and potential future returns in MDPs. Therefore, a sensible choice for the baseline function would be an estimate of the value function. This assumption presumes that the estimate is continuously updated alongside the policy, which requires the value function to be represented by parameters ϕ. The parameterization allows the algorithm to update both policy- and value-functions with the same trajectory.

Using an estimate of the environment's value function introduces a new intuition

to the REINFORCE algorithm. The value function is a function of the state alone, recall:

$$V(s_t) = \sum_a \pi(a|s_t) Q(s_t, a)$$

Hence the update rule for policy and baseline (in this case state-value function $V(s_t)$):

$$\begin{aligned} \delta &= G_t - V_\phi(s_t) \\ \phi_{t+1} &\leftarrow \phi_t + \alpha^\phi \delta \nabla V(s_t) \\ \theta_{t+1} &\leftarrow \theta_t + \alpha^\theta \delta \nabla \log \pi(a_t|s_t) \end{aligned} \tag{3.26}$$

However, the real returns of the trajectory are conditioned on the specific actions the agent has selected. For one, we can interpret δ, that is, the discrepancy between the estimated and real value of state s_t as a measure of surprise or reward prediction error. How much better than expected via $V(s_t)$ did the agent do by selecting the action a_t? Additionally, this disparity represents a measure of information about one action's value out of the sum over all other possible actions. For example, a high δ amount represents an advantage of the chosen action A_t over the average return of all actions. The inverse is true if δ takes on a negative value. Due to this property, the algorithm will enforce (i.e., increase the probability of) only such actions that surpass the average return of all actions. If the chosen action matches the average return, REINFORCE will neither reduce nor increase its associated probability.

With these considerations, REINFORCE serves as an essential precursor to a large family of modern RL algorithms, namely (Advantage) Actor-Critic approaches, which we will examine in detail in the following sections.

3.3 Actor-Critic Methods

Actor-critic methods are a family of state-of-the-art policy gradient approaches that constitute a counterpart to more traditional value-based or solely policy-based RL solutions like, for example, Q-learning or REINFORCE (Konda and Tsitsiklis, 2000). They implement straightforward policy search with gradient-based methods. However, unlike the larger family of policy gradients, they also maintain an explicit value function estimate for gradient steps with increased stability.

3.3.1 Temporal Difference Learning

The REINFORCE algorithm with baseline does not qualify for the descriptor of "actor-critic method," despite its inclusion of the two central elements: parameterized policy (actor) and value-function estimate (critic). The reason for this is the introduction of the δ_t term. Actor-critic methods replace the Monte-Carlo return G_t

in the REINFORCE update rule with a *Temporal-Difference* (TD) estimate, as follows:

$$\begin{aligned} \delta_t &= r_t + \gamma V(s_{t+1}) - V(s_t) \\ \text{and } V(s_t) &\leftarrow V(s_t) + \alpha[\delta_t] \end{aligned} \tag{3.27}$$

For the TD(0), that is, the single-step case, this means that the value function estimate does not only serve as a baseline but also as a critical evaluation of the future states that result from the current action. In other words, the value function estimate is not only present to measure the advantage of actions taken and reduce variance in the learning rule but also to update the value estimates recursively. Sutton and Barto term this process bootstrapping. It involves updating the value function estimate with a single empirical return and an estimate of the successor state's value - using the current value function estimate, instead of Monte-Carlo trajectories and their associated returns G_t. This method at the center of TD learning introduces bias and an asymptotic dependence on the quality of the approximate value function. However, this bias is beneficial to learning stability, in that it reduces the variance of otherwise unbiased estimators like the one used in REINFORCE. The TD target can be computed across an arbitrarily sized horizon by bootstrapping empirical returns from up to n steps into the future, instead of only a single step. TD(0) easily extends to the general case called TD(λ). The λ hyperparameter denotes the bias of the TD-target, with $\lambda = 0$ introducing a large bias towards the value function estimate and $\lambda = 1$ constituting the fully unbiased Monte-Carlo returns, analogous to G_t.

$$\begin{aligned} z_t &\leftarrow \gamma\lambda z_{t-1} + \nabla V(s_t) \\ \phi_{t+1} &\leftarrow \phi_t + \alpha^{\phi}\delta z_t \end{aligned} \tag{3.28}$$

Intuitively, this formula describes how lesser λ values relegate the influence of empirical future returns, while λ values towards 1 approach the unbiased estimator. Using TD as a target for value function estimates in actor-critic models thus requires $0 \leq \lambda < 1$. Note, that unlike the Monte-Carlo estimate used in REINFORCE, this approach does not require the episode to terminate, because it relies on value estimates instead of empirical returns. In theory, this allows actor-critic methods to update both policy- and value-functions at any point in time. In the case of TD(0), updates are possible at every time step.

As mentioned in the previous section, the value function estimate attempts to fit target values that comprise information about the advantage of taking one action above the other. The same is true for the TD-target. Therefore, most modern actor-critic algorithms present their TD-target estimator as an advantage function or estimate, which reduces to the difference between state-action-value and state-value functions:

$$A(s_t) = Q(s_t, a_t) - V(s_t) \tag{3.29}$$

The intuition behind this approximate equivalence is that the empirical return $r(s, a)$ takes the action a_t as an argument. The discounted value estimate of the successor state is added to the expression. While the value estimate itself is not conditioned

on a_t, the probability of reaching the successor state s_{t+1} is, as becomes evident in the following equality:

$$\begin{aligned} Q(s_t, a_t) - V(s_t) &\approx r(s_t, a_t) + \gamma V(s_{t+1}) - V(s_t) \\ &\approx r(s_t, a_t) + \gamma V(P(s_{t+1}|s_t, a_t)) - V(s_t) \end{aligned} \tag{3.30}$$

Thus, the term before subtracting the baseline is a biased estimate of the true Q-value function. It is biased because it relies on a future value estimate, provided by the current approximate value function.

One example of an algorithm that unites all these considerations is the *Advantage Actor-Critic* (A2C), which remains a significant baseline in various studies, along with its later developed asynchronous counterpart (A3C).

Algorithm 1: Advantage Actor-Critic (A2C)

Initialize step counter $t = 0$, and maximum horizon T
Initialize parameters for policy θ and value-function estimate ϕ
for $k = 1, 2, ...$ **do**
  Run policy $a_t \sim \pi_\theta(a|s_t)$ for T time-steps to collect trajectory $\{s_0, a_0, r_0, s_1, a_1, r_1, ..., s_T, a_T, r_T\}$
  $R = \begin{cases} 0, & \text{if } s_t \text{ is terminal} \\ V(s_t), & \text{else for non-terminal } s_t \text{ bootstrap from last state} \end{cases}$
  for $i = t - 1, ..., t - T$ **do**
    $R \leftarrow r_i + \gamma R$
    Accumulate gradients for policy and value-function estimate
    w.r.t. $\theta' : d\theta \leftarrow d\theta + \nabla_{\theta'} \log \pi(a_i|s_i)(R - V(s_i))$
    w.r.t. $\phi' : d\phi \leftarrow d\phi + \frac{\partial (R - V(s_i))^2}{\partial \phi'}$
  end
  Update parameters θ and ϕ using accumulated gradients $d\theta$ and $d\phi$
end

3.3.2 Generalized Advantage Estimation

One key issue in RL is the credit assignment problem; that is, the difficulty of attributing the contribution of an earlier action to an eventual positive or negative reward encountered several time steps after the fact. Reliance on a biased value function estimate is an effective step to reconcile this problem. It allows us to assess an action's quality before the delayed reward arrives, especially when it has converged after ample training on semi-biased TD-targets.

However, bias can also prevent a learning algorithm from converging. The alternative is to use an unbiased estimator, as described in the section on REINFORCE and its Monte-Carlo expression. This approach will converge invariably, but the time it requires to do so is entirely dependent on the variance in the environment.

Schulman et al. propose *Generalized Advantage Estimation* (GAE) as a formal framework to establish the delicate balance between too much and too little learning bias.

The approach is analogous to the TD(λ) formalism, notably in this case, not for value function estimate but the approximate advantage function. The traditional advantage estimation breaks down to the following definitions:

$$\begin{aligned} \delta_t &= r_t + \gamma V(s_{t+1}) - V(s_t) \\ \hat{A}_t^{(k)} &= \sum_{l=0}^{k-1} \gamma^l \delta_{t+l} \\ &= -V(s_t) + r_t + \gamma r_{t+1} + ... + \gamma^{k-1} r_{t+k-1} + \gamma^k V(s_{t+k}) \end{aligned} \tag{3.31}$$

Thus the classic case poses a k-step estimate of the advantage function. The GAE case then, is an expontentially-weighted average of a sequence of such δ_t estimations:

$$\begin{aligned} \hat{A}_t^{\text{GAE}(\gamma,\lambda)} &= (1-\lambda)\left(\hat{A}_t^{(1)} + \lambda \hat{A}_t^{(2)} + \lambda^2 \hat{A}_t^{(3)} + ...\right) \\ &= \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l} \end{aligned} \tag{3.32}$$

Despite the simple formulation in this derivation's ultimate step, the two provided parameters serve entirely different purposes. Where γ regulates the contribution of each future return to a single advantage estimate via discounting of steps further in the future (including the nth value function estimate), and λ assigns weights to advantage estimates with different k-step returns. In other words, γ discounts future returns, while λ discounts less-biased and, therefore, higher variance advantage estimates. The GAE is thus wholly defined by the two hyperparameters, γ and λ. GAE provides the same special cases as TD learning, such that:

$$\begin{aligned} &\text{GAE}(\gamma, 0) : \hat{A}_t = \delta_t &&= r_t + \gamma V(s_{t+1}) - V(s_t) \\ &\text{GAE}(\gamma, 1) : \hat{A}_t = \sum_{l=0}^{\infty} \gamma^l \delta_{t+l} &&= \sum_{l=0}^{\infty} \gamma^l r_{t+l} - V(s_t) \end{aligned} \tag{3.33}$$

That is, $GAE(\gamma, 0)$ is the biased, one-step advantage estimate, and $GAE(\gamma, 1)$ corresponds to the unbiased Monte-Carlo advantage estimate. Choosing λ such that $0 < \lambda < 1$ will allow us to strike a balance in the bias-variance trade-off sensibly and with consideration to the behavior of the selected environment.

3.4 Stabilizing Policy Gradients

Most RL problems are notoriously non-stable. One of the premier sources of instability is that the agent's experience is conditioned on its own, continually changing behavior. As the agent adapts to and begins to overcome the environment's challenges, it will invariably discover new states and thus never before seen observations. This change in observations and encountered states brings with it novel state-action-reward mappings for the agent to learn. In more formal terms, as the

agent adapts its policy to earn more rewards in the environment, the distribution of sampled states and rewards will change as a result, adding high variance across the entirety of the learning process. Worse still, if the agent pursues a suboptimal policy, the sampled data might reinforce bad behavior. By this mechanism, the reliance on the agent's policy to collect further data can lead to a vicious cycle: the agent updates its policy on a bad sample leading to a suboptimal policy, leading to more flawed samples, further reinforcing a disastrous drop in performance.

The challenge for the learning algorithm then is to remain general enough, i.e.w, not to overfit the policy parameters to observations encountered in the early stages of learning, so as to maintain convergence properties once the environment is more fully explored. Value-based methods Like Q-Learning struggle significantly with this problem because they rely on accurate value estimates for action selection. If the approximate value function converges too quickly, the agent will be unable to predict beneficial actions, should it ever encounter harder to reach states of the environment. Gradient methods are affected by the issue to a lesser extent, but as mentioned previously, they are plagued by the significant variance in most learning environments and demand a large number of state-action-reward samples to converge reliably.

This problem ties in closely with the exploration-exploitation dilemma, which describes the difficult task of deciding between the exploitation of prior knowledge and the acquisition of new experience to improve performance. Value-based RL methods commonly attempt to mitigate the issue by introducing ϵ-greedy policies, where the ϵ parameter denotes the probability of choosing an action at random instead of choosing the action that is associated with the highest Q-value:

Algorithm 2: Q-Learning Action Selection

With probability ϵ sample action a_t randomly
otherwise select $a_t \leftarrow \text{argmax}_{a_t}\, q(a_t, s_t)$

The ϵ value needs to be chosen carefully and will often decrease throughout the learning process under a previously chosen function of time, inducing a gradual shift in focus from exploration to exploitation.

Gradient-based methods, on the other hand, tackle the same issue by defining a stochastic policy and maintaining relative entropy with the addition of a penalty term in the objective function:

$$\mathcal{L}(\theta) = \mathbb{E}_t[\log \pi_\theta(a_t|s_t)\; G_t] + \beta \underbrace{\sum_a \pi(a|s_t) \log \pi(a|s_t)}_{\text{Entropy}} \tag{3.34}$$

Here, the entropy coefficient introduces additional bias, as it comes in the form of a hyperparameter β. However, the shift from exploration to exploitation is a more natural property of the above formulation. When the agent learns to maximize the objective function more reliably by earning rewards, it will naturally trade away entropy. These strategies compel the agent to sample the environment more generously by "keeping an open mind" and thus offset the eventual convergence of its

behavior. Another approach is to regulate the learning impact of each sample, that is, limit the amount to which an agent adapts its behavior with every single experience. Using a large number of samples embodies this strategy. However, Trust Region methods promise a similar effect while reducing the number of required samples and virtually eliminating gradient-methods' most significant drawback of sample inefficiency.

3.4.1 Trust-Region Policy Optimization

The objective function of advantage actor-critic policy gradient methods denotes a maximization of the probability of taking a specific action a_t under parameters θ, given state s_t or observation o_t when the associated advantage estimate is positive and vice-versa:

$$\theta^* = \underset{\theta}{\operatorname{argmax}}\, \mathbb{E}_t[\log \pi_\theta(a_t|s_t)\ A_t^\pi] \tag{3.35}$$

Notably, this function posits an unpractical numerical optimization problem, because unlike value-based methods, it does not truly optimize a function by finding global maxima or minima. This is due to the instability of a problem, where advantages will change based on policy and thus maximizing a single action probability at any given time will not necessarily increase the overall performance. In theory, value-based methods can use data no matter when the data was recorded, much like a supervised learning algorithm. To apply policy gradients effectively, one wants to prevent the actual convergence of the above expression, as optimizing it would always result in an action-probability of 1.0 if the advantage were positive, and 0.0 if it was negative. Trust Region methods and specifically *Trust-Region Policy Optimization* (TRPO) (Schulman et al., 2015), can mitigate this predicament. Instead of allowing unbounded gradient steps, this approach limits each update subject to a predetermined maximum change of the resulting distribution δ - in this case, this distribution is the policy, i.e., distribution over actions given the state or observation:

$$\begin{aligned} &\underset{\theta}{\text{maximize}}\ \mathbb{E}_t[\log \pi_\theta(a_t|s_t)\ A_t^\pi] \\ &\text{subject to}\ \mathbb{E}_t[D_{KL}[\pi_{\theta_{old}}(\cdot\,|s_t)||\pi_\theta(\cdot\,|s_t)]] \leq \delta \end{aligned} \tag{3.36}$$

The intuitive constraint over the change in distribution has very practical reasons. Specifically, this approach attempts to limit the change in parameters at each update step, but evaluating the change in the resulting distributions given a change in parameters is not always possible, especially when a complex nonlinear function approximator parameterizes the distribution, such as a neural network. The TRPO algorithm encapsulates a true optimization target, which is equivalent to policy gradients subject to a constraint of change in policy. Using the method of Lagrange-multipliers, the above formulation can be restated as a penalized "surrogate" objective, instead of an explicitly constrained objective function, making differentiation possible as follows:

$$\mathcal{L}(\theta) = \mathbb{E}_t \left[\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} A_t^\pi \right] - \beta \mathbb{E}_t [D_{KL}[\pi_{\theta_{old}}(\cdot|s_t)||\pi_\theta(\cdot|s_t)]] \tag{3.37}$$

Notably we here replace the log term with a ratio of policies under differing sets of parameters. By the chain rule, these terms are equivalent but the ratio has an important interpretation if numerator and denominator differ. This term then captures the change of the expectation conditioned on the change of the parameterized distributions. We will examine this interesting property in detail in the following section.

Optimizing the expression in Equation 3.37 above requires the computation of the second-order derivative of the KL-term, which is the Fisher information matrix. Intuitively, this provides the derivative with a measure of how much the D_{KL}-term changes with respect to the parameters θ. The expression is then maximized with the conjugate gradient method by approximating $\mathcal{L}(\theta)$ linearly and the KL-constraint quadratically as follows:

$$\begin{aligned} &\underset{\theta}{\text{maximize}}\ g * (\theta - \theta_{old}) - \frac{\beta}{2}(\theta - \theta_{old})^\top F(\theta - \theta_{old}) \\ &\text{where } g = \frac{\partial}{\partial \theta} L_{\pi_{\theta_{old}}}(\pi_\theta),\ F = \frac{\partial^2}{\partial^2 \theta} D_{KL\pi_{\theta_{old}}}(\pi_\theta) \end{aligned} \tag{3.38}$$

Despite this severely reductive approximation of the objective function, computing the Fisher information matrix F is computationally expensive and complicated to implement in practice. Schulman et al. further line out a number of issues with this approach. Crucially, TRPO does not perform well in experiments involving function approximation with CNNs and RNNs, two out of three of the most prevalent neural network paradigms.

3.4.2 Proximal Policy Optimization

Proximal Policy Optimization (PPO) is a state-of-the-art actor-critic policy gradient method that matches the performance of TRPO with only first-order derivatives in its update rule. It uses simple clipping to implement an equivalent, heuristic trust region approach to parameter optimization. Like most modern deep-RL methods, state-action-reward samples are gathered in N parallel environments, rolled out over T time-steps and for a total of K learning epochs. Concrete values for each of these hyperparameters will be provided in the experiments section.

Objective Function

The adapted surrogate objective function expresses a pessimistic bound on the policy parameter updates by clipping the ratio $r_t(\theta)$ to a predetermined ratio interval $[1 - \epsilon, 1 + \epsilon]$ as follows:

$$\begin{aligned} \mathcal{L}^{CLIP}(\theta) &= \mathbb{E}[\min(r_t(\theta)\ A_t^\pi, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\ A_t^\pi)] \\ \text{where } r_t(\theta) &= \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)},\ \text{such that } r_t(\theta_{old}) = 1 \end{aligned} \tag{3.39}$$

The motivation behind this objective is as follows. The first term inside the minimization corresponds to the unpenalized TRPO objective in Equation 3.37, while the second term is a clipped version of the same. The advantage estimate can take any form, however, Schulman et al. use the GAE as presented in the previous section. This objective reduces to much simpler terms when we frame it in light of the sign of the advantage estimate A_t^π. Therefore, Equation 3.39 can be restated as follows:

$$\begin{aligned} \mathcal{L}^{CLIP}(\theta) &= \mathbb{E}[\min(r_t(\theta)\ A_t^\pi, g(\epsilon, A_t^\pi))] \\ \text{where } g(\epsilon, A_t^\pi) &= \begin{cases} (1+\epsilon), & A_t^\pi \geq 0 \\ (1-\epsilon), & A_t^\pi < 0 \end{cases} \end{aligned} \tag{3.40}$$

such that **positive advantage** will be maximized using the following expression:

$$\begin{aligned} &\underset{\theta}{\text{maximize}}\ \min\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}, (1+\epsilon)\right)\ A_t^\pi \\ &\text{with } A_t^\pi \geq 0 \end{aligned} \tag{3.41}$$

Because the advantage is positive, increasing the action probability with parameter updates to θ, relative to its previous probability under θ_{old} will maximize the expression. In other words, the ratio will be larger than one. When this ratio reaches its ceiling, the $\min$ operation limits further increases, which means that the objective function can not be optimized further by increasing the respective action probability until a reinitialization occures, such that $\theta_{old} \leftarrow \theta$.

Correspondingly, **negative advantage** impacts the objective function in the following way:

$$\begin{aligned} &\underset{\theta}{\text{maximize}}\ \max\left(\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}, (1-\epsilon)\right)\ A_t^\pi \\ &\text{with } A_t^\pi < 0 \end{aligned} \tag{3.42}$$

Here, the opposite is true. If the advantage is negative, the optimization procedure is compelled to minimize the ratio of action probabilities in order to maximize the overall expression. Again, this process is limited by the clipping term, such that the policy under the updated parameters θ can not deviate too far from the policy under θ_{old}. The behavior of the objective as a function of ratio $r_t(\theta)$ and for the cases of positive and negative advantage are neatly illustrated in Figure 3.4.1, adapted from the original paper by Schulman et al.

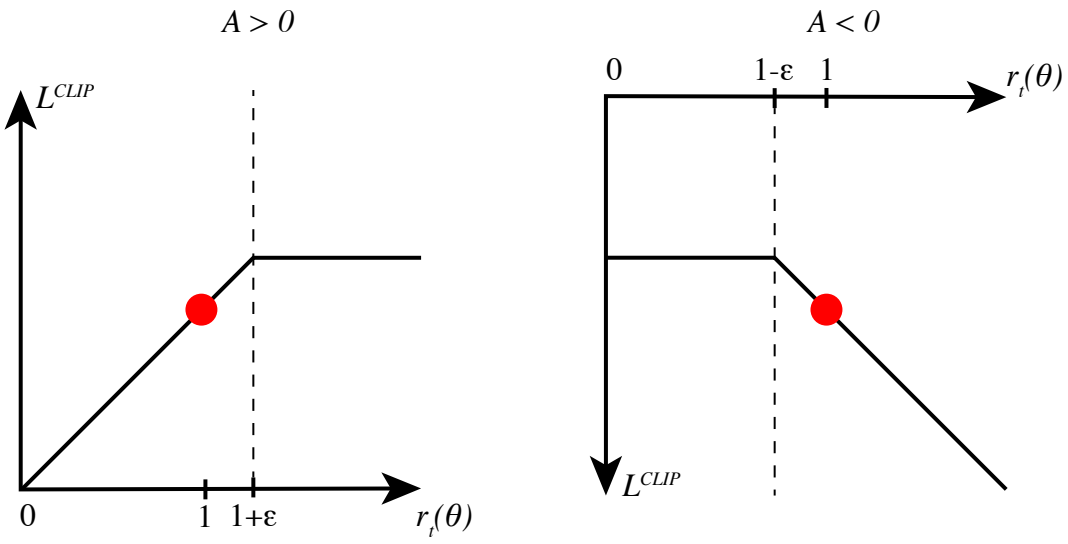


Figure 3.4.1: PPO clip objective

Implementation Details

The clipping introduced by PPO will, of course, require some additional processing steps, as the ratio in the expressions above will always be 1, if we assume that $\theta = \theta_{old}$ as we initiate the parameter update. In order to enforce the constraint, PPO thus takes multiple steps per update step, continuously updating θ and holding θ_{old} constant. This allows the ratio to change within a single update, and allows the policy's deviation from the original to be limited. Engstrom et al. correctly point out that this procedure will not impose a constraint on the first of the multiple update steps, as the new policy is initialized with the old policy parameters. Thus the first update is unbounded, and the limit only enforced on subsequent update steps. Despite this fault in the concept of PPO, the algorithm has proven successful across a large body of experimental settings. Citing theoretically possible stability issues in PPO, Wang et al. have introduced an explicit solution to the problem by providing a rollback option. This allows the algorithm to roll back the policy parameters after the multi-step update has concluded and the resulting ratio been deemed too large or small respectively.

In the present work, we have decided to exclude the rollback option, however, Engstrom et al. identify a number of implementation level details as crucial pre-requisites for the stable performance of PPO - most of which we consider in our implementation (refer to section C in the supplementary materials for a detailed list).

Algorithm

In conclusion, here follows the complete procedure of the PPO algorithm.

Algorithm 3: PPO Clip

Initialize policy θ_{old} and value ϕ_{old} parameters.
for *k = 1, 2, ...* **do**
  for *environment = 1, 2, ..., N* **do**
    Run policy $\pi_{\theta_{old}}$ for T time-steps to collect trajectories
    Compute advantage estimates $\hat{A}_t, ..., \hat{A}_T$ using current value function $V_{\phi_{old}}$
  end
  Optimize surrogate objectives **for** *d = 1, 2, ..., D* **do**
    (with minibatch sizes $M \leq NT$)
    $\theta_{d+1} \leftarrow \theta_d + \alpha \nabla \mathcal{L}^{CLIP}(\theta_d)$
    $\phi_{d+1} \leftarrow \phi_d - \alpha \nabla \mathcal{L}^{V}(\phi_d)$
  end
  $\theta_{old} \leftarrow \theta_D$
  $\phi_{old} \leftarrow \phi_D$
end

For each out of K epochs, the agent collects T time-steps of observation-action-reward tuples in N parallel environments. Correspondingly, PPO computes the generalized advantage estimates per time-step per environment as such:

$$\begin{aligned} \hat{A}_t &= \delta_t + (\gamma\lambda)\delta_{t+1} + ... + (\gamma\lambda)^{T-t+1}\delta_{T-1} \\ \text{where } \delta_t &= R_t + \gamma V(s_{t+1}) - V(s_t) \end{aligned} \tag{3.43}$$

Afterwards, the two surrogate objectives are optimized, either concurrently or separately depending on whether parameters are shared, using some method of gradient ascent. As mentioned previously, it is advisable to employ the Adam optimizer for PPO. For concurrent updates, the objective functions can easily be merged, for example:

$$\mathcal{L}^{PPO} = \hat{\mathbb{E}}[\mathcal{L}_t^{CLIP}(\theta) - c_1\mathcal{L}_t^{V}(\phi) + c_2\mathrm{H}(\pi_\theta)] \tag{3.44}$$

where c_1, c_2 are coefficients for the value function error and an entropy bonus H over the policy π_θ respectively. As described in the section on policy gradients, the entropy bonus helps prevent premature convergence on a sub-optimal policy. Only when the policy objective increases beyond the maximum entropy will the algorithm converge on action probabilities associated with reliably high returns.

Lastly, when the parameters θ and ϕ have run through all D current minibatches, the previous parameters will be updated with their values.

Chapter 4

Computational Theory to Artificial Life.

In this chapter we present the central mechanism of our attentional deep-RL approach. We translate a computational model of the interaction between attention and learning, originating in neuroscience, into an applied mechanism for deep-RL and demonstrate the equivalency of our formalism by constructive proof.

4.1 An Integrated Model of Attentional Learning

AWRL is the culminant computational model of a long line of research into the origins, effects, and benefits of attention in biological RL. The central question it attempts to answer is how attention and RL interact in biological organisms to implement efficient and flexible learning. Niv et al. critically examine the limitations of the computational RL framework by highlighting the "curse of dimensionality" in multidimensional environments, i.e., environments with a large variety of features per single observation. The number of available features in the observation space has a severe impact on the convergence rates and peak performances of traditional, or naïve, RL algorithms. This issue stems in part from the holistic treatment of observations in most RL algorithms; that is, observations are not explicitly deconstructed into their features but instead regarded as a whole. For value prediction, this means that each observation promotes an isolated judgment on its value with little to no relation to similar observations. While the application of deep neural networks, convolutional networks specifically, may somewhat mitigate the problem through learned non-local filters and translation into high-dimensional decision space, there is still no guarantee that similar input features will generate similar value predictions. In other words, there is no telling whether an RL algorithm indeed identifies task-relevant features or merely a highly specialized, value-predictive observation - thus sacrificing the potential for generalization across tasks and environments.

However, humans are remarkably efficient at identifying and generalizing based on task-relevant features in the abundantly featured natural world. With its selec-

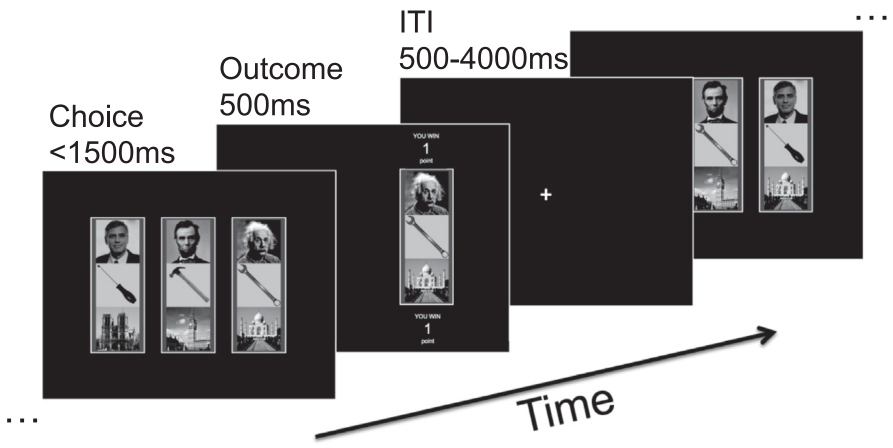


Figure 4.1.1: Schematic of the dimensions task as presented in (Leong et al., 2017). Participants are instructed to select one of the three presented stimuli. If the stimulus contains the target feature-value (specific face, landmark, or tool) the participant is awarded a point.

tive properties, the mechanism of attention is a prime candidate for the process of breaking down high-dimensional observations into lower-dimensional representations (Corbetta and Shulman, 2002). Niv et al. term these representations "task-state representations," implying several additional properties. (i) a set of feature-correlated weights implements task-state representations, (ii) they are task-specific (Bar-Gad et al., 2003), and (iii) dynamically adjustable as the agent shifts task-focus (Fabian Canas, 2010; Frank and Badre, 2012). The authors provide a simple example to illustrate these properties: The (human) agent stands on the side of the road, looking for a taxi-cab. As they observe the oncoming traffic, they will pay special attention to the color of passing cars (depending on the country, looking for, e.g., a bright-yellow vehicle). Once the agent has flagged down a cab, they might want to cross the road to reach it. The task has shifted, and now the agent pays attention to the speeds and locations of cars, disregarding their color entirely. If we refrain, for a moment, from considering the mental effort of integrating knowledge (about taxi colors, signaling conventions, and other factors) into the equation, we can see that the agent's applied task-state representations will translate to most any other location on the planet.

Physiologically, the inductive bias of perceiving the world as a collection of features is highly plausible. The visual system prominently implements a series of filters for orientation, shape, and color, whose compounds represent the objects in front of our eyes (Hubel and Wiesel, 1959). The same is true for the auditory system, where certain combinations of waveforms and frequencies align to evoke highly salient representations (Wang, 2018). With these considerations, Niv et al.; Leong et al. introduced the "dimensions task" - a simple problem to examine and compare subjects' decisions to a set of computational models, with and without attentional bias. The problem presents the participant with a selection of three stimuli, each of which comprises three stimulus dimensions, i.e., features (face, tool, and landmark

[1]) (see Figure 4.1.1). Only one feature expression, i.e., one feature value, is predictive of a higher reward while all others are irrelevant. Participants are instructed to choose the stimulus they believe contains the relevant feature value and will be rewarded an abstract point if they chose correctly. After 15-25 trials, the target feature expression will change randomly. Participants are notified of the switch but gain no further information about the new target feature. The authors compare a variety of computational models fitted to the results. Three of these models create a critical perimeter of possible and plausible mechanisms in the brain, from the naïve to the computationally optimal approach:

1. Naïve RL, as described above this model will treat every stimulus individually. With three features and three possible expressions per feature, this mechanism will need to learn a separate value prediction for 27 possible stimulus combinations using traditional TD- (or Rescorla-Wagner-) learning:

$$V^{new}(S_c) = V^{old}(S_c) + \alpha(R_t - V^{old}(S_c))$$

   Where the subscript c denotes the chosen stimulus.

2. Bayesian "optimal observer," this model maintains a probability distribution of rewards over individual stimulus features. Within a single trial set, the mechanism will integrate all previous choices and outcomes in order to update its beliefs. Note that this approach is not only statistically optimal but introduces the above-mentioned inductive bias of treating features in exclusivity and not holistically. There is no explicit learning rule, since evaluating the Bayesian beliefs incorporates the full trajectory of the trial run:

$$V(S_i) = p(R_t = 1|S_{chosen}, D_{1:t-1}) = \sum_{f \in S} p(R_t = 1|f = f^*, S_i)p(f = f^*|D_{1:t-1})$$

   where $D_{1:t-1}$ is the trajectory of previous stimuli and choices and f^* is the target feature dimension. The value function estimate then reduces to the probability of receiving a reward given the choice of stimulus S_i, and the trajectory so far.

3. f-RL (feature-RL), this model matches the naïve approach closely, with the exception that value prediction is not a function of the entire stimulus but a sum of predictions conditioned on individual features - again introducing the feature-based inductive bias. The value function is as follows:

$$V(S_i) = \sum_{f \in S} v(f)$$

   where $v(f)$ should be interpreted as a value function of features. The associated learning rule too is feature specific, but otherwise the same as traditional TD-learning.

Out of these three approaches, f-RL demonstrated the best fit to the behavioral data from the experiments, lending credence to the theory that humans base deci-

[1]This is the second iteration of the "dimensions task," introduced in (Leong et al., 2017) which has a much more intuitive structure than the original presented in (Niv et al., 2015)

sions on task-relevant features instead of whole stimuli. At the same time, the less optimal fit of the Bayesian model does not come as a surprise, since a fully optimal Bayesian treatment of input stimuli quickly becomes intractable as the number of feature dimensions increases. However, f-RL is only a precursor to the AWRL model, in that it forgoes an explicit attention mechanism.

In a successive publication, Leong et al. extend the model to expressly incorporate attentional focus from behavioral studies, thus introducing the attention-weighted RL framework. The novel element of this formulation is a feature-specific attention weight $\phi_t(f)$, such that the full AWRL framework is defined as:

Value function

$$V_t(S_i) = \sum_f \phi_t(f) v_t(f, S_i) \tag{4.1}$$

Reward prediction error

$$\delta_t = r_t - V_t(S_c) \tag{4.2}$$

Learning rule

$$v_{t+1}(f, S_c) \leftarrow v_t(f, S_c) + \alpha \left[\phi_t(f)\delta_t\right] \tag{4.3}$$

note a slight difference in notation to the f-RL model. The f variable now represents a feature index, instead of the actual feature content such that the attention weight is a function of that index, and the value function takes two arguments: the currently examined stimulus S_i and sub-selected feature f. The authors provide a more visual explanation of the model, seen in Figure 4.1.2, using the schematic style of the task description. This corresponds to an addressing mechanism, or, in established terms spatial attention in abstract space as opposed to feature- or content-based attention. With this definition, the framework aligns well with the feature-similarity gain model, as it implements a modulation of neural gain in the value function and learning rule.

In the original study, attention weights were derived from eye-tracking and fMRI data. The model exhibited an improved fit to data collected in the "dimensions task," leading the authors to conclude that human learners most likely apply attention weightings to a naïve RL approach instead of learning value properties of holistic stimuli, solve the task statistically optimally with a Bayesian approach, or employ hypothesis testing. The above expressions line out the bi-directional interaction of attention and RL. Equation 4.1 describes how attention constrains value prediction to selected stimuli, thus influencing the corresponding reward prediction error by possibly increasing the prediction's magnitude. Larger prediction errors lead to more extensive updates to the feature-specific value estimates, again weighted by their respective attention weights. Through this process, attention modulates what the agent learns about by guiding decisions. Conversely, learning about attended features may eventually outweigh the influence of attention bias if a given feature's value estimate is large enough. Hence the central hypothesis that attention constrains learning and learning shifts the focus of attention.

It is of note that the study did not propose a way of learning the attention function $\phi(f)$, and instead relied on eye-tracking and fMRI recordings to compute weights,

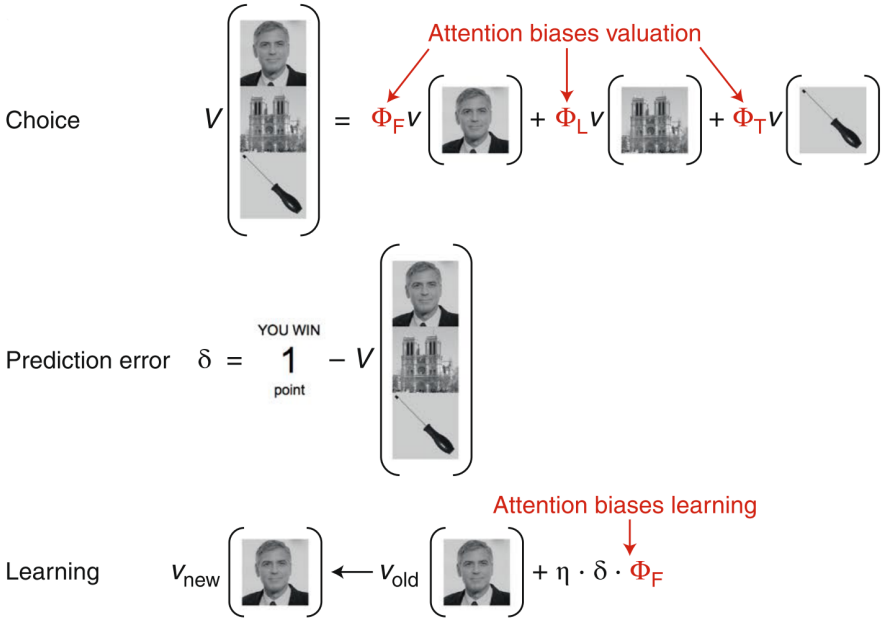


Figure 4.1.2: A visual description of the AWRL-framework as seen in Leong et al. (2017).

fitting no more than the learning rate to the experimental data. With this in mind, one might argue that the model has not fully demonstrated the latter element of the central finding: "learning shifts the focus of attention". Additionally, model comparisons showed that the introduced attentional bias in Equation 4.3 (attention at learning) has only negligible influence on the model's overall fit. This finding is quite intuitive from a calculus perspective since the value estimate of feature f, and, consequently, the reward prediction error, is already modulated by the respective attention weight in Equation 4.1. If one were to apply gradient descent to optimize the feature-based value estimate, the reward prediction error would serve as an attention-weighted objective function. Value estimates with larger associated attention weights would be optimized more readily than others. This perspective harbors a second major advantage, in that it allows a differentiation with respect to the attention function, so long as that function itself is differentiable. Lastly, the experiments featured a simple multi-armed bandit problem without possible future rewards in a single trial. There is little to no evidence that the model would function equally well in a dynamic environment.

4.2 The Self-Attention Mechanism

The popular *Dot-Product Attention* (DPA) mechanism serves as an excellent candidate for a differentiable weighting function. Initially, it was developed for applications in natural language processing in the "transformer" architecture, where it replaced more intricate recurrent models with more complex computation graphs (Bahdanau et al., 2015; Vaswani et al., 2017). The general processing is simple and geared towards sequential data (see Figure 4.2.1 for a graphical illustration of the mechanism). In the first step, DPA creates three vector representations $\boldsymbol{q}, \boldsymbol{k}, \boldsymbol{e} \in$

$\mathbb{R}^{1\times d_k}$, termed query, key, and embedding respectively [2], for each discrete token in a sequence $\mathbf{S} \in \mathbb{R}^{L\times d_{in}}$ of length L (such as a sentence of natural language with word embeddings) by applying three disparate parameterized, linear mappings $q(\cdot), k(\cdot), e(\cdot)$. Note that the representations of the complete sequence may also be represented in matrix-form, such that $q, k, e : \mathbf{S} \mapsto \mathbf{Q}, \mathbf{K}, \mathbf{E} \in \mathbb{R}^{L\times d_k}$. Importantly, the mappings are applied to each token individually, such that parameters are shared across the entire sequence, and there is no cross-influence of individual tokens at this stage. The second step prescribes how these matrices of vector representations are used to redefine each individual input token as a weighted sum of embedding vectors. This step introduces the possibility of selecting and deselecting input tokens for further processing and capturing the relationship between every available pairing of input tokens by summing up their embedding vectors. Queries and keys combine into weights by taking the dot-products of each possible pairing, such that for every sequence element $\mathbf{S}_{i,:}$, represented here as a tensor slice, there is a weight vector $\boldsymbol{a}_i$ of length L that captures the relationship of all key tokens to the ith query token:

$$\begin{aligned} \mathbf{A} &= \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right) \\ \text{s.t. } a_{ij} &\approx \langle q(\mathbf{S}_{i,:}), k(\mathbf{S}_{j,:})\rangle \\ &= \sum_{m}^{d_k} q(\mathbf{S}_{i,:})_m k(\mathbf{S}_{j,:})_m \end{aligned} \tag{4.4}$$

where $\mathbf{A} \in \mathbb{R}^{L\times L}$ is called an attention map. Once the attention weights are computed, they are used to generate an attention-weighted sum of embedded inputs $e(\mathbf{S}_{i,:})$ per input sequence element to form the output of the self-attention operation:

$$\begin{aligned} \hat{\mathbf{S}} &= \mathbf{A}\mathbf{E} \\ \text{s.t. } \hat{\mathbf{S}}_{i,:} &:= \sum_{j=1}^{L} a_{ij}\mathbf{E}_{j,:} \end{aligned} \tag{4.5}$$

These processing steps are straightforward and effective at capturing relationships between sequence tokens, but so far, they are agnostic towards the defining factor of a sequence: order. In a language context, the model would thus relate words to each other without considering their position inside the sentence. The standard solution to this problem is the introduction of a "positional encoding." Generally, this encoding is a function of the token index in the sequence and static during training. It produces a vector representation of the index $f : \mathbb{N} \mapsto \mathbb{R}^{d_{enc}}$ for effective processing in a neural network architecture. Examples include the sine-cosine encoding, which was introduced by Vaswani et al. (see Figure 4.2.2), and produces a vector of the following form:

[2] In the original publication, the embedding is called value as a reference to database terminology. The choice of the term "embedding", for the translated content of the feature vector, is to prevent confusions with value definitions in RL.

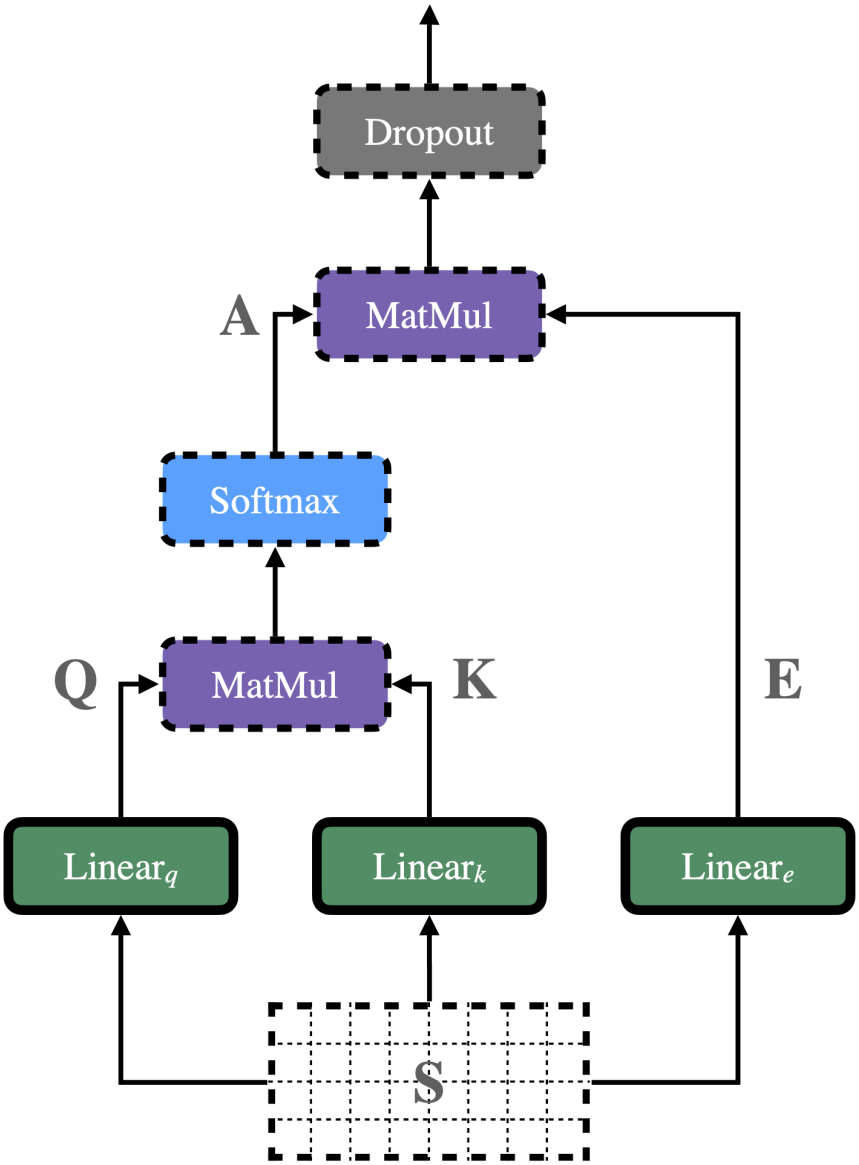


Figure 4.2.1: A graph representation of a single self-attention layer. The previous layer output (or stimulus input), represented by a sequence of vectors $\mathbf{S}$ is translated into three disparate matrices $\mathbf{Q}, \mathbf{K}, \mathbf{E}$. The pairwise dot-products of sequence elements $\mathbf{Q}_{i,:}$ and $\mathbf{K}_{j,:}$ correspond to the attention weight a_{ij}. The attention vectors $\mathbf{A}_{i,:}$ are softmaxed and subsequently multiplied with the sequence of embeddings to create one weighted sum of $\mathbf{E}$ per sequence element.

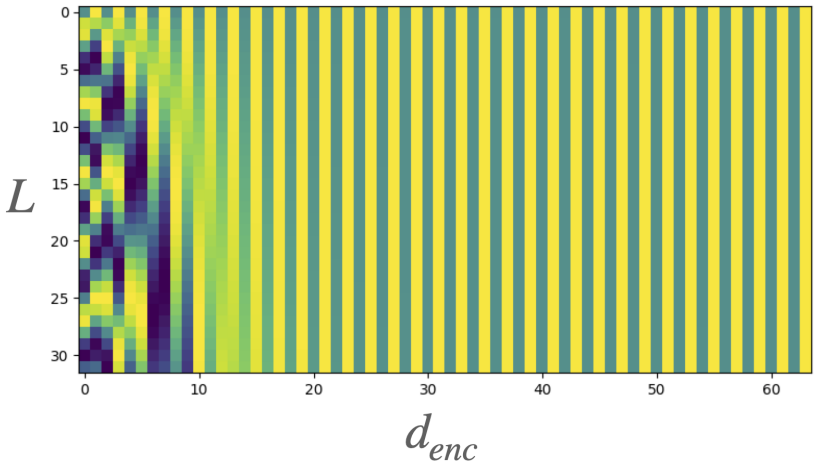


Figure 4.2.2: An example of the sine-cosine encoding, meant for a sequence of $L = 32$ tokens with an encoding dimension of $d_{enc} = 64$. Thus each row represents a single vector encoding of the respective index.

$$\boldsymbol{p}_i = \begin{bmatrix} \sin(\omega_1 i) \\ \cos(\omega_1 i) \\ \vdots \\ \sin(\omega_{d_{enc}} i) \\ \cos(\omega_{d_{enc}} i) \end{bmatrix} \tag{4.6}$$

$$\text{with } \omega_k = \frac{1}{10000^{2k/d_{enc}}}$$

Each of the positional encoding vectors is appended to their respective token, such that $\mathbf{S}^p \in \mathbb{R}^{L \times [d_{in} + d_{enc}]}$, before all other processing steps. These annotations are an absolute necessity for language applications - for our treatment of feature type observations, they open up a palette of different possibilities. For example, if we treat feature observations as sequences of feature embeddings, we might want to keep track of which feature goes where in the input volume, especially when a certain feature is located at the same index every time. Alternatively, the "positional" encoding may become a feature identifier in an unordered input sequence. Further details on the versatile uses of positional encodings will follow in the experiments section.

With all steps of the DPA mechanism defined, we will now turn towards another central contribution of Vaswani et al.'s "transformer" architecture, the employment of multiple attention heads in a single DPA layer, giving rise to the *Multi-head Dot-Product Attention* (MHDPA). This extension of the model is explained quite simply: instead of maintaining a single set of parameters for the linear transformations $q(\cdot), k(\cdot), e(\cdot)$, each layer holds h sets of weights per function, each of which process the input sequence in exclusivity. When each attention head has run its course, the outputs are concatenated along the feature axis, such that $\hat{\mathbf{S}}_{concat} \in R^{L \times [d_k \times h]}$. Another fully connected neural network layer then processes each sequence element

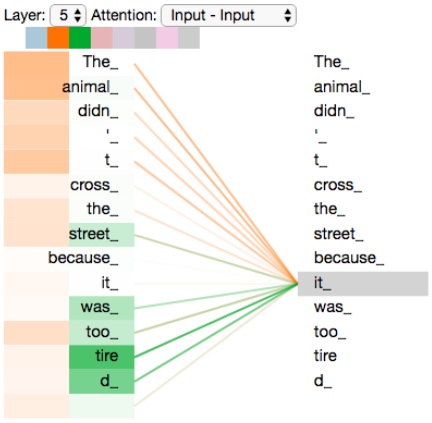


Figure 4.2.3: A visual example of multi-head self-attention in natural language processing (Alammar, 2018). Intuitively, words like the sentence subject refer to and are referred to multiple times, thus requiring exclusive connections among different sets of related word-tokens.

a final time, mapping each attended, re-weighted, and translated token as such $f : \mathbb{R}^{1\times[d_k\times h]} \mapsto \mathbb{R}^{1\times d_{out}}$. The intuition behind multiple parallel processing steps is that each of them can capture disparate relationships between sequence elements. In natural language, this ability will allow the model to track multiple references to the same (word-) token without conflating them (see Figure 4.2.3). For our application in feature space, we hope to see different compound features modeled in each of the attention heads - such as relating the positions of different game objects to each other or using relative distance as features for further processing.

4.3 Deep and Generalized AWRL

Now that we have defined both the theoretical foundation and a candidate mechanism for our attentive deep-RL agent, we will demonstrate that the MHDA paradigm, in fact, implements a generalized version of the computational AWRL model.

In deep-RL, we rarely compute feature based value function estimates as proposed in the AWRL framework in Equation 4.1, largely, because input stimuli (such as images) are not easily separable into distinct features. However, under the assumption that distinct features (represented by a matrix as seen above) are used for value estimation, we can represent the feature-based value estimate with a subset of neural network weights from a fully connected linear layer, such that:

$$\begin{aligned} v(i,\mathbf{S}) &= W_t^i \mathbf{S}_{i,:}^\top \\ \text{and } V_t(\mathbf{S}) &= W_t \mathbf{S}^\top = \sum_{i=1}^{L} v(i,\mathbf{S}) \end{aligned} \tag{4.7}$$

where W_t^i is the weight matrix solely associated with the ith input feature. Now,

we choose to represent the attention function $\phi(i)$, which maps the feature index to a scalar attention weight, with the formulation of self-attention:

$$\phi_t(i) \equiv q_t(i)k_t(i)^\top \tag{4.8}$$

So the value function estimate of the entire attention-weighted stimulus becomes:

$$V_t(\mathbf{S}) = \sum_{i=1}^{L} \langle q_t(i), k_t(i) \rangle W_t^i \mathbf{S}_{i,:}^\top \tag{4.9}$$

This equation is a special case of Equation 4.5, where only the diagonal elements of the attention matrix may take on nonzero values. Note that for this to have an effect, one must not apply a softmax activation as in Equation 4.4 but instead use the approximately equal inner product of query and key representations. Alternatively, the softmax may be applied across the diagonal of the attention map, instead of each row, to limit the magnitude of attention weights. The softmax is also a crucial element in translating the competitive nature of attention, theorized in various models, where a high attention value will actively suppress lower values. Through the addition of a temperature parameter κ_{ij} the competitive nature of the attention layer can be increased and decreased at will, allowing more or fewer features to be considered for further processing.

Having defined the classic case of AWRL in formal terms, we now go on to expand the model by exploring the full potential of the self-attention mechanism. Instead of selecting and weighing individual features in exclusivity, the self-attention mechanism introduces a relational inductive bias that allows the mixing of features. To explain the reasoning behind this, we expand on one of the examples provided by Niv: When we are crossing a road, the speed of a car may be only relevant if it is coming towards us, but not when it is moving away. Thus not only the individual feature (speed), but its relation to other features (e.g., direction) provides necessary information for decision-making. The same will be true for the tasks in our experiments. Thus we reformulate AWRL to allow mixtures of features as follows:

$$V_t(\mathbf{S}) = \underbrace{\sum_{i=1}^{L} q_t(i)k_t(i)^\top W_t^i e_t(\mathbf{S}_{i,:})^\top}_{\text{deep AWRL}} \Longrightarrow \underbrace{\sum_{i=1}^{L} W_t^i \left(\sum_{j=1}^{L} q_t(i)k_t(j)^\top e_t(\mathbf{S}_{j,:}) \right)}_{\text{generalized AWRL}} \tag{4.10}$$

In this formulation, mixtures, while possible, are not mandatory. Depending on the requirements of the environment, optimizing this value function estimate will select and/or relate features as necessary and is therefore a general expression of AWRL.

Notably, here we choose to map indices to attention weights, where in the introductory paragraph on self-attention, we chose the actual information content as arguments for the $q(\cdot)$ and $k(\cdot)$ functions. Both approaches have their advantages,

but the former is a more accurate translation of the AWRL model. Most applications of the self-attention mechanism apply a mixture of the two principles, in that the information content of a sequence element is extended with a spatial basis like the positional encodings described in the previous section. In the following chapter, we will explore a number of architectures and constellations to investigate the benefits of these approaches and model several computational theories of attention with the highest possible accuracy. All these models will implement generalized AWRL as their core selection mechanism presented here.

4.4 Reading (Self-)Attention Maps

The attention maps denoted as $\mathbf{A}$ in Figure 4.2.1 and Equation 4.5 correspond to the pairwise dot-products of $q_t(i)$ and $k_t(i)$. The result is a matrix denoting in each row by what mixture of features j, feature i will be represented. Given a reference feature-observation drawn from a moment in the game *Pong* seen in Figure 4.4.1, that will be used throughout the experiments section, these maps can be visualized simply as a 2d-grid of attention scores as seen in Figure 4.4.2.

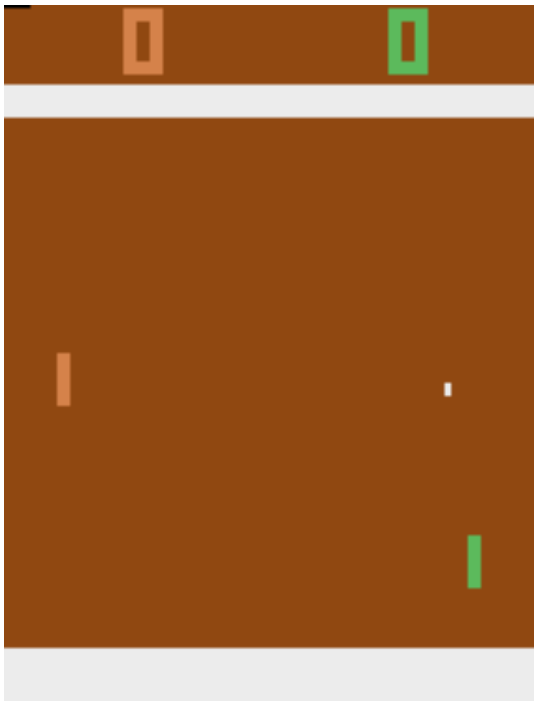


Figure 4.4.1: A reference frame of the game *Pong*, that will be used in the following experiments chapter to visualize attention maps. The ball is moving towards the player and very close to the border of the playing field. Generally, this is one of the most critical moments in the game, prompting an immediate response.

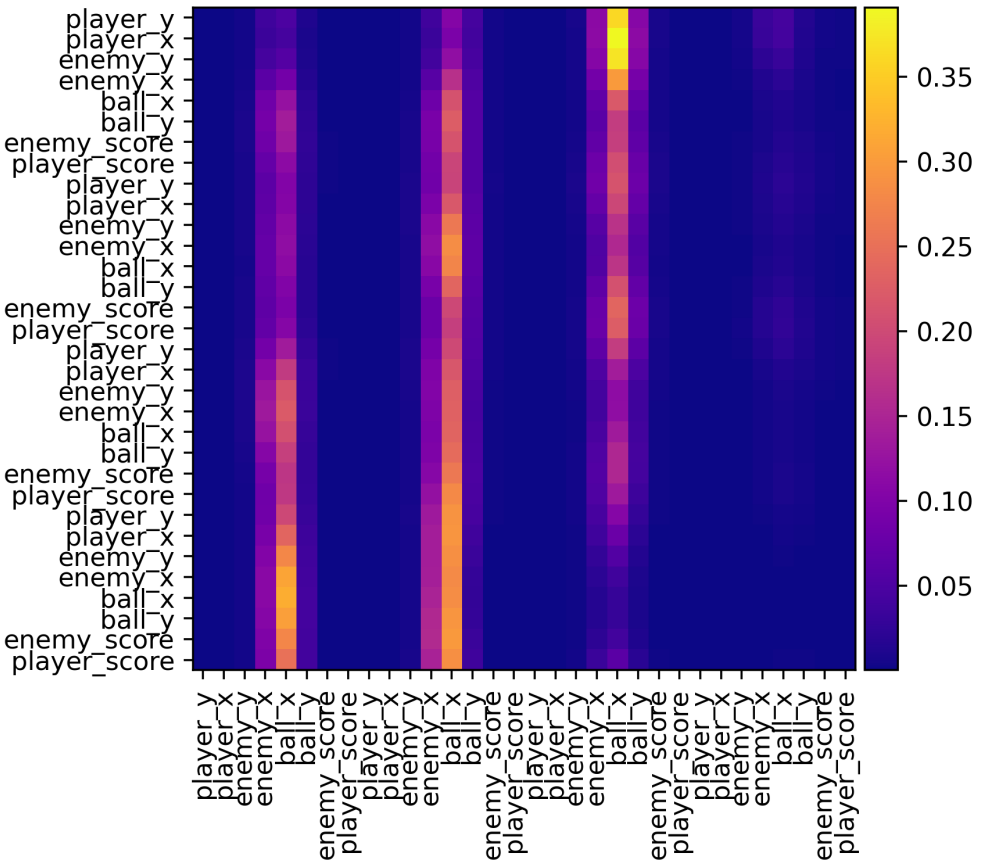


Figure 4.4.2: Example of a feature attention map for the above reference frame. This attention map originates in one of the multiple attention heads per self-attention layer. Rows comprise softmaxed distributions over relevant features that will be used in mixing to represent the feature at this particular row index. Additionally, attention weights are computed for a stack of 4 sequential feature observations, where the most recent set of observations is to the right and the oldest to the left. As an example: the 'player score' feature in the bottom row is represented by a mixture of 'ball x, enemy x' at $t-3$, 'ball x, enemy x' at $t-2$, and barely noticeably 'ball x' at $t-1$.

Chapter 5

Experiments

5.1 The Arcade Learning Environment

The *Arcade Learning Environment* (ALE) (Bellemare et al., 2013) is a software interface built on top of an Atari 2600 emulator. As such, the interface allows access to over a hundred Atari games for experiments in planning, learning, and decision-making by providing observations, rewards, and terminal states for each of the games. While these games can vary significantly in their challenges, they share a set of exploitable commonalities. For one, each game comprises a time- and action-discrete decision problem in that each frame allows a user-input selected from a limited set of button presses. ALE returns the numerical score gained in the last cycle, as well as a flag to indicate whether or not the game has finished. Observations are delivered in several ways. The traditional venue is visual, i.e., the environment will return a single frame of the game per time-step at a resolution of 210x160x3. Another type of observation is the RAM state, which must hold the entirety of the current game state in 1024 bits as per the design of the Atari gaming system. Depending on the game, this will include on-screen positions of game entities, scores, and environmental flags (e.g., some currently unobservable door has been opened in the current stage). Thus the interface allows researchers to run experiments in partially observable (image-based) and fully observable (RAM-based) Markov decision processes. Additionally, most Atari games' low visual fidelity and simple rule-sets provide an excellent sanity check for applied learning methods. Whether or not an artificial agent is performing well is immediately apparent upon visual examination of either gameplay or the trajectory of accumulated rewards over time. Lastly, the vast catalog of available games provides a gradient of complexity in the provided rule-sets. While many games take place on the same screen and with low variance in visual complexity (e.g., *Pong* or *Breakout*), some extend across multiple levels or stages, providing ever-changing visual layouts. Some games, such as *Montezuma's Revenge* and *Pitfall!* further demand multi-tasking by posing various challenges like jumping, collecting, or merely waiting to avoid traps in a 2d-game world.

ALE and its Python interface were widely established as a benchmark for deep-RL applications by the breakthrough success of the paper "Playing Atari with deep-

RL," which proposed an image-based deep Q-learning approach to play 55 select games at a human-, some even at a superhuman level (Mnih et al., 2013). This method computed Q-values using a convolutional neural network on RGB observations, i.e., raw sensory data. Most, if not all, deep-RL methods illustrated throughout this work have been tested on the ALE.

5.2 Feature Observations

The theoretical model introduced and translated in the previous section considers a set of discrete features during learning and decision-making. In other words, it presupposes a processing step that renders sensory stimuli (and, theoretically, memory contents) as neural representations of perceptible elements in the world. Therefore, the experiments described in this section will occur in feature-space instead of either image- or RAM-space. Here, "features" comprise a set of task-relevant entities derived from the RAM-state, such as positions of on-screen objects, scores, and numbers of resources. The notable difference between feature- and RAM-space is that the entities mentioned above are decoded and discretized into disparate values. This decoding step is taken care of by the *Atari Annotated RAM-Interface* (AtariARI) (Anand et al., 2019), which provides single-value translations of RAM-sub-states for a selection of 22 Atari games. Most developers of Atari games would use 4-8 contiguous bits to store such values, e.g. the position of the ball in *Breakout*, which follows naturally from the requirements to encode numbers as large as 192 (the vertical resolution of the Atari 2600). Of course, each set of feature-vectors contains only task-relevant features due to the significant memory constrains of the console. To demonstrate the selective prowess of an attention mechanism, we thus need to extend the observation space by a set of additional, task-irrelevant features - effectively mirroring the challenge of the dimensions-task in a dynamic, i.e., multi-step, decision-process. Therefore, each observation contains multiple sets of feature-vectors, where each additional set beyond the first originates in a parallel instance of the respective game that is not acted upon by the learning agent. Lastly, every single observation will contain a short trajectory of recent observations. Like the images returned by the ALE, the RAM-state and its decoded feature representation provide a set of static game variables, such as positions. However, a learning agent will need to consider dynamic values such as speed and direction to make sensible decisions. Mnih et al. used four stacked images to provide the learning algorithm with a way of deriving dynamics. The same is true for the following experiments, where four feature-observations are stacked to provide a sliding window across time.

As an example, the game *Pong* provides eight task-relevant features ('player y', 'player x', 'enemy y', 'enemy x', 'ball x', 'ball y', 'enemy score', 'player score'). Suppose we add one distractor environment and stack observations, as illustrated above. A single observation $\mathbf{S}$ then is of the dimensions $\mathbf{S} \in \mathbb{R}^{[8\times2\times4]\times d_{in}}$, where d_{in} is the dimensionality of a single feature-vector.

5.3 Model and Baseline Architectures

Having defined the shape and markup of the input space, we now introduce the model architecture as well as two baseline models that work without the inclusion of an explicit attention mechanism. The proposed d-AWRL and g-AWRL models (see Figure 5.3.1) comprise two sequential attention modules. Each module consists of a multi-head self-attention layer compiling four parallel heads, a linear up-scaling layer applied separately to each feature vector, and a final batch-norm layer. The intuition behind several sequential attention modules is the possibility for hierarchical processing. For example, the initial layer may attend to several positions over time, thus integrating speed and direction, while the second layer attends to the speeds of multiple objects to compute relative velocity. The number of four parallel attention heads is the result of a brief hyperparameter search that suggested no significant benefits by applying more (8-16) heads, but significant performance increases over models with fewer (1-2) heads. To reiterate, multiple attention heads are useful to track multiple diverse relationships between features in exclusivity. For example, one might want to capture the relative distances of object A and object B in one attention head, then, exclusively store the relative distances of object A and object B in another. The final representation of object A's position will then include both of these relational representations. The two attention layers are succeeded by a final, fully connected layer that feeds into policy- and value-outputs in the case of the g-AWRL and baseline models. The d-AWRL model, true to the original formulation in (Leong et al., 2017), will forgo this additional fully connected layer, such that value truly is computed on a feature-by-feature basis. By feeding the feature sequence directly into the value-estimation layer represented by a fully connected layer, the value estimates are calculated in exclusivity and subsequently summed up (see Equation 4.7). To ensure that no feature mixing is applied in each of the self-attention layers, a mask is applied to the attention maps, such that only values along the main diagonal are considered for feature representations. Notably, the vast majority of trainable parameters are shared between policy π_θ and value-function estimate V_ϕ, except for their final output step. This approach is relatively common in actor-critic models (Schulman et al., 2017; Mnih et al., 2013, 2016), and constitutes the basis of all proposed models and baselines. For additional hyperparameters of the attention-based models, see section D in the supplementary materials.

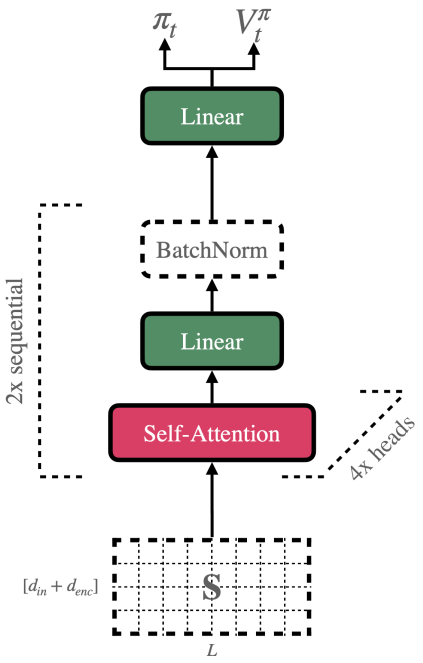


Figure 5.3.1: The neural network architecture used for both AWRL-based agents.

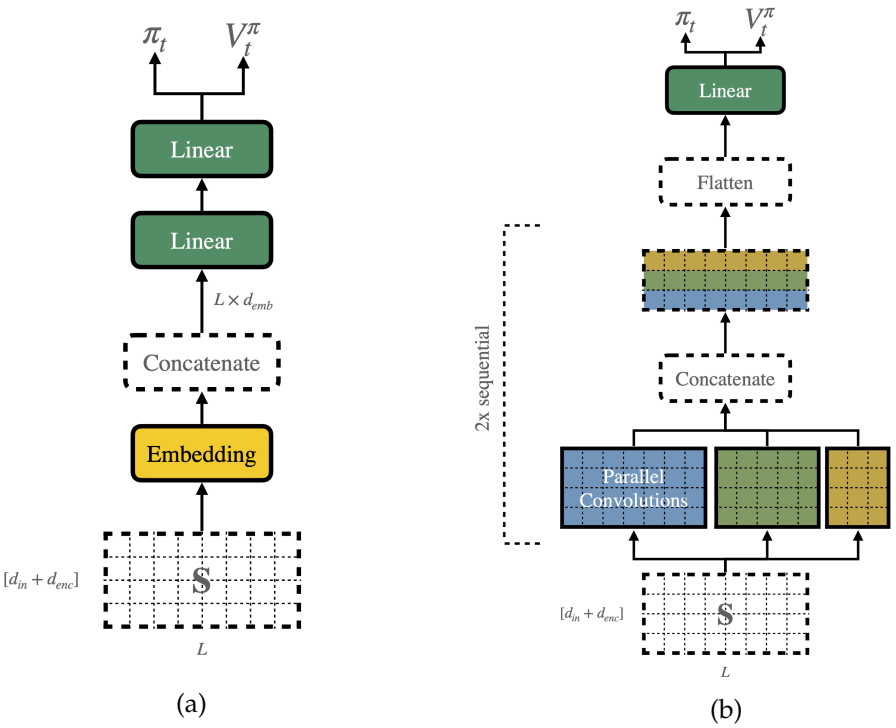


Figure 5.3.2: Neural network architectures for the fully-connected (a) and the two convolutional (b) baseline agents.

The attention models are posited against three baselines. The first baseline (FC) (see Figure 5.3.2 a) is a fully connected neural network, i.e., every vector element in the input volume is connected to each neuron in the first network layer, and so on. To make the model more competitive, we employ an initial feature-wise embedding layer that allows the network to mutate each feature-vector before processing the input volume in its entirety. Note that this step adds a notion of location invariance to this model that would otherwise be absent. The network has two sequential, fully connected layers after the embedding layer. The number of neurons per layer was chosen such that the number of trainable parameters is approximately equal to the attention models. However, it follows logically that the number of trainable parameters must increase quadratically as a function of the number of features as we add distractor elements to the input stimuli.

The second baseline (CNN) (see Figure 5.3.2 b) is a convolutional neural network of the style commonly applied to natural language tasks. Multiple parallel layers comprising filters of different sizes process the input volume (here, 3, 5, and 7), i.e., a matrix representing L features of dimensionality d_{in}. The feature maps created by each of these layers are then concatenated with each other along the channels dimension. In simple terms, this allows the convolutional layer to compute multi-step relationships between up to seven contiguous features at a time. The fully processed output at a single sequence index then includes multiple relationships along the channel-axis from a maximum distance of one to three neighbours. A subsequent layer will then be able to relate between three and seven one- to six-neighbour relationships at a time and so on. Unlike the fully connected baseline, the convolutional model is limited in its capabilities to relate spatially distant vec-

tors to each other by filter- and step-size. Specifically, this means that the same filter would never process the first and last feature in the observation simultaneously without further hierarchical processing. Therefore, we investigate a base- and advanced case of the convolutional baseline model. The base-case consists of a simple two-layer convolutional network with enough filters to match the attention models' number of parameters. The advanced case (a-CNN) constitutes the third baseline and is equipped with up to 20 sequential layers, depending on the number of features L in the observation space, so as to cover the full length of the input volume with a single filter.

5.4 Faithful and Generalized AWRL

The first round of experiments aims to investigate the potential benefits of computing the value function of compound features, as opposed to singular features. That is, comparing the fully faithful deep AWRL model to the expanded generalized AWRL approach. Section 4.3 illustrated this generalized case of the neuroscience-derived AWRL (Leong et al., 2017; Niv, 2019) model through the use of the original formulation of self-attention (Vaswani et al., 2017). We expect this mechanism to select relevant features from the input space and further combine them as needed to create improved task-state representations. As stated above, the d-AWRL model will instead compute the value function for each feature individually and then sum the results to arrive at a state-value estimate. The same will effectively be true for the pre-softmax logits of the policy output. We test both models on a selection of three Atari games. Each agent will train for $1e7$ steps and with individually initialized parameters for each new training run.

5.4.1 Task Description

The choice of games is based on their respective observation space and task complexity, with particular attention to the relevance of each feature in the input. For example, the game *Breakout* does provide a large observation space (35 individual features per time-step, stacked along four time-steps), but most features refer only to the location of one of the colored bars at the top. However, any agent can achieve a perfect *Breakout* score by merely keeping the ball from falling below the paddle, i.e., paying attention to ball and paddle positions. Thus most features provided by AtariARI are irrelevant, and both the observation space and task of relatively low complexity.

Hence the choice of games:

- **Pong** (low task, low observation complexity); *Pong* is a simple 2d table tennis game. The goal is to align the player paddle with the oncoming ball and hitting it past the opponent's paddle. If a player misses the ball, their opponent receives a point. The game ends when either player achieves 21 points or if the player drops five balls. As noted in a previous section, the observation state consists of the following features: ('player y', 'player x', 'enemy y', 'en-

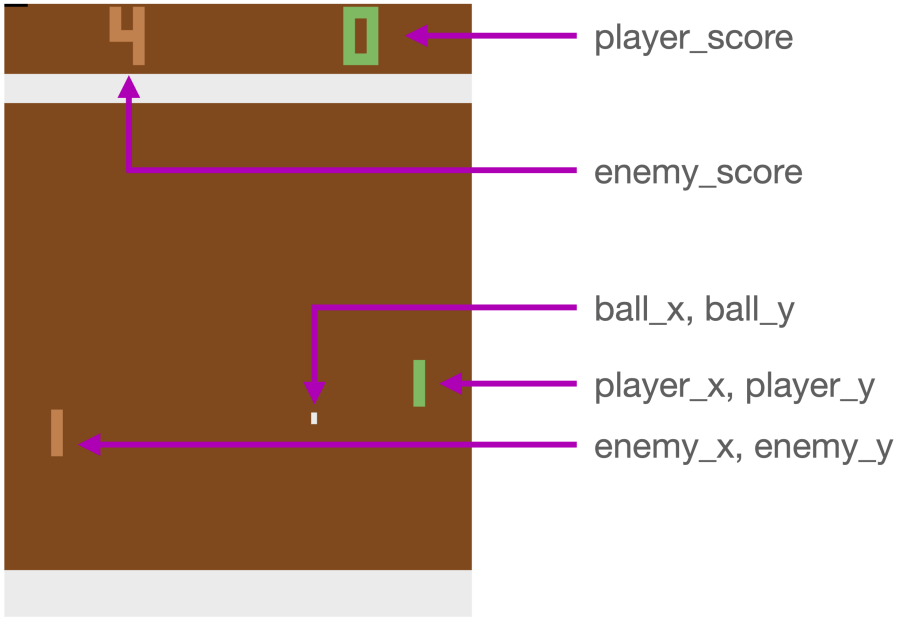


Figure 5.4.1: Feature observation in the game *Pong*. All features that are part of the RAM-derived observation are visible on-screen at all times.

emy x', 'ball x', 'ball y', 'enemy score', 'player score') – all of which have a significant impact on either policy or value function estimate.

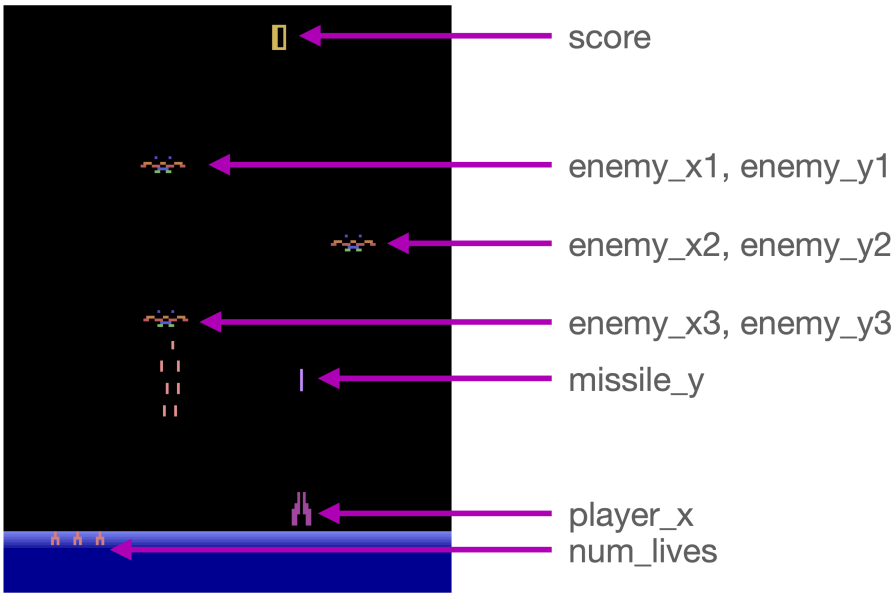


Figure 5.4.2: Feature observations in the game *DemonAttack*. Notably, the horizontal location of the missile fired by the player is not part of the feature observation.

- **DemonAttack** (medium observation, medium task complexity); *DemonAttack* is a comparatively simple shooting game, where the player needs to aim a spaceship horizontally at one of three randomly placed enemies. Enemies

fire shots at the player, so the task is to shoot them down by aligning the spaceship and pressing fire while avoiding enemy projectiles. The observation consists of ten features: ('player score', 'player x', 'enemy x1', 'enemy x2', 'enemy x3', 'missile y', 'enemy y1', 'enemy y2', 'enemy y3', 'num lives') Like in *Pong*, the agent needs to align itself with the another object. The task complexity is increased because of the additional challenge of avoiding enemy fire and selecting whichever enemy to engage next.

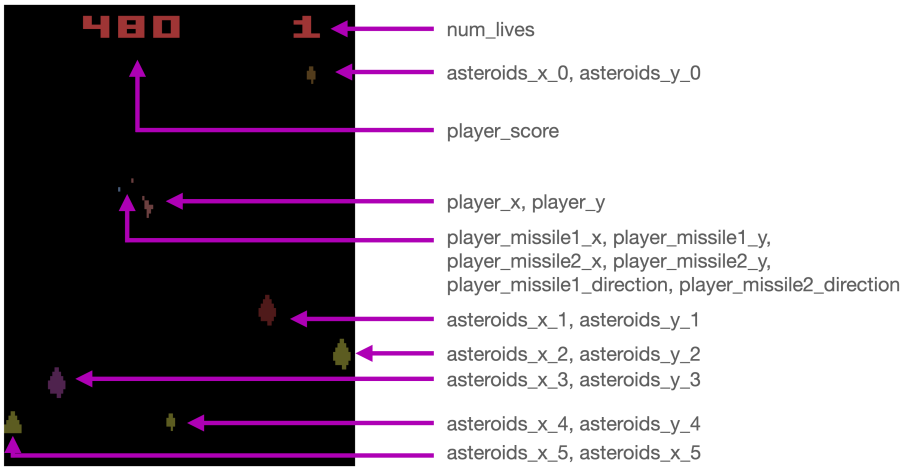


Figure 5.4.3: Feature observation in *Asteroids*. Only part of the available features are illustrated above since the RAM-state tracks up to 15 asteroid positions. Shooting an asteroid successfully will generate two new, and smaller asteroids each with their own features represented in RAM.

- **Asteroids** (high observation, very high task complexity); this game's task is to avoid and shoot several moving asteroids. The complexity arises from the possibility to steer the agent (in this case, a spaceship) in one of sixteen directions on a 2d map. Additionally, the space ship's speed must be adjusted manually and will not decrease to zero once the throttle button is no longer pressed. Each asteroid, once shot, will split into two smaller ones that travel along the same trajectory. When all asteroids are destroyed, the game resets the stage, spawning a greater number of large asteroids. The observation space includes the positions of up to 14 asteroids, resulting in a total number of 41 task-relevant features per time-step: ('player x', 'player y', 'num lives direction', 'player score high', 'player score low', 'player missile x1', 'player missile x2', 'player missile y1', 'player missile y2', 'player missile1 direction', 'player missile2 direction', 'enemy asteroids y0', 'enemy asteroids y1', 'enemy asteroids y2', 'enemy asteroids y3', 'enemy asteroids y4', 'enemy asteroids y5', 'enemy asteroids y6', 'enemy asteroids y7', 'enemy asteroids y8', 'enemy asteroids y9', 'enemy asteroids y10', 'enemy asteroids y11', 'enemy asteroids y12', 'enemy asteroids y13', 'enemy asteroids y14', 'enemy asteroids x0', 'enemy asteroids x1', 'enemy asteroids x2', 'enemy asteroids x3', 'enemy asteroids x4', 'enemy asteroids x5', 'enemy asteroids x6', 'enemy asteroids x7', 'enemy asteroids x8', 'enemy asteroids x9', 'enemy asteroids x10', 'enemy asteroids x11', 'enemy asteroids x12', 'enemy asteroids x13', 'enemy asteroids x14')

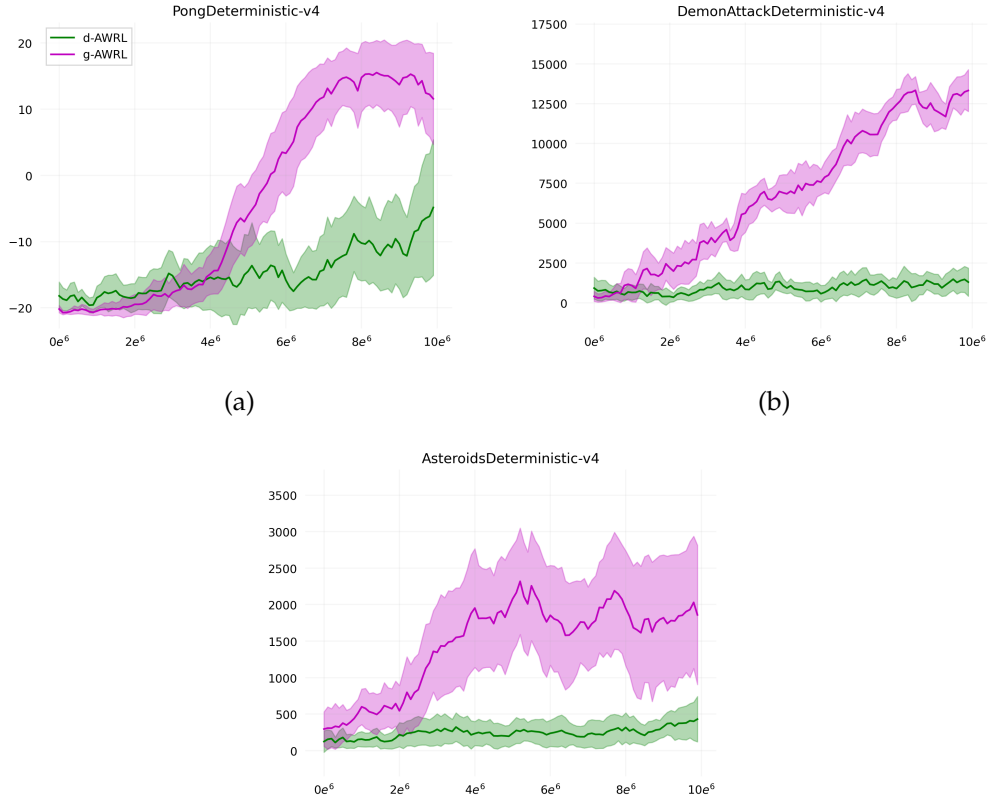


Figure 5.4.4: Learning curves for the faithful d-AWRL and the extended g-AWRL agents. In all but one game, the d-AWRL agent fails to demonstrate somewhat effective learning behavior.

5.4.2 Results

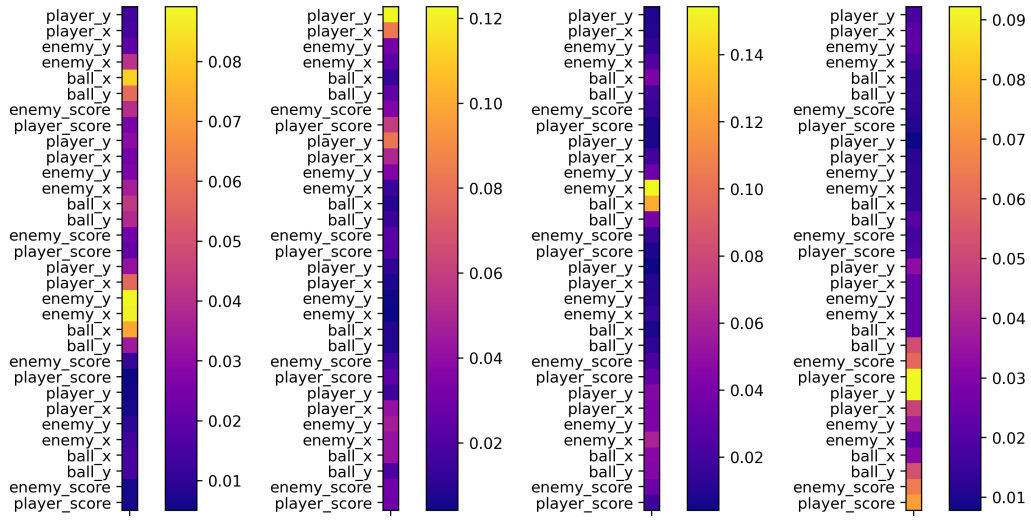
Results from ten individual runs per game suggest a clear benefit of relating features before computing their respective value contribution. The faithful d-AWRL model is unable to reach competitive performances in two out of three games, only achieving some improvement on the *Pong* task. Recall that this model is not relating features, but instead their respective value-predictions by summing them up. The same is true for the policy network, which effectively sums several logits, each based on a singular feature to arrive at a policy distribution. This process limits the model to capturing linear relationships between features, in addition to only one relational computation step as opposed to multiple hierarchical ones in the generalized model. The acceptable performance on the game *Pong* (see Figure 5.4.4 a) may be explained by its simple mechanics and generally low observation complexity since the game can be played with some success (winning a majority of matches) by a linear agent, minimizing the vertical distance between ball and player.

However, the other two games demand a higher fidelity of compound feature representations. In *DemonAttack*, (see Figure 5.4.4 b) the player needs to line up a spaceship, fire a shot, and observe whether the shot finds its target, all the while

avoiding enemy fire. Several parallel relationships need to be tracked in order to determine whether a reward will be received. For a simple example of this requirement, if the player agent is destroyed by enemy fire, any shot they gave off before being hit will pass through enemies. As a result, no points are received, so merely tracking the relative distance of one enemy and a fired bullet will not accurately predict value. Additionally, more than the vertical distance is required to confirm that a shot will hit its target. Unlike *Pong*, where the ball's horizontal position, for example, beyond the player or enemy paddle, will immediately determine reward (negative or positive, respectively), *DemonAttack* requires two-dimensional evaluations of position. Predictably, the d-AWRL falls short of finding representations of a high enough fidelity to derive accurate value-function estimates and policies.

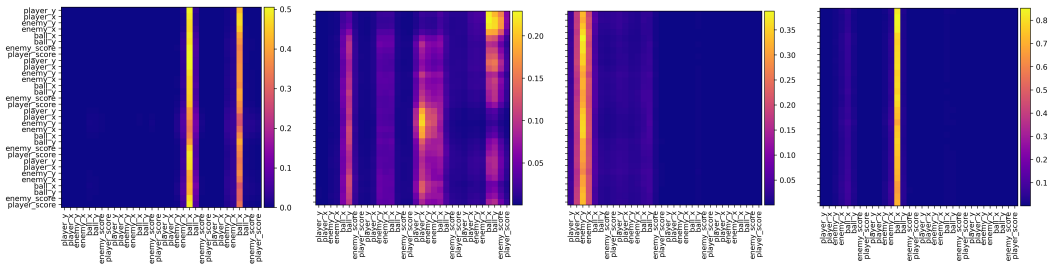
The same challenges are exacerbated in *Asteroids*, (see Figure 5.4.4 c) where the number of objects to be tracked increases significantly, and each of the objects can move in sixteen directions instead of two in *DemonAttack*. In this final and most complex game, multiple objects may approach the player position from different angles. For this reason, planning and executing an evasive maneuver requires the consideration of multiple (relative) velocities and directions, compound features in and of themselves, beyond simple relative positions. This task is so demanding that modern convolutional approaches have trouble matching human-level performance and thus makes for an excellent upper bound on the capabilities of both our proposed models. Again, the d-AWRL falls predictably short of displaying significant learning behavior. While the g-AWRL does not reach human-level top scores, it surpasses traditional convolutional approaches from previous works by a small margin.

The associated attention maps' patterns illustrate that both models seem to converge on highlighting the most task-relevant features. Of course, the d-AWRL agent is limited to selection only, yet it still identifies most of the features that explain the variance in the environment's value-function at any given time. The g-AWRL model, on the other hand, has the ability to either select or mix learned feature representations to find value-predictive task-states. Thus, the model demonstrates that some tasks demand feature compilation to translate into lower-dimensional task-state representations through emergent behavior that follows the value-function optimization process. Indeed, by examining the attention maps of the g-AWRL, we find that feature compilation does occur. Two particular mixing patterns can be found in all game settings: (i) mixing the same feature across several time-steps (suggesting changes in position, score, or other dynamic values) (e.g., Figure 5.4.6 a, where 'ball x' is integrated over two time steps) and (ii) modal mixing of features in a single time-step (that is, mixing features of the same modality like the x-coordinates of two separate objects) (e.g., Figure 5.4.6 b, where 'ball x' is mixed with 'player x'). The first style of compound features is ubiquitous in all attention-layers and heads. More so than relating different aspects of the observation, this behavior can be interpreted as time integration to formulate new low-level features such as speed and direction. The second mixing pattern occurs equally frequently, but it rarely exhibits the same amount of selective magnitude, which means that those features selected for compilation are rarely selected as sparsely (with as large of an attention weight) as in the first mixing pattern. This behavior may suggest that, while feature compilation is necessary to achieve peak performances, selection is the far more common mode attention operates on for the generation of task-state



(a) (b) (c) (d)

Figure 5.4.5: **d-AWRL** attention maps of the four heads in the first self-attention layer. This agent is purely selective, thus only a single weight is associated with each of the input features - true to the original formulation in neuroscience literature. Unlike the g-AWRL agent, which consistently discards information about player scores, d-AWRL highlights them consistently. Without a notion of relating positions to each other, this may be the safest way of predicting value in the *Pong* environment.



(a) (b) (c) (d)

Figure 5.4.6: **g-AWRL** attention maps in the first self-attention layer. Despite the added ability to relate features, the agent has learned to use mixing sparsely (e.g., in heads a and b), and highlighting particularly relevant features exclusively (e.g., in heads c and d).

representations.

Lastly, we find that the self-attention based agents tend to discard a large number of input features by choosing to represent them as the same feature multiple times. For example, the attention head in Figure 5.4.6 (d) elects to represent every input feature as the 'ball x'-position almost exclusively.

5.5 g-AWRL and Baselines

The second round of experiments examines whether the application of attention mechanisms indeed harbors algorithmic benefits for a deep-RL agent. Here, the g-AWRL agent competes against the baselines introduced in the architecture section 5.3. As mentioned previously, most features provided by the AtariARI interface can be considered task-relevant. Hence the introduction of distractor environments (see section 5.2), parallel instances of each respective game whose feature observations are appended to the observation passed on to the agent. The agent interacts with only one of these environments and has to identify the correct set of corresponding features. Each of the models should, in theory, be able to solve the task condition, even as the number of observation components increases drastically.

5.5.1 Task Description

The selection of Atari games remains the same as in the previous experiment. Additionally, we extend the task-set by a low-noise and a high-noise version of each of the games. The low-noise condition adds only a single distractor environment. Thus the number of features in each observation is doubled. In the high-noise version, a total of three distractor environments are added - quadrupling the size of the observation per time-step. At the outset of each training run, the sequence of features in the observation window is randomized to minimize correlations between spatial and associative relationships. Each training run allows $1e^7$ interactions with the environment, and consists of as many time-steps. The agents train on each environment and condition separately and with newly initialized parameters. The number of features in observation space is charted in the following table:

Task	L features	$2L$	$4L$
Pong	8	16	32
DemonAttack	10	20	40
Asteroids	41	82	164

5.5.2 Results

The learning curves illustrate a notable advantage for the attention agent in high-noise environments. However, the a-CNN agent is a close match in terms of final

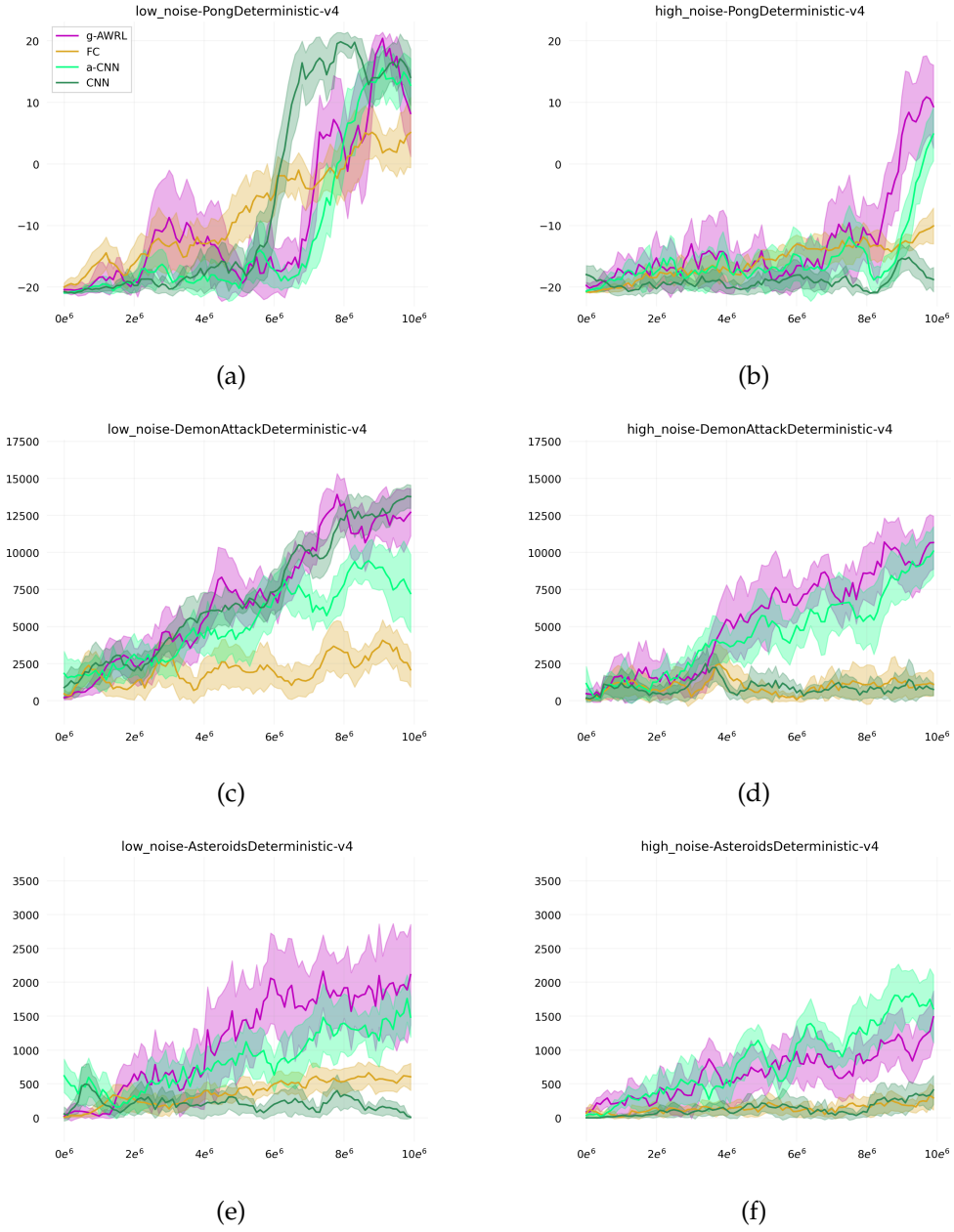


Figure 5.5.1: **(Left)** learning curves for all agents in the low-noise setting (single distractor environment) **(Right)** learning curves for the high-noise setting (three distractor environments). The g-AWRL agent performs well in all tasks - in most cases slightly above the a-CNN agent, despite a much more shallow neural network architecture.

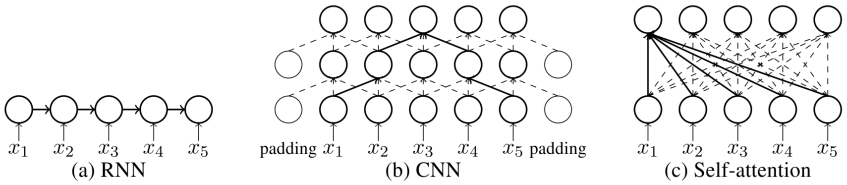


Figure 5.5.2: Locality in computation graphs of popular natural language translation neural networks (Tang et al., 2020).

performance, and in some cases, even outranks the attention agent in convergence speed. The base-case CNN model's top-performances then round out support for the notion that some form of spatial invariance contributes significantly to performance in feature space. The idea is intuitive, given that the fully connected baseline model maintains a large number of weights exclusively processing task-irrelevant features, presenting an ineffective architecture for the data.

Except for the *Asteroids* task, the basic CNN agent demonstrates stable improvements in low-noise environments, likely due to its computation graph's low complexity. Another compelling explanation is the proximity of task-relevant features within the window of observation. At a low number of components in the observation, each convolutional filter will create a mixed representation of almost the entire width of the observation (along the feature index axis). Effectively, each filter acts as one attention head, albeit with a far less complex computation graph, a simplified output sequence, and a lower number of parameters to train. As the number of features in the observation window increases, these benefits begin to dwindle since single convolutional filters will be less likely to capture multiple task-relevant and relatable features in a single operation. This issue does not affect the g-AWRL agent, which will consider all combinations of features locally and non-locally as training commences, depending on the query and key mappings' initialization. This quality is a hallmark of the self-attention mechanism, and one of the premier reasons for its popularity in neural machine translation. Traditionally, a target sentence will be encoded sequentially by a recurrent neural network with an internal state, or memory. Depending on the representative quality of the internal state, some words may be dropped as the network moves over the whole sequence, meaning that essential information could quite literally be lost in translation. The non-locality of self-attention networks like the Transformer ensures that every word token is considered at every translation step, i.e., every input token is connected to every output token in the computation graph. CNNs are theoretically able to implement such non-local processing as well, depending on the size of the applied kernels. With smaller kernels, several hierarchical layers are necessary to span the whole input sequence as mentioned in the previous section of employed architectures (Cordonnier et al., 2019). For an illustration of this concept see Figure 5.5.2.

5.6 Exclusive Spatial- and Content-based Attention

This final round of experiments seeks to illustrate the differences between conditioning attention exclusively on either spatial addresses or feature content. Spatial addresses here refer to a positional encoding of the sine-cosine variety that serves as a neural-network-interpretable positional index for every feature vector. While features are randomly sorted at every initialization of a learning environment, their position in the observation space does not change throughout the training run. The intuition behind these experimental conditions is as follows: the agent will attempt to approximate the environment's value-function. In the spatial attention case, the agent is forced to select those feature channels that may hold task-relevant information using address-based attention. However, the observed information within the selected channels may not be relevant at any given moment. This process corresponds to a crude interpretation of feature search. Intelligent agents seek out features using top-down attention mechanisms that modulate the response strength of those neurons that represent the sought-after feature (Treue and Martínez Trujillo, 1999). For example, if an intelligent agent searches for a red ball, the neurons expressing the color red and a round shape will activate more strongly if and only if these features are detected. The attention itself is entirely separated from the actual content of the observation. In the content-based case, on the other hand, attention is conditioned exclusively on currently observable features in the observation. This type of attention corresponds to stimulus-driven, i.e., bottom-up processes, where a particular observation will draw the attentional spotlight. As mentioned in the introductory sections, bottom-up attention is a necessary mechanism in biological intelligence that allows agents to react to essential stimuli on the fly, without engaging in a specific task (e.g., responding to the presence of a predator or an unexpected food-source).

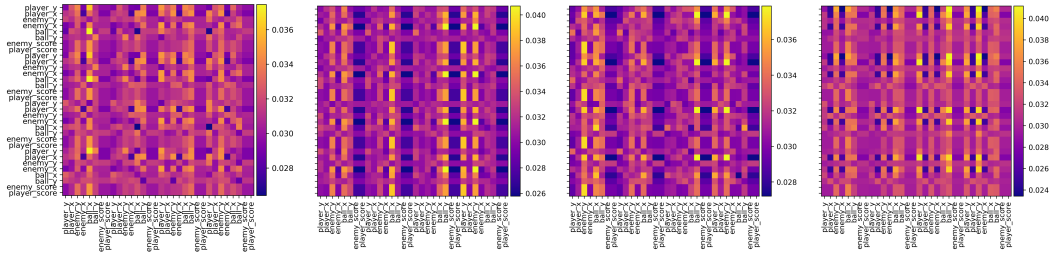
5.6.1 Task Description

As before, each agent learns for $1e7$ steps in the environment. Here, both models compete in only one Atari game, as the interest lies in the attention patterns exhibited after training, more so than the general performance across several tasks. Both models are comprised of the g-AWRL architecture presented previously. The observations received from the environment are static for the computation of attention maps in the address-based case and comprised of feature values in the content-based case. To clarify, the address-based agent will still receive the contents of selected features for later processing steps, and merely the selection process is limited to positional encodings.

5.6.2 Results

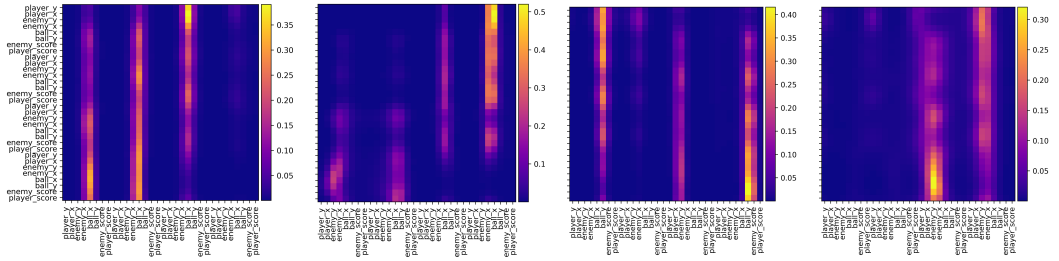
As expected, attention patterns vary significantly between purely content- vs. purely address-based attention. In the content-based case (Figure 5.6.1), attention patterns fluctuate between observations. In line with the concept of "tripwires" (Mott et al., 2019), we find that particular feature ranges trigger larger attention scores in a

bottom-up fashion. For example, in "*Pong*," the ball position will not be attended to by the agent unless it reaches a distinct set of values - specifically, positions in close proximity to the player and enemy agents. Recall that the attention score, in this case, is computed by the dot products of vector embeddings of that particular position. In other words, the network adapts to finding specific feature expressions, or values, that correspond to better explanations of variance in the value-function.



(a) (b) (c) (d)

Figure 5.6.1: **Content-based** attention maps in the first self-attention layer of the g-AWRL agent. These maps exhibit much higher entropy and capture feature expressions exclusively. This leads to possible confusions since, e.g., the position of a game object can be represented by the same neural code as the color of another object. This attention mapping will grade both features equally.



(a) (b) (c) (d)

Figure 5.6.2: **Spatial** attention maps for the same agent demonstrate a much sparser selection of features. Note, that the agent has learned to represent multiple input features with mixtures of the same 2-3 features, albeit with slightly different weightings. For example (a), a large number of features are represented with mixtures of 'ball x' across $t-3$ and $t-2$.

Conversely, in the address-based attention agent (Figure 5.6.2), patterns do not fluctuate at all after training. The agent has no perceivable notion of either time or the current observation until the attention-weighted observation enters the value-function- and policy-networks. Attention is fully conditioned on task-learning. Thus, the agent is continuously forced to highlight those feature channels that may or may not hold task-relevant information. The corresponding attention patterns are close to static, highlighting these channels. Different attention heads converge to select and combine specific features like player position over time. In terms of performance, we find that the content-based agent tends to reach higher scores much faster but fails to converge on a stable policy. Instead, performances continue to fluctuate heavily between top-benchmark scores and random-action scores at all

stages of training. One particular issue of purely content-based attention, in this case, is that the agent has no broader understanding of the observations it makes. If, for example, a meteorite approaches the player from the top-right, this specific (relative) position will be noted as influencing future value if the ship is destroyed. This observation results in a small update to the value-function-estimate. However, if the same meteorite were to approach the ship from the bottom-left, the feature would need to be examined without any transferrable knowledge. I.e., the agent has no concept of the meteorite and its position, but instead of meteorites in the top-right position. Due to the simplicity of our model, this generalization can not be abused since individual meteorite positions are treated as unique features in the observation, leading to the requirement of learning about each of them and each of their possible values in exclusivity. The address-based attention agent's learning curve does not approach top-scores at all, but it remains comparatively stable throughout the learning phase. The agent appears to identify the correct task-relevant features and tracks them continuously. However, much of the heavy-lifting in terms of breaking down large observations into simpler task-state-spaces is handled by the final fully connected layer instead of the attention heads. This process is reasonably consistent with the neuroscientific AWRL model since the model does not explain temporal dynamics, with some features being task-relevant only some of the time. If task-state representations as imagined by Leong et al.; Niv et al. truly only highlight a set of features for a given amount of time (regardless of their content), then the address-based model can be understood as the most faithful translation.

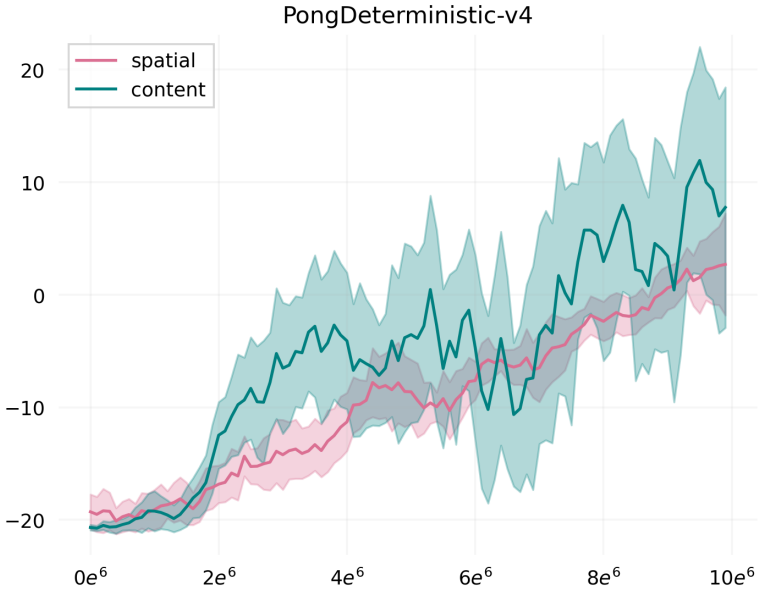


Figure 5.6.3: Learning curves for the exclusively spatial- and content-based agents.

Chapter 6

Conclusion

This work presents an empirical account of applying the theoretical *Attention-Weighted Reinforcement Learning* (AWRL) framework, a model of feature-based attention in biological RL, in canonical deep-RL tasks. The framework was first translated into a trainable neural network architecture, using the ubiquitous self-attention mechanism popularized in natural language processing tasks. Taking the workings of this attention mechanism and contemporary neuroscience literature as inspiration, the original AWRL model was extended to include the ability of mixing/compiling feature representations to integrate features over time and across modalities. The notion that biological intelligence uses combinations of features to represent complex stimuli as lower-dimensional task-state representations is well-founded in neuroscience literature. AWRL illustrates a mathematically sound way of how organisms select multimodal stimulus dimensions to form such representations in concrete task-settings. This process is necessary since most naturally occurring tasks faced by living organisms present their relevant features in a sea of equally salient noise. However, the original AWRL mechanism only provides a linear way of combining features. In this sense, the proposed generalized AWRL approach offers a more complete solution to the *Binding Problem* - a formulation of the challenging task to select and relate stimuli, knowledge, and memory to represent a current experience neurally. Our approach introduces a hierarchical processing step and nonlinear and nonlocal relational processing to solve the binding problem.

The applied d-AWRL and g-AWRL models were trained using state-of-the-art policy gradient methods, specifically, an updated version of PPO that implements several recent improvements. Task-state representations correspond to neural codes that allow for accurate value-prediction, conditioned on a given task, in the original AWRL model. The same is true for our applied version, where feature-specific attention weights are made differentiable with respect to value-prediction errors.

6.1 Discussion

In a series of experiments, the applied models were tested to answer three specific questions: **(i)** does the faithful d-AWRL model perform as theorized in the neuroscience literature, and can the extended g-AWRL approach improve it? (see section 5.4) **(ii)** Does the better performer of the two methods measure up to established feature-based models in the same learning regime? (see section 5.5) **(iii)** What are the effects of conditioning attention on feature-channel or feature-content alone (corresponding to a naïve interpretation of top-down and bottom-up attention, respectively)? (see section 5.6)

(i) The results of the first experiment indicate that a more comprehensive approach to solving the binding problem outperforms the originally theorized and purely selective AWRL model. Feature compilation, while not as prominent in the trained attention agents' behavior, is paramount to solving complex tasks such as Atari video games. The emergent dynamics of the attention mechanism suggest that agents integrate features over time, as well as modal features that provide information such as relative distances between objects.

(ii) The second experiment demonstrates that a self-attention based agent has a notable advantage over comparable models in settings with many noisy features per observation, exhibiting greater performance retention as the number of distractor features increases. Notably, these environments come much closer to the challenges faced by biological intelligence, as they are rich with salient, but task-irrelevant stimuli. The performance retention is most likely the result of nonlocal processing, which allows the AWRL agent to consider every possible combination of features for its task-state representations. At the same time, more traditional models like the fully-connected- and both convolutional baselines struggle to capture larger observation spaces concisely. Meanwhile, the g-AWRL collapses the space of possible feature combinations effectively by optimizing its value-function estimate with respect to its attention weights.

(iii) The attention agents used feature observations throughout all prior experiments where each feature was concatenated with a positional encoding. This vectorized positional index serves as an identifier of the feature, provided that the component occurs in the same place across all observations. One possible interpretation that relates closely to neuroscience is that this embedding holds information on which neuron or cluster of neurons represents a particular feature. In the final experiment, we examined differences in attention patterns when the attention scores were conditioned on only this identifier or only on the actual feature content. Results indicate that a combination of both is undoubtedly superior. The content-based attention agent did reach top-scores in the game "*Pong*," but it failed to converge on a stable peak-performing policy. Meanwhile, the purely address-based attention agent could not reach top-scores at all, albeit exhibiting a much more stable learning trajectory.

6.2 Future Work

With the initial success of the presented experiments and the immense popularity of self-attention, attention in general, and the advent of neuroscience-inspired artificial intelligence, many venues remain to be explored. Here, in the interest of brevity, there is a selection of enhancements and other possible approaches that pertain directly to the presented thesis work. Apart from these condensed sections, there is a vast body of research on improving the computational methods applied here. Some examples include using linearly complex self-attention methods (Wang et al., 2020; Shen et al., 2019), different forms of RL, specifically off-policy methods that would allow active-sampling and prioritized experience replay (Schaul et al., 2016; Mattar and Daw, 2018) for the integration of the Pearce-Hall and Mackintosh models of attention learning (Mackintosh, 1975; Pearce and Hall, 1980).

6.2.1 Focus Control with Softmax Temperature

The softmax activation function is an integral part of the self-attention mechanism. To reiterate, it turns the logits, representing how much each sequence element is related to the current query element, into a distribution. Recall the computation of the attention map, containing a full distribution over sequence elements per sequence element:

$$\mathbf{A} = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right) \tag{6.1}$$

It is common to introduce the concept of temperature to the softmax function, where the temperature parameter T governs the entropy of the resulting distribution. The parameter factors into the function as follows:

$$y_j = \frac{e^{\frac{a_j}{T}}}{\sum_{i=1}^{M} e^{\frac{a_i}{T}}} \tag{6.2}$$

Thus, higher temperatures will increase the entropy of the output variable while lower temperatures enforce a "sparse" distribution, weighted toward one specific feature. In RL, this principle often finds application as a control mechanism of how exploratory/exploitative the agent's policy is. In the above application of self-attention as a feature selector, lower temperatures would enforce sparser feature selection, possibly resulting in attention heads that carry only a single feature representation. The effects on learning of sparse and dense heads need to be researched in future works, as sparser heads might lead to more compact task-state representations and even better interpretability of the model behavior. Unlike the explicit comparison of single-selection vs. feature-mixing in this paper, reducing the temperature parameter will allow the agent to mix features as long as they are weighted equally.

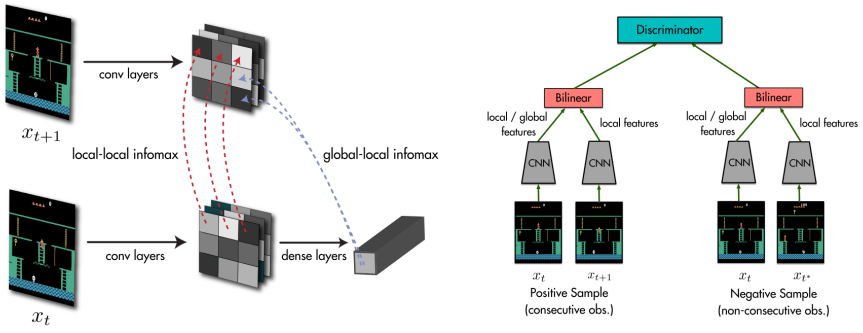


Figure 6.2.1: Unsupervised state-representation learning: Bilinear combinations of CNN-embeddings are distinguished by a discriminator to be either valid successive states or false combinations maximizing the representative quality of the embedding. (Anand et al., 2019)

6.2.2 Unsupervised Value-Agnostic Feature Learning

An overwhelming majority of deep-RL models are trained end-to-end, meaning that each trainable parameter is optimized with respect to a value-function (in some cases, a policy) - from stimulus input to action output. This paradigm corresponds to a severe case of over-specialization, where the trained agent gears its entire being towards the solution of a single task. The scientific community has seen significant successes of such agents on a narrow set of challenges, but as trends shift towards more general and integrated perspectives of intelligence, the interest in multi-tasking agents increases.

Of course, an agent whose visual processing pipeline is optimized to solve one task will have trouble "seeing" essential features for an entirely different value-function. On the other hand, biological sensory systems have evolved to solve various necessary tasks and leave enough room for adjustments over the agent's lifetime. Most of these mechanisms are constructed according to genetic plans as the organism grows, with no need to explicitly learn or optimize their function. They tend to account for most definitions of bottom-up attention. While there are some evolved biases (e.g., the shape of the pupil and position of the eyes), at a base level, all sensory organs are simple stimulus-response mechanisms that create neural representations of the world around us.

Following this logic, it should be more acceptable to introduce pre-trained or out of the box models as sensory modules to RL, especially in the light of required multi-task performance. Unsupervised representation learning is a popular topic in machine learning research. In visual domains, variational autoencoders have achieved great success in representing interpolable class representations at small scales, i.e., small feature vectors (Kingma et al., 2014; Kingma and Welling, 2014). With vision being the most prominent domain on which RL agents act, there have been sporadic attempts of merging VAEs with learning agents. However, some limitations of the VAE approach prove fatal to learning for decision-making. Most glaringly, the reliance upon pixel-based loss functions which measure the importance of visual features based on the relative frequency of their associated pixel-values, and

omitting smaller but semantically significant features from their representations. A recent approach for unsupervised representation learning attempts to remedy the problem by introducing a mutual information-based loss function (Devon Hjelm et al., 2019; Anand et al., 2019), called Spatiotemporal Deep InfoMax. In this way, a convolutional neural network will generate feature vectors from RGB-images that both capture local features of small image patches and a global representation relating all image patches. The representations are trained outside the RL cycle and promise to be general enough to be applied to several tasks within the same environment. Additionally, the authors introduce a creative temporal bias, in that the model does not encode simple images but image transitions, ensuring that the most salient parts of the image are captured in the limited representation space. We find that using this pre-trained sensory apparatus reduces training and convergence times compared to the traditional end-to-end approach in an initial trial run.

The feature-based models presented in this work certainly need a robust stimulus-to-representation encoder to function in a broader and more applied range of scenarios. The Infomax approach is a promising candidate for a lightweight solution to this challenge.

6.2.3 Working Memory as Temporal Attention

As mentioned in the introductory section on attention and memory (subsection 2.1.2), there is a significant conceptual overlap between the two mechanisms (Ahissar and Hochstein, 2000). In theory, attention governs access to working memory, severely affecting long-term memory formation and, of course, situational cognition and decision-making (Aly and Turk-Browne, 2017; Córdova et al., 2019). In practice, such a mechanism must exist so as to select those feature aspects of the perceivable world that are most relevant for a current goal, disregarding vast amounts of uncorrelated but readily available data. This requirement arises from the limited budget of working memory, theorized to hold no more than six to eight elements of information at any given time.

Conversely, attention is necessary to recall experiences from deeper storage mechanisms such as long-term memory, conditioned on a given situation or task-requirement. Whether and how these mechanisms are implemented on a neural level remains unclear. However, self-attention may serve as one candidate algorithm to introduce them to an artificial RL agent. Since dot-product attention was initially introduced as a model for natural language processing problems, it is by design a sequence model, meant to capture interactions between words, phrases, and paragraphs that extend over a temporal dimension.

The above experiments already illustrate that a self-attention agent will select the same feature across several time-steps, likely in order to capture dynamic movement and other rates of change. It seems only plausible that this behavior would scale with the rolling observation window's size, meaning that temporally far-removed feature observations could potentially affect online decision-making. Examples of this include visual cues that indicate the advantage of one action over the other as the agent navigates a labyrinth. More specifically, if an agent finds a

key, they might want to recall a locked door observed earlier and make a return.

In a sense more faithful to classic deep-RL applications, self-attention may replace models with more complex computation graphs. For example, the established IMPALA-architecture involves an LSTM-cell following the convolutional encoder. This recurrent subnetwork is used to maintain "longer-term" memories of previously seen RGB-observations or some of their most important aspects. LSTM networks are notoriously difficult to train, an issue that is exacerbated by their application in shifting learning paradigms such as RL. Additionally, while LSTMs are more effective retainers of memory than basic recurrent neural networks, there is a good chance that important information will exit the internal network state before it needs to be recalled. While more complex in terms of physical memory requirements, self-attention may act on large sequences of experiences translating each observation into query, key, and embedding representations for further processing. As described previously, the relation between every pair of previous experiences is removed by only one edge in the computation graph. Consequently, every previously recorded memory is related to the most recent observation and can thus be recalled with no loss of fidelity.

Many recent improvements to the self-attention mechanism, such as linear complexity and internal caches, are well poised to facilitate an attention-based artificial working memory. For example, the computational complexity does not necessarily need to be quadratic since the most crucial query element is the newest observation. In this way, the agent would likely benefit from only relating memory to the newest observation, resulting in linear complexity with respect to the size of the memory buffer.

6.2.4 Architectures for Multi-Modal Attention

The distinction between top-down and bottom-up attention is a frequent topic in attention research and across this work. In the conducted experiments, the agents were trained on specific task-scenarios, suggesting a development of top-down attention. However, through the conditioning of both queries and keys, the attention maps were largely feature- and, by extension, stimulus-driven. The one exception was the spatial g-AWRL introduced in section 5.6.

Nevertheless, the self-attention mechanism is immensely versatile, and having it implement different forms of attention is a definite possibility (see Figure 6.2.2). In order to implement top-down attention as originally theorized in neuroscience, the query vectors could be conditioned on something other than the input features. For one, they could be derived from a task-description in natural language. In this way, the mechanism could easily implement task-conditioned top-down attention enabling multitasking similar to the FiLM model, but with significantly less structural overhead since there is no need to generate parameters for the underlying network. To account for shifts in the environment, the agent could either combine a simple content-based attention map with this top-down attention or alternatively add a notion of recent memory to the query input, such as an internal LSTM state. Several designs are conceivable, including different conditioning of the query- and key-representations, as well as temporal delays as suggested in (Mott et al., 2019).

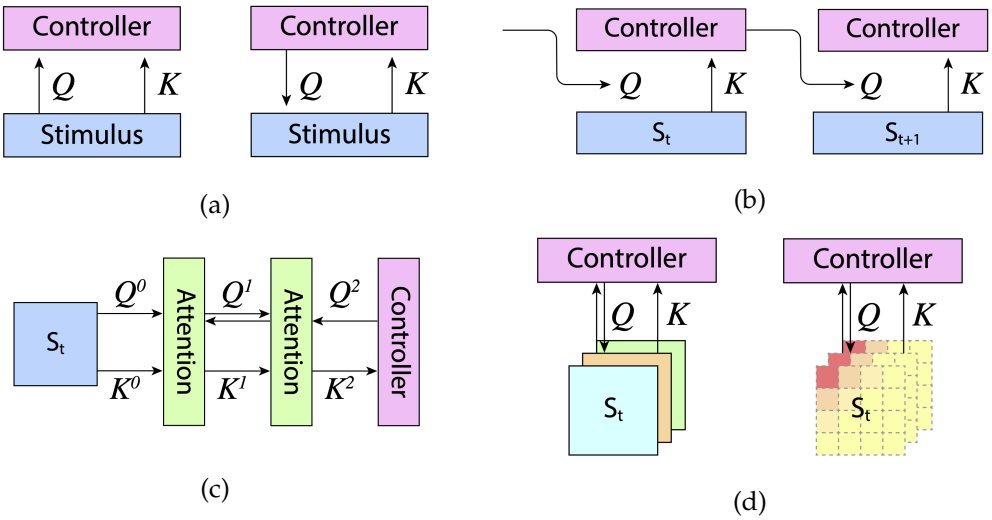


Figure 6.2.2: Versatile self-attention architectures for various modes of attention. (a) Query-only bottom-up and top-down attention. (b) Time-dilated query conditioning based on internal controller state, compare (Mott et al., 2019). (c) Intermittent hierarchical top-down and bottom-up attention. (d) Channel-wise, i.e. feature-based, and spatial attention.

Bibliography

Merav Ahissar and Shaul Hochstein. The spread of attention and learning in feature search: Effects of target distribution and task difficulty. *Vision Research*, 40(10-12):1349–1364, 2000. ISSN 00426989. doi: 10.1016/S0042-6989(00)00002-X. URL https://www.sciencedirect.com/science/article/pii/S004269890000002X.

Jay Alammar. The Illustrated Transformer – Jay Alammar – Visualizing machine learning one concept at a time, 2018. URL http://jalammar.github.io/illustrated-transformer/{%}0Ahttps://jalammar.github.io/illustrated-transformer/{%}0Ahttp://jalammar.github.io/illustrated-transformer/.

Mariam Aly and Nicholas B. Turk-Browne. How hippocampal memory shapes, and is shaped by, attention. *The Hippocampus from Cells to Systems: Structure, Connectivity, and Functional Contributions to Memory and Flexible Cognition*, pages 369–403, jan 2017. doi: 10.1007/978-3-319-50406-3_12. URL https://link.springer.com/chapter/10.1007/978-3-319-50406-3{_}12.

Ankesh Anand, Evan Racah, Sherjil Ozair, Yoshua Bengio, Marc Alexandre Côté, and R. Devon Hjelm. Unsupervised state representation learning in atari. Technical report, 2019. URL http://arxiv.org/abs/1906.08226.

Brian A. Anderson, Patryk A. Laurent, and Steven Yantis. Value-driven attentional capture. *Proceedings of the National Academy of Sciences of the United States of America*, 108(25):10367–10371, jun 2011. ISSN 00278424. doi: 10.1073/pnas.1104047108. URL www.pnas.org/cgi/doi/10.1073/pnas.1104047108.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight Experience Replay. Technical report. URL https://goo.gl/SMrQnI.

Jimmy Ba, Volodymyr Mnih, and Koray Kavukcuoglu. Multiple Object Recognition with Visual Attention. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, dec 2014. URL http://arxiv.org/abs/1412.7755.

Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. Technical report, 2015.

Izhar Bar-Gad, Genela Morris, and Hagai Bergman. Information processing, dimensionality reduction and reinforcement learning in the basal ganglia, dec 2003. ISSN 03010082.

Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The Arcade Learning Environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, jun 2013. ISSN 10769757. doi: 10.1613/jair.3912. URL http://stella.sourceforge.net/.

Richard Bellman. Dynamic programming. *Science*, 153(3731):34–37, jul 1966. ISSN 00368075. doi: 10.1126/science.153.3731.34. URL http://science.sciencemag.org/.

Marisa Carrasco. Visual attention: The past 25 years, jul 2011. ISSN 00426989.

Leonardo Chelazzi, Andrea Perlato, Elisa Santandrea, and Chiara Della Libera. Rewards teach visual selective attention. *Vision Research*, 85:58–72, jun 2013. ISSN 18785646. doi: 10.1016/j.visres.2012.12.005.

Jonathan D. Cohen, Steven D. Forman, Todd S. Braver, B. J. Casey, David Servan-Schreiber, and Douglas C. Noll. Activation of the prefrontal cortex in a nonspatial working memory task with functional MRI. *Human Brain Mapping*, 1(4):293–304, 1994. ISSN 10970193. doi: 10.1002/hbm.460010407.

Charles E. Connor, Howard E. Egeth, and Steven Yantis. Visual attention: Bottom-up versus top-down, oct 2004. ISSN 09609822.

Andrew R.A. Conway and Randall W. Engle. Individual Differences in Working Memory Capacity: More Evidence for a General Capacity Theory. *Memory*, 4(6):577–590, 1996. ISSN 09658211. doi: 10.1080/741940997. URL https://www.tandfonline.com/action/journalInformation?journalCode=pmem20.

Maurizio Corbetta and Gordon L. Shulman. Control of goal-directed and stimulus-driven attention in the brain. *Nature Reviews Neuroscience*, 3(3):201–215, 2002. ISSN 14710048. doi: 10.1038/nrn755. URL www.nature.com/reviews/neuro.

Jean Baptiste Cordonnier, Andreas Loukas, and Martin Jaggi. On the relationship between self-attention and convolutional layers. Technical report, 2019. URL http://arxiv.org/abs/1911.03584.

Natalia I. Córdova, Nicholas B. Turk-Browne, and Mariam Aly. Focusing on what matters: Modulation of the human hippocampus by relational attention. *Hippocampus*, 29(11):1025–1037, nov 2019. ISSN 10981063. doi: 10.1002/hipo.23082.

Aurelio Cortese, Benedetto De Martino, and Mitsuo Kawato. The neural and cognitive architecture for learning from a small sample, apr 2019. ISSN 18736882.

Susan M. Courtney, Leslie G. Ungerleider, Katrina Keil, and James V. Haxby. Transient and sustained activity in a distributed neural system for human working memory. Technical Report 6625, 1997. URL https://www.nature.com/articles/386608a0.

Peter Dayan, Sham Kakade, and P. Read Montague. Learning and selective attention. *Nature Neuroscience*, 3(11s):1218–1223, 2000. ISSN 15461726. doi: 10.1038/81504. URL http://neurosci.nature.com.

Harm De Vries, Florian Strub, Jérémie Mary, Hugo Larochelle, Olivier Pietquin, and Aaron Courville. Modulating early visual processing by language. Technical report, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/6fab6e3aa34248ec1e34a4aeedecddc8-Abstract.html.

R. Devon Hjelm, Karan Grewal, Phil Bachman, Alex Fedorov, Adam Trischler, Samuel Lavoie-Marchildon, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. Technical report, 2019.

Kenji Doya. Reinforcement learning: Computational theory and biological mechanisms. *HFSP Journal*, 1(1):30–40, may 2007. ISSN 1955-2068. doi: 10.2976/1.2732246/10.2976/1. URL https://www.tandfonline.com/action/journalInformation?journalCode=tfls21.

John Duncan, Glyn Humphreys, and Robert Ward. Competitive brain activity in visual attention. *Current Opinion in Neurobiology*, 7(2):255–261, apr 1997. ISSN 09594388. doi: 10.1016/S0959-4388(97)80014-1.

Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. Implementation matters in deep policy gradients: A case study on ppo and trpo. Technical report, 2020. URL http://arxiv.org/abs/2005.12729.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymyr Mnih, Tom Ward, Boron Yotam, Firoiu Vlad, Harley Tim, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures. Technical report, 2018.

Matt Jones Fabian Canas. Constructing Representations from Indirect Feedback Permalink Attention and Reinforcement Learning: Constructing Representations from. 2010. ISSN 1069-7977. URL https://escholarship.org/uc/item/1w83t8ct.

Michael J. Frank and David Badre. Mechanisms of hierarchical reinforcement learning in corticostriatal circuits 1: Computational analysis, mar 2012. ISSN 10473211. URL https://academic.oup.com/cercor/article/22/3/509/325387.

John M. Gardiner and Alan J. Parkin. Attention and recollective experience in recognition memory. *Memory & Cognition*, 18(6):579–583, nov 1990. ISSN 0090502X. doi: 10.3758/BF03197100.

Jacqueline Gottlieb, Mary Hayhoe, Okihide Hikosaka, and Antonio Rangel. Attention, Reward, and Information Seeking. *Soc NeuroscienceSign in*, 2014. doi: 10.1523/JNEUROSCI.3270-14.2014. URL https://www.jneurosci.org/content/34/46/15497.short.

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural Turing Machines. oct 2014. URL http://arxiv.org/abs/1410.5401.

Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, Adrià Puigdomènech Badia, Karl Moritz Hermann, Yori Zwols, Georg Ostrovski, Adam Cain, Helen King, Christopher Summerfield, Phil Blunsom, Koray Kavukcuoglu, and Demis Hassabis. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626): 471–476, 2016. ISSN 14764687. doi: 10.1038/nature20101.

Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *Proceedings - IEEE International Conference on Robotics and Automation*, pages 3389–3396. Institute of Electrical and Electronics Engineers Inc., jul 2017. ISBN 9781509046331. doi: 10.1109/ICRA.2017.7989385.

D O Hebb. *The Organization of Behavior: A Neuropsychological Theory*. 1949. ISBN 9780805843002. URL https://pdfs.semanticscholar.org/efee/3a0d3e8b34e45188dca4e19c15e6b6029edd.pdf{%}3C/eref.

Mark K. Ho, David Abel, Thomas L. Griffiths, and Michael L. Littman. The value of abstraction. *Current Opinion in Behavioral Sciences*, 29:111–116, 2019. ISSN 23521546. doi: 10.1016/j.cobeha.2019.05.001. URL https://doi.org/10.1016/j.cobeha.2019.05.001.

Ignace Th C. Hooge and Casper J. Erkelens. Peripheral vision and oculomotor

control during visual search. *Vision Research*, 39(8):1567–1575, apr 1999. ISSN 00426989. doi: 10.1016/S0042-6989(98)00213-2.

Wei Hu, Lechao Xiao, and Jeffrey Pennington. Provable Benefit of Orthogonal Initialization in Optimizing Deep Linear Networks. Technical report, 2020. URL http://arxiv.org/abs/2001.05992.

D. H. Hubel and T. N. Wiesel. Receptive fields of single neurones in the cat's striate cortex. *The Journal of Physiology*, 148(3):574–591, oct 1959. ISSN 14697793. doi: 10.1113/jphysiol.1959.sp006308.

Sabine Kastner and Leslie G. Ungerleider. Mechanisms of visual attention in the human cortex. *Annual Review of Neuroscience*, 23(1):315–341, mar 2000. ISSN 0147006X. doi: 10.1146/annurev.neuro.23.1.315. URL http://www.annualreviews.org/doi/10.1146/annurev.neuro.23.1.315.

Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*. International Conference on Learning Representations, ICLR, dec 2015. URL https://arxiv.org/abs/1412.6980v9.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings*. International Conference on Learning Representations, ICLR, dec 2014. URL https://arxiv.org/abs/1312.6114v10.

Diederik P. Kingma, Danilo J. Rezende, Shakir Mohamed, and Max Welling. Semi-Supervised Learning with Deep Generative Models. *Advances in Neural Information Processing Systems*, 4(January):3581–3589, jun 2014. URL http://arxiv.org/abs/1406.5298.

Eric I. Knudsen. Fundamental components of attention. *Annual Review of Neuroscience*, 30(1):57–78, jul 2007. ISSN 0147006X. doi: 10.1146/annurev.neuro.30.051606.094256. URL http://www.annualreviews.org/doi/10.1146/annurev.neuro.30.051606.094256.

Vijay R Konda and John N Tsitsiklis. Actor-Critic Algorithms. Technical report, 2000.

George Konidaris. On the necessity of abstraction. *Current Opinion in Behavioral Sciences*, 29:1–7, 2019. ISSN 23521546. doi: 10.1016/j.cobeha.2018.11.005. URL https://doi.org/10.1016/j.cobeha.2018.11.005.

Máté Lengyel and Peter Dayan. Hippocampal contributions to control: The third way. Technical report, 2009.

Yuan Chang Leong, Angela Radulescu, Reka Daniel, Vivian DeWoskin, and Yael Niv. Dynamic Interaction between Reinforcement Learning and Attention in Multidimensional Environments. *Neuron*, 93(2):451–463, 2017. ISSN 10974199. doi: 10.1016/j.neuron.2016.12.040. URL http://dx.doi.org/10.1016/j.neuron.2016.12.040.

Penelope A. Lewis, Günther Knoblich, and Gina Poe. How Memory Replay in Sleep Boosts Creative Problem-Solving, jun 2018. ISSN 1879307X.

Zhaoping Li. A saliency map in primary visual cortex. *Trends in Cognitive Sciences*, 6(1):9–16, 2002. ISSN 13646613. doi: 10.1016/S1364-6613(00)01817-9.

N. J. Mackintosh. A theory of attention: Variations in the associability of stimuli with reinforcement. *Psychological Review*, 82(4):276–298, jul 1975. ISSN 0033295X. doi: 10.1037/h0076778.

Marcelo G. Mattar and Nathaniel D. Daw. Prioritized memory access explains planning and hippocampal replay. *Nature Neuroscience*, 21(11):1609–1617, 2018. ISSN

15461726. doi: 10.1038/s41593-018-0232-z. URL http://dx.doi.org/10.1038/s41593-018-0232-z.

Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133, dec 1943. ISSN 00074985. doi: 10.1007/BF02478259.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with Deep Reinforcement Learning. dec 2013. URL http://arxiv.org/abs/1312.5602.

Volodymyr Mnih, Nicolas Heess, Alex Graves, and Koray Kavukcuoglu. Recurrent models of visual attention. Technical Report January, 2014.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous Methods for Deep Reinforcement Learning. feb 2016. URL http://arxiv.org/abs/1602.01783.

Alex Mott, Daniel Zoran, Mike Chrzanowski, Daan Wierstra, and Danilo J. Rezende. Towards interpretable reinforcement learning using attention augmented agents. Technical report, 2019.

Anh Nguyen, Alexey Dosovitskiy, Jason Yosinski, Thomas Brox, and Jeff Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. Technical report, 2016.

Yael Niv. Reinforcement learning in the brain. *Journal of Mathematical Psychology*, 53(3):139–154, 2009. ISSN 00222496. doi: 10.1016/j.jmp.2008.12.005. URL http://dx.doi.org/10.1016/j.jmp.2008.12.005.

Yael Niv. Learning task-state representations. *Nature Neuroscience*, 22(10):1544–1553, 2019. ISSN 15461726. doi: 10.1038/s41593-019-0470-8. URL https://doi.org/10.1038/s41593-019-0470-8.

Yael Niv, Reka Daniel, Andra Geana, Samuel J. Gershman, Yuan Chang Leong, Angela Radulescu, and Robert C. Wilson. Reinforcement learning in multidimensional environments relies on attention mechanisms. *Journal of Neuroscience*, 35(21):8145–8157, may 2015. ISSN 15292401. doi: 10.1523/JNEUROSCI.2978-14.2015.

John M. Pearce and Geoffrey Hall. A model for Pavlovian learning: Variations in the effectiveness of conditioned but not of unconditioned stimuli. *Psychological Review*, 87(6):532–552, 1980. ISSN 0033295X. doi: 10.1037/0033-295X.87.6.532. URL https://psycnet.apa.org/record/1981-02676-001.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual reasoning with a general conditioning layer. In *32nd AAAI Conference on Artificial Intelligence, AAAI 2018*, pages 3942–3951. AAAI press, sep 2018. ISBN 9781577358008. URL www.aaai.org.

Diego Perez-Liebana, Katja Hofmann, Sharada Prasanna Mohanty, Noburu Kuno, Andre Kramer, Sam Devlin, Raluca D. Gaina, and Daniel Ionita. The Multi-Agent Reinforcement Learning in Malm\"O (MARL\"O) Competition. jan 2019. URL http://arxiv.org/abs/1901.08129.

Michael I. Posner, Charles R. Snyder, and Brian J. Davidson. Attention and the detection of signals. *Journal of Experimental Psychology: General*, 109(2):160–174, jun 1980. ISSN 00963445. doi: 10.1037/0096-3445.109.2.160. URL https://psycnet.apa.org/journals/xge/109/2/160.

Alexandre Pouget, Jan Drugowitsch, and Adam Kepecs. Confidence and certainty:

Distinct probabilistic quantities for different goals. *Nature Neuroscience*, 19(3): 366–374, 2016. ISSN 15461726. doi: 10.1038/nn.4240.

Zhuwei Qin, Fuxun Yu, Chenchen Liu, and Xiang Chen. How convolutional neural networks see the world - A survey of convolutional neural network visualization methods. *arXiv*, 1(2):149–180, 2018. ISSN 2577-8838. doi: 10.3934/mfc.2018008.

Angela Radulescu, Yael Niv, and Ian Ballard. Holistic Reinforcement Learning: The Role of Structure and Attention. *Trends in Cognitive Sciences*, 23(4):278–292, 2019. ISSN 1879307X. doi: 10.1016/j.tics.2019.01.010. URL https://doi.org/10.1016/j.tics.2019.01.010.

Martin Rolfs and Marisa Carrasco. Rapid simultaneous enhancement of visual sensitivity and perceived contrast during saccade preparation. *Journal of Neuroscience*, 32(40):13744–13752, oct 2012. ISSN 02706474. doi: 10.1523/JNEUROSCI.2676-12.2012. URL https://www.jneurosci.org/content/32/40/13744https://www.jneurosci.org/content/32/40/13744.abstract.

David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. Technical Report 6088, 1986. URL https://www.nature.com/articles/323533a0.

Melissa Saenz, Giedrius T. Buracas, and Geoffrey M. Boynton. Global effects of feature-based attention in human visual cortex. *Nature Neuroscience*, 5(7):631–632, 2002. ISSN 10976256. doi: 10.1038/nn876. URL https://www.nature.com/articles/nn876.

Adam Santoro, Paul W. Frankland, and Blake A. Richards. Memory transformation enhances reinforcement learning in dynamic environments. *Journal of Neuroscience*, 36(48):12228–12242, 2016. ISSN 15292401. doi: 10.1523/JNEUROSCI.0763-16.2016.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In *4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings*. International Conference on Learning Representations, ICLR, 2016.

John Schulman, Sergey Levine, Philipp Moritz, Michael Jordan, and Pieter Abbeel. Trust region policy optimization. Technical report, 2015.

John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. Technical report, 2016. URL https://sites.google.com/site/gaepapersupp.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. Technical report, 2017. URL http://arxiv.org/abs/1707.06347.

B. M. Sheliga, L. Riggio, and G. Rizzolatti. Orienting of attention and eye movements. *Experimental Brain Research*, 98(3):507–522, 1994. ISSN 00144819. doi: 10.1007/BF00233988. URL https://link.springer.com/content/pdf/10.1007/BF00233988.pdf.

Xiangxiang Shen, Chuanhuan Yin, and Xinwen Hou. Self-attention for deep reinforcement learning. In *ACM International Conference Proceeding Series*, pages 71–75. Association for Computing Machinery, apr 2019. ISBN 9781450362580. doi: 10.1145/3325730.3325743.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Pan-

neershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, jan 2016. ISSN 14764687. doi: 10.1038/nature16961.

Marijn F. Stollenga, Jonathan Masci, Faustino Gomez, and Juergen Schmidhuber. Deep networks with internal selective attention through feedback connections. Technical Report January, 2014.

RS Sutton and AG Barto. *Reinforcement learning: An introduction*. 2018. URL https://books.google.de/books?hl=en{&}lr={&}id=uWV0DwAAQBAJ{&}oi=fnd{&}pg=PR7{&}dq=sutton+barto+reinforcement+learning{&}ots=minLs52Yg4{&}sig=lqMMFFxnuXpBGUBt6O4G{_}J4Q5ZU.

Gongbo Tang, Mathias Müller, Annette Rios, and Rico Sennrich. Why self-attention? A targeted evaluation of neural machine translation architectures. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018*, pages 4263–4272, aug 2020. doi: 10.18653/v1/d18-1458. URL http://arxiv.org/abs/1808.08946.

Jan Theeuwes, Artem Belopolsky, and Christian N.L. Olivers. Interactions between working memory, attention and eye movements. *Acta Psychologica*, 132(2):106–114, oct 2009. ISSN 00016918. doi: 10.1016/j.actpsy.2009.01.005.

Stefan Treue and Julio C. Martínez Trujillo. Feature-based attention influences motion processing gain in macaque visual cortex. *Nature*, 399(6736):575–579, jun 1999. ISSN 00280836. doi: 10.1038/21176. URL www.nature.com.

Naotsugu Tsuchiya and Christof Koch. The Relationship Between Consciousness and Top-Down Attention. In *The Neurology of Consciousness: Cognitive Neuroscience and Neuropathology*, pages 71–91. Elsevier Inc., jan 2015. ISBN 9780128011751. doi: 10.1016/B978-0-12-800948-2.00005-4.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. Technical report, 2017.

Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-Attention with Linear Complexity. *arXiv*, jun 2020. URL http://arxiv.org/abs/2006.04768.

Xiaoqin Wang. Cortical coding of auditory features, jul 2018. ISSN 15454126. URL https://www.annualreviews.org/doi/10.1146/annurev-neuro-072116-031302.

Yuhui Wang, Hao He, Chao Wen, and Xiaoyang Tan. Truly proximal policy optimization. *arXiv*, mar 2019. URL http://arxiv.org/abs/1903.07940.

Ziyu Wang, Tom Schaul, Matteo Hessel, and Marc Lanctot. Dueling Network Architectures for Deep Reinforcement Learning Hado van Hasselt. Technical report, jun 2016. URL https://www.youtube.com/playlist?list=.

Kelvin Xu, Jimmy Lei Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard S. Zemel, and Yoshua Bengio. Show, attend and tell: Neural image caption generation with visual attention. Technical report, 2015.

Vinicius Zambaldi, David Raposo, Adam Santoro, Victor Bapst, Yujia Li, Igor Babuschkin, Karl Tuyls, David Reichert, Timothy Lillicrap, Edward Lockhart, Murray Shanahan, Victoria Langston, Razvan Pascanu, Matthew Botvinick,

Oriol Vinyals, and Peter Battaglia. Deep reinforcement learning with relational inductive biases. Technical report, 2019.

Jingwei Zhang, Lei Tai, Joschka Boedecker, Wolfram Burgard, and Ming Liu. Neural SLAM: Learning to Explore with External Memory. jun 2017. URL http://arxiv.org/abs/1706.09520.

Li Zhaoping. A new framework for understanding vision from the perspective of the primary visual cortex, oct 2019. ISSN 18736882.

Chapter 7

Supplementary Materials

A Neural Network Special Cases and Architectures

So far, we have discussed the traditional MLP, an artifical neural network whose nodes connect to each other node in the successive layer. These types of layers are also called fully-connected or dense, but there are numerous other layers, each of which provides unique capabilities to the overall model. We will describe three variants, that are used extensively in the experimental work, in detail.

A.1 Convolutional Neural Networks

One of the most prolific types is the convolutional layer. As the name suggests, it implements the convolution operation using a predefined number of differentiable filters per layer. By far, the most popular application for these layers is image data, using two-dimensional filters that traverse an image and create corresponding feature maps. However, convolutional layers have also found wide application in processing sequence data, be it written language or audio recordings.

The major contribution of convolutional layers is their ability to relate and capture spatial structure in the input data. The fundamental principle facilitating this property is parameter sharing, such that each individual filter with its set of adaptable parameters will traverse the entire input volume. For example, in images, the convolutional filters will thus adapt and capture recurring and characteristic visual features that have statistically significant effects on the output prediction. In this way, earlier layers of convolutional filters will recognize simple shapes, orientations, and colors, while later filters learn to capture their compound responses. Much of this process hinges on a number of hyperparameters, such as the filters' size, the number of filters per layer, and their step size. For images, the size of a filter can be given by the number of pixels it covers for a single convolution operation. The step size, too, would be indicated in how many pixels the filter passes over per step. Thus the formula for the output dimensions of a single convolutional layer, which unlike the fully connected counterpart also depends on the size of the input is: $(W - F + 2P)/S + 1 \mapsto \mathbb{Z}^+$ Where W is the input volume size, F is the size

of the filters, S is the stride or step-size, and P is the amount of additional padding (0 pixels for images) around the input volume.

Another important factor is the number of channels. Most inputs, treated with convolutional layers, will have multiple channels such as colors (for images) or frequency bands (in sound and associated methods from e.g. medical diagnostics). A 2d-convolutional filter of size $F = 3$, for example, will have the dimensions $\mathbb{R}^{3\times3\times D_{in}}$, where D_{in} is the number of channels at the input of the layer. The number of output channels is determined by the number of filters in the layer, such that each output channel is a spatial map of where the filter-associated feature occurs. In spite of this final hidden complexity, convolutional layers can drastically reduce the number of trainable parameters when compared to fully-connected layers.

In the following sections, we will explore how these layers can aid RL agents to perceive the world they interact with - be it in the form of images or more abstract, and semantic terms.

A.2 Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are another special case of an artificial neural network architecture. This method is largely applied to sequence data (such as language, meteorological data, biofeedback, etc.) and, similar to the CNN, exploits the principle of parameter sharing for specialized processing steps. A simple RNN-cell implements a forward step, that is remarkably similar to the basic MLP proposition with the notable difference that the mapping takes an additional argument: the previous hidden state of the cell $\boldsymbol{z}^{(t-1)}$ (note that the superscript here denotes the sequence index, not the index of a layer in an MLP). The computation thus unrolls into the following steps:

$$\begin{aligned}\boldsymbol{a}^{(t)} &= \mathbf{U}\boldsymbol{z}^{(t-1)} + \mathbf{W}\boldsymbol{x}^{(t)} \\ \boldsymbol{z}^{(t)} &= \tanh(\boldsymbol{a}^{(t)}) \\ \boldsymbol{o}^{(t)} &= \mathbf{V}\boldsymbol{z}^{(t)}\end{aligned} \tag{7.1}$$

Recall that $\boldsymbol{a}^{(t)}$ is a vector of linear activations of a single layer, and $\boldsymbol{z}^{(t)}$ is a vector of activations after applying a selected activation function (in this case, the hyperbolic tangent). Unlike the classic MLP, there is an additional step to formulate the actual layer output $\boldsymbol{o}^{(t)}$ - running the activation through a separate set of weights $\mathbf{V}$. Note that instead of maintaining a single set of weights per RNN cell, we maintain three; $\mathbf{W}$ for the input at sequence step (t), $\mathbf{U}$ for the internal state at $(t-1)$ (the combination of which generates the internal state at (t)), and $\mathbf{V}$ to generate the output at time step (t). The above operations can be summarized in a recursive function, as they are reapplied across all time steps $t = \{1, 2, 3, ..., T\}$.

RNNs are flexible in how they return outputs that translate to predictions of sequence elements, or classifications thereof. An RNN cell can output a prediction at each time step (for example to predict the following word in a sentence), or at the end of a sequence (for example to classify the sentiment of a sentence). In RL,

RNNs are generally used to capture previously observed dynamics and thus act as an intelligent memory buffer. This application can cover short-term dynamics, such as representing movement between two still images, or longer-term developments such as previously observed information that is no longer represented by the input at the current time step.

B Policy Gradient Theorem

Equation 3.13 formulates an optimization approach, which can constitute a direct search of the policy space. As a gradient-based method, it does so by mapping a finite set of parameters θ to a set of policies. Because of this property, action probabilities change smoothly as a function of the learned parameters, which endows the method with strong convergence capabilities as opposed to the previously mentioned (tabular) state-action methods. Additionally, this method does not require a value function for action selection, although it might use one to learn the policy parameters. The policy's continued dependence on its parameters enables this approximate gradient-based method.

However, the function approximation approach may still appear difficult because the objective function depends not only on actions taken but also on the distribution over states in the environment. Given a state, the effects of changes in the policy parameters on actions and rewards are easily computable. But the policy's impact on the distribution over states is a function of the environment and, in most cases, unknown. This fact raises the issue of how to compute the gradient of the objective function with respect to the policy parameters when it depends on the unknown effects of the policy on the state distribution.

Therefore, the policy gradient theorem states that there is an analytic expression of the gradient of performance with respect to the policy parameters, which does not involve the derivative of the state distribution. This gradient is proportional to the real gradient, up to a proportionality constant equaling the average length of an episode.

$$\nabla J(\theta) \propto \sum_{s} \mu(s) \sum_{a} q_{\pi}(s,a) \nabla \pi_{\theta}(a|s) \tag{7.2}$$

Where μ is the relative frequency of state s, q_{π} is the state-action value of state s and action a, and π is the probability of action a, given state s and parameters θ.

C Implementation-level Details of the PPO Algorithm

These details stem from the original code by Schulman et al. and are listed hereafter:

1. **Value function clipping:** As an actor-critic algorithm, PPO maintains a value

function estimate $V_\phi(s_t)$ parameterized by ϕ to compute advantage estimates. Although Schulman et al. state that the value function is optimized towards the following objective:

$$\mathcal{L}^V = (V_\phi - V_{target})^2, \tag{7.3}$$

their code uses a clipped objective function, similar to the PPO policy update (Equation 3.39):

$$\mathcal{L}^V = \min\left[(V_\phi - V_{target})^2, (\text{clip}(V_\phi, V_{\phi_{old}} - \epsilon, V_{\phi_{old}} + \epsilon) - V_{target})^2\right] \tag{7.4}$$

clipping V_ϕ around previous value estimates. ϵ is the same parameter that is used in the clipping of policy ratios and $V_{target} = A^\pi(s_t) + V_\phi(s_t)$. If one were to employ the single-step advantage estimate analogous to TD(0), the V_{target} expression would further equate to $r_t + V(s_{t+1})$. For GAE, the case is somewhat more complex but the equality holds approximately true.

2. **Orthogonal Initialization:** Instead of the standard initializer, the authors use an orthogonal initialization scheme for neural network parameters. This method finds common application in RNNs for its demonstrable stability across training procedures. Additionally, Hu et al. recently compiled and proved the empirical benefits of employing orthogonal initialization in deep neural networks, specifically in terms of convergence speed.

3. **Adam learning rate annealing:** The Adam optimization method (Kingma and Ba, 2015) already maintains adaptive learning rates for each parameter based on estimates of lower-order moments. Regardless, the implementation of Schulman et al. appliy a linear annealing to the learning rate, such that the original rate decays over the course of training.

4. **Reward clipping:** The rewards observed by the agent as it engages with the environment are rarely provided to the learning algorithm without change. The use of a steady regime of reward shaping ensures that the algorithm is applicable in a variety of experimental tasks. In this case, the rewards are clipped to the intervals of $[-5, 5]$ and $[-10, 10]$ thus reducing value estimate errors and, by extension, the size of corresponding gradient updates. In our implementation, we further limit this interval to $[-1, 1]$, which led to demonstrable stability in earlier works (Mnih et al., 2014).

5. **Observation Normalization:** Normalizing inputs to all manner of learning algorithms is a common method to reduce variance and increase separability of input features by limiting them to the same scale. The original as well as our code normalizes all inputs, both image- and feature-based, to mean-zero, variance-one matrices and vectors respectively.

6. **Global gradient clipping:** Lastly, after computing the gradient of both policy and value networks, the magnitude of the concatenated gradient updates for all parameters is clipped such that the global l_2-norm does not exceed 0.5.

Engstrom et al. make note of three further details that we have chosen to exclude based on their reduced significance in our testing period. These are:

1. Reward scaling: Employing the same mean-zero, variance-one scaling as for observations.

2. Observation Clipping: limiting values in the inputs to the $[-10, 10]$ interval. In practice, normalizing the inputs as listed above sufficiently reduces variance and makes the occurrence of such large values exceedingly rare.

3. Hyperbolic tangent activations: Tanh activations aid in keeping the gradients small and thus prevent their explosion. However, we did not find any necessity for this and instead observed reduced convergence rates when employing these activations instead of ReLU.

D Architecture Hyper-Parameters

Hyperparameter	**Value**
Horizon (T)	128
Learning Rate	2.5×10^{-4}
Num. PPO Epochs	3
Minibatch Size	32×8
Discount (γ)	0.99
GAE Parameter (λ)	0.95
Number of Actors	8
Clipping Parameter	0.2
Value Coeff.	0.5
Entropy Coeff.	0.01
Gradient Norm	0.5
Number of Attention Heads (n^h)	4
Query/Key dimensionality (d_k)	32
Embedding dimensionality (d_v)	32

Table 7.1: Hyperparameters for feature-level Atari experiments.

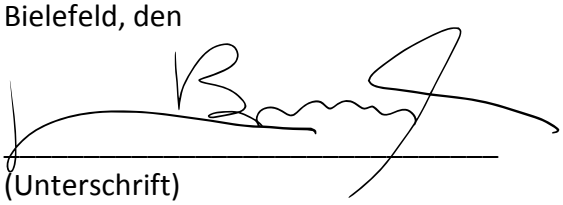
No. of parameters	**Inference**	**Policy (π)**	**Value (V^π)**
attention	$131,712$	$6,150$	$1,025$
convolutional	$125,184$	$19,974$	$3,329$
fully-connected	$146,080$	198	32

Table 7.2: Number of learnable parameters by model/sub-network for attention- and baseline-agents in feature-level Atari experiments. Number of layers specifies shared inference layers + separate policy- and value-layers.

Eigenständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Masterarbeit selbstständig verfasst und gelieferte Datensätze, Zeichnungen, Skizzen und graphische Darstellungen selbstständig erstellt habe. Ich habe keine anderen Quellen als die angegebenen benutzt und habe die Stellen der Arbeit, die anderen Werken entnommen sind - einschließlich verwendeter Tabellen und Abbildungen - in jedem Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht.

Bielefeld, den



(Unterschrift)