AN ABSTRACT OF THE THESIS OF

Mohamad Hosein Danesh for the degree of Master of Science in Computer Science presented on June 9, 2021.

Title: Re-understanding Finite-State Representations of Recurrent Policy Networks

Abstract approved: _____

Alan P. Fern

We propose an approach for understanding control policies represented as recurrent neural networks. Recent work has approached this problem by transforming such recurrent policy networks into finite-state machines (FSM) and then analyzing the equivalent minimized FSM. While this led to interesting insights, the minimization process can obscure a deeper understanding of a machine's operation by merging states that are semantically distinct. To address this issue, we introduce an analysis approach that starts with an unminimized FSM and applies more-interpretable reductions that preserve the key decision points of the policy. We also contribute an attention tool to attain a deeper understanding of the role of observations in the decisions. Our case studies on 7 Atari games and 3 control benchmarks demonstrate that the approach can reveal insights that have not been previously noticed.

Re-understanding Finite-State Representations of Recurrent Policy Networks

by

Mohamad Hosein Danesh

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Presented June 9, 2021
Commencement June 2021

Master of Science thesis of Mohamad Hosein Danesh presented on June 9, 2021.

APPROVED:

Major Professor, representing Computer Science

Head of the School of Electrical Engineering and Computer Science

---

Dean of the Graduate School

I understand that my thesis will become part of the permanent collection of Oregon State University libraries. My signature below authorizes release of my thesis to any reader upon request.

Mohamad Hosein Danesh, Author

ACKNOWLEDGEMENTS

I would like to sincerely thank my family for their unconditional love and support throughout this journey. I would also like to thank my advisor, Professor Alan Fern, for his constructive mentoring during my presence at Oregon State University. Further, I would like to thank Anurag Koul and Saeed Khorram for the joint discussions and collaborations.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF TABLES

LIST OF ALGORITHMS

Chapter 1: Introduction

1.1 Motivation

With the emergence of deep neural networks in reinforcement learning, problems were solved that previously were thought to be intractable, like playing video games directly from pixel inputs. Besides all the attention deep reinforcement learning is getting, there is an important concern regarding its explainability. Due to the black-box nature of deep neural networks, it is difficult to interpret and understand what governs agents' decisions. Since deep networks became popular, researchers have tried to demystify the inner workings of them. For example, the saliency method, which is mainly used in computer vision, is an approach to achieve that goal. It highlights parts of input data that are most salient to the model. Similar approaches have been applied to deep reinforcement learning settings. However, there are a few significant differences between deep reinforcement learning and computer vision, among which are the presence of memory in deep RL. Since saliency maps do not take the role of memory into account, then the explanations provided by them could be easily deceptive. An important insight of this could be that deep reinforcement learning methods are not always best served by directly importing techniques from other applications.

1.2 Contribution

First, we argue that minimization is not always the best approach in order to gain insights into recurrent policy networks. Minimization merges states that are not semantically related, which removes the interpretable structure and the ability to prune the individual non-merged decision points. Next, we introduce interpretable reductions to preserve the key features of the representation of the policy while not affecting the underlying behavior. This leaves us only decision points that are states in which the future behavior of the agent depends on the given observation. Then, we contribute an attention tool based on the Integrated Gradient [21] approach to obtain a deeper understanding of

the role of observations in decision points. Finally, we focus on decision points and functionally prune branches. The goal is for functional pruning to reveal redundant paths (hence unnecessary decision points in the machine). This is what leads to the insight that the policies are essentially open-loop.

Chapter 2: Related Works

2.1 Interpretability in Reinforcement Learning

The explainable reinforcement learning literature is vast, with a variety of approaches trying to understand the decision making process of an agent. For instance, [17] explains policy outputs by visualizations and t-SNE embeddings, [24] utilizes a high-level programming language, and [11] demonstrates how investigating important states in a trajectory can improve the trust for end-users. Although these works provide interesting insights into how RL agents decide, but they fail to show how they utilize their memory.

2.1.1 Attention maps

Attention maps have been used as a tool to identify the most relevant parts of the input with respect to the agent's decision [6, 12, 7, 1]. Perturbation-based attention methods have been investigated to gain insight into learned Atari policies [6, 12, 7], but have been criticized for relying on the application of networks to potentially nonsensical perturbed states [1]. This has been partly addressed by using more advanced counterfactual state generation [19]. In general, attention approaches produce a "local explanation" for the decisions made at specific states, which is in contrast to "global explanations" we primarily focus on in this work. Attention methods are also limited in the type of insights they can provide. For example, while they are applied to recurrent policies, they provide no insight into how memory is used and the strategic role of salient inputs. As we will show, our results suggest that a human's intuition about the role of salient inputs can be highly misleading.

2.1.2 Hybrid Architectures

There have been efforts to induce explanations components in the architecture to make agents implicitly explainable [28, 18]. For example, soft attention modules have been used in a recurrent architecture to gain insight into the "attention" of an agent [18]. While

attention has been used in several context for explanation, the soundness of this approach is not clear. In particular, the attention weights are computed from the raw observations and memory, which is ignored in the explanation process. Thus, it is difficult to determine whether the attention patterns carry key strategic information based on other parts of the raw observation, or whether, indeed, it is only the information highlighted by the attention that is key to decisions. Investigating ways to distinguish these two cases is interesting future work.

2.2 Extracting Finite-State Machine

There has been significant prior work on extracting FSMs from RNNs in the context of formal language learning of fixed finite alphabets [22, 3, 26, 27]. Only recently have there been attempts to learn FSMs for complex reinforcement learning problems [13] where machine minimization was used to extract high-level insights. Our work builds on [13] and aims to significantly advance the depth of understanding that such methods can provide.

Chapter 3: Re-understanding Finite-State Representations of Recurrent Policy Networks

3.1 Introduction

What roles do observations and memory play in the decision making of deep policy networks for complex control tasks? While such networks have yielded state-of-the-art performance in reinforcement and imitation learning (e.g. [16, 8, 25, 23, 9]), there are limited tools and approaches for giving insight into this question. This is particularly the case for policies represented as recurrent networks, e.g. LSTMs and GRUs [10, 5, 4], which condition on high-dimensional memory vectors with no preconceived semantics. Prior works have attempted to gain insight via attention maps over the network input, e.g. [6, 12, 7, 1], however, this ultimately involves subjective human interpretation of the underlying "strategic role" of the attended-to elements. In this thesis, we develop an analysis approach that reveals such human interpretations can sometimes be highly misleading.

Our work builds on a recent approach for understanding recurrent policy networks by quantizing memory and observations within an RNN [13]. A quantized RNN is a type of FSM, known as a Moore Machine (MM), which can be visualized and analyzed. In particular, the originally large MMs were algorithmically minimized, resulting in small machines for a variety of domains, which yielded high-level insights. For example, the minimal MM for Atari Pong showed that memory was not actually needed (i.e. a state-action mapping), while for Atari Bowling the MM was an open-loop strategy that ignored observations.

While analyzing minimal MMs allows for determining global properties, such as memory/observation use or not, we have found it difficult to gain more in-depth insight from minimal MMs. To see this, Figure 3.4 (a) and (b) show minimal MMs for Pong and Bowling and Figure 3.5 shows a set of frames/observations for each MM that all map to a single state in the corresponding MM. For a human, the frames are semantically distinct, whereas the minimization process was able to merge them all into a single state.

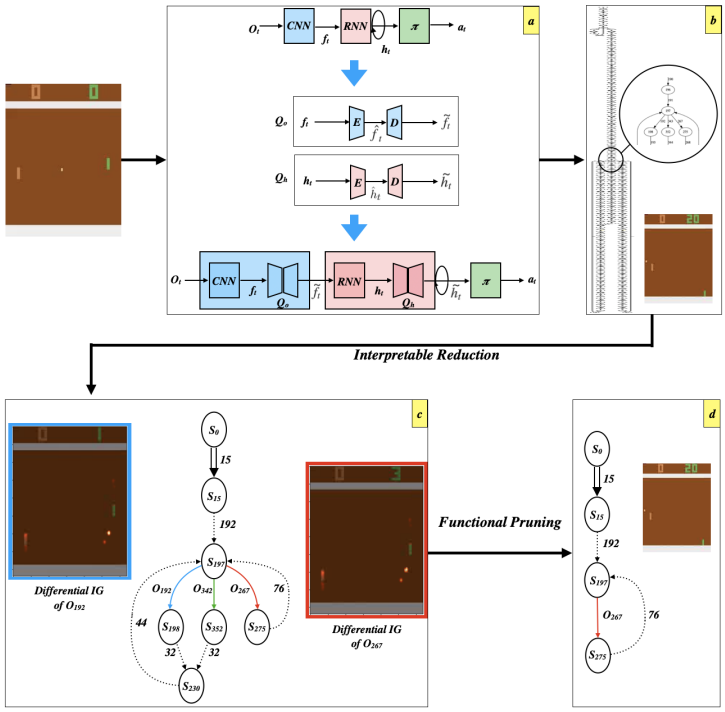


Figure 3.1: Overall approach for Pong. a) *QBN Insertion* (Section 3.2) discretizes observations and memory of the original RNN (top). b) Resulting *MM* with finite states and observations. c) *Interpretable reductions* (Section 3.3.1) are applied to the MM, yielding a single decision point s_{197} conditioning on observations. *Differential attention* (Section 3.3.2) is used to understand decisions in terms of observations. d) *Functional pruning* (Section 3.3.3) removes unnecessary branches, leaving an open-loop policy.

This is due to the minimization focusing on maintaining logically equivalence to the unminimized MM policy, rather than maintaining any semantic meaning of states. Thus, we have found it very difficult to understand the strategic role of different states in such MMs to gain deeper insight into their decision logic. This experience has led us to the view that minimal MMs are *unlikely* to be a good starting point for gaining a deeper understanding of recurrent policies.

Our main contribution is to develop a new approach for reinforcement learning researchers and advanced practitioners to analyze MM policy representations. We demonstrate that this approach can yield significantly more insight into the decision making of recurrent policies. Rather than start with a minimal MM after quantization, we start

with the unminimized MM, which is often quite large, and apply "interpretable reductions" in order to reduce the visual complexity. The reductions are intentionally simple in order to preserve the inherent decision structure in the machine while effective at compressing the visual representation. For example, one reduction operation replaces a fixed series of non-branching memory states (i.e. a macro) with a single abstract transition. The result is a simplified machine with a fixed set of "branching states" where the flow of the machine depends on observations.

In order to help understand the decisions made at branching states, we further develop a new differential attention tool, aiming to identify parts of observations that are most important for selecting one branch over another. Using the tool, we found that often, especially in Atari, the attention was unintuitive from a strategic point of view, which led us to question whether observations were used for strategically important reasons, or whether they were "arbitrary" branches. This led us to consider the "functional pruning" reduction to assess that issue. In particular, this operation eliminates all but one branch at a decision point to test whether the observation-conditioned decision among multiple branches was strategically important, or just an artifact of learning a non-compact policy.

We explore this approach by studying of 7 deterministic Atari games and 3 continuous control environments. For the control tasks, the approach identifies interpretable machines, whose decision points are understandable and strategically meaningful. In contrast, for Atari, the analysis reveals new and surprising insights. Prior works have attempted to understand the most salient pixels for policy decisions in Atari games [6, 12, 7], however, little insight was gained into how that information was used. Our approach reveals that for the Atari policies, observations were not used for "strategically useful" purposes. In particular, at each state that branches on observations, it was possible to remove all branches, except for one, resulting in an open-loop policy that ignores observations, while maintaining performance. We call such policies, *pruned open-loop policies*, and observe that all of our Atari policies are of this type. Finally, we identify limits of the current approach as problems become more stochastic, which suggests important avenues of future work.

3.2 Recurrent Networks to Moore Machines

In this section we review *recurrent policy networks (RPNs)* and how they can be converted to *MMs* as illustrated in Figure 3.1a. This work is agnostic about how a policy is learned, which, for example, could be via RL, imitation learning, or other training approaches.

3.2.1 Recurrent Policy Networks (RPNs)

An RPN is an RNN policy that, at each time step, is given an observation o_t and outputs an action a_t. As illustrated in Figure 3.1a top, during execution, an RPN maintains a continuous-valued hidden memory state h_t, which is updated on each transition and influences the action choice a_t. Specifically, given current observation o_t and current state h_t, an RPN outputs an action $a_t = \pi(h_t)$ where π may be a feed-forward network. Then, it updates the state according to $h_{t+1} = \delta(f_t, h_t)$, where f_t is a set of features extracted from o_t, for example, using a CNN when observations are images. δ is the transition function, which is often implemented via different types of gating networks such as LSTMs or GRUs.

3.2.2 Moore Machines

Understanding action choices of an RPN is complicated by the memory's high-dimensionality and lack of predefined semantics. Recent work [13] has attempted to address this issue by transforming RPNs to MMs, which allows for visualization and analysis of a finite system. An MM is a finite-state machine defined by a finite set of labeled hidden states H, a distinguished initial state $h_0 \in H$, a finite set of observation symbols O, and a transition function $\Delta : H \times O \to H$, which returns the next state $h_{t+1} = \Delta(h_t, o_t)$, given the current state and observation symbol. The label associated with each state corresponds to an action. An MM policy initializes the state to h_0 and then updates the state as observations arrive and outputs the action associated with each state.

3.2.3 Quantized Bottleneck Insertion (QBN Insertion)

We now overview the approach of *Quantized Bottleneck Insertion* [13] for transforming an RPN to an MM policy, which is illustrated in Figure 3.1a bottom. Full details of this approach can be found in the original paper and are not critical to the contributions of this thesis. The key components of the approach are *Quantized Bottleneck Networks (QBNs)*, which are simply auto-encoders, for which the encoder produces a quantized latent representation. In this work, bottleneck representation is composed of discrete units with output values in $\{-1, 0, 1\}$. Given a trained RPN, a representative set of RPN trajectories is produced and the resulting sets of hidden states $\{h_t\}$ and observation features $\{f_t\}$ are collected. Next, a hidden-state QBN Q_h and observation QBN Q_o are trained to minimize reconstruction error on the data sets. The encoders of Q_h and Q_o can be viewed as discretizing the state and observation spaces. The trained QBNs are then inserted into the RPN in place of the "wires" that propagate the continuous memory vector h_t and observation features f_t (see Figure 3.1a). This creates a discrete representation of the hidden states $\hat{h}_t$ and observation features $\hat{f}_t$ in the RPN. In practice, QBNs have reconstruction errors, which may impact the RPN performance. Supervised fine-tuning of the discretized RPN can be used to improve performance via imitation learning with respect to the original RPN.

Trajectories of the discretized RPN are then run to collect a representative transition set $\{(\hat{h}_t, a_t, \hat{f}_t, \hat{h}_{t+1})\}$, indicating that action a_t was taken in discrete state $\hat{h}_t$ and a transition to $\hat{h}_{t+1}$ was observed when the discrete observation was $\hat{f}_t$. The MM is constructed by creating a transition graph edge for each data tuple. The key parameters relevant to this thesis are sizes of the bottlenecks of Q_h and Q_o, denoted N_h and N_o respectively. Larger values give more potential to produce finer grained quantization and in turn more states.

3.3 Analyzing non-minimal Moore Machines

RPNs learned for complex problems can result in MMs with large numbers of discrete states and observations. In order to aid understanding, previous work [13] used a standard minimization algorithm [20] to produce equivalent minimal state MMs. As described in Section 3.1, the minimal machines allowed for interesting global insights into

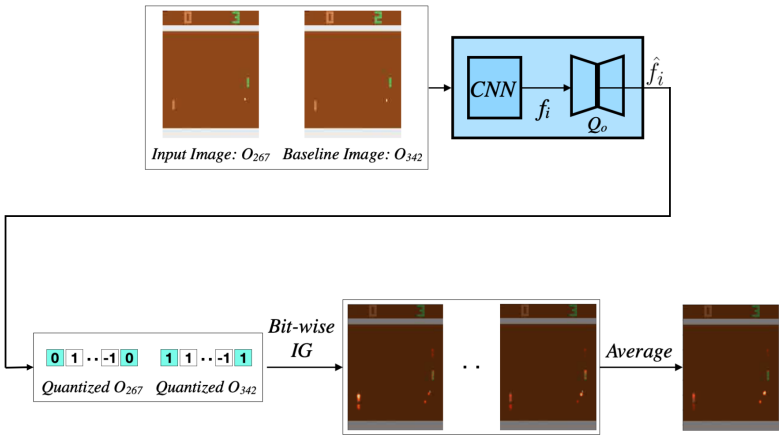


Figure 3.2: Differential Attention Pipeline. The pair of images under comparison result in discrete representations that differ on the highlighted features. We produce an attention map for each of those features using the Integrated Gradient (IG) approach and average the magnitude of the maps for an overall differential attention map.

the general use of states and observations. However, as also described in Section 3.1, we have found that minimal MMs obscure the decision making behavior, since they compress the original MM structure with no regard for interpretability. For example, the observations mapped to single states often appear to be semantically very different (Figure 3.5), making it difficult to understand the role of states and how observations influence their choices. Thus, in this work, we start with the unminimized MMs with the aim to gain a more detailed understanding. Below, we describe the steps of our analysis approach.

3.3.1 Interpretable Reductions

Figure 3.3a shows an example MM, which illustrates the main structures we observed in the learned MMs we analyzed. These structures provide several opportunities for simple *interpretable reductions*, which simplify the visualization of an MM without obscuring decision structure. *The reductions are very simple by design and one of our contributions is to notice that this simple set can be effective for interpreting very large MMs.* The first stage of our approach applies the following interpretable reductions.

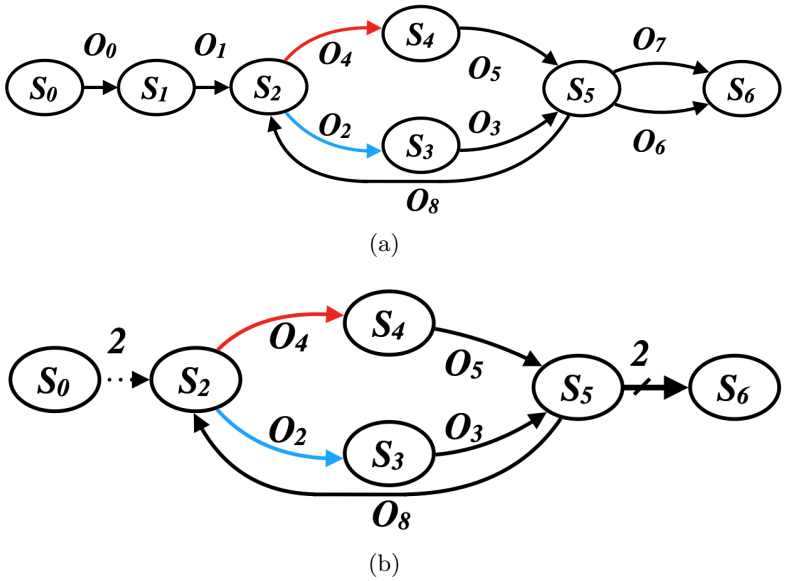


Figure 3.3: a) An example of an MM with 7 memory states and 9 observations. The initial state is S_0, and based on the input observation, it goes to the next state until it reaches to the state S_5, then it loops back to S_2. In this example, MM has a decision point at S_2. b) Interpretable reductions applied to the MM.

3.3.1.1 Sequence Reduction

It is common to see long sequences of states with a single observation between consecutive states (e.g. S_0 to S_2 in Figure 3.3a). These sequences are open open-loop macros that simply execute a fixed sequence of actions whenever encountered. Examples are S_0 and S_1 in the sequence of state-transitions from state S_0 to S_2 that have no branching states. We reduce these sequences to a single "macro arc" represented as a dotted line with a number indicating its length.

3.3.1.2 Loop Unrolling Reduction

There are many loops attached to sequences that are only traversed once when the sequence is visited. These loops increase the visual complexity of the MMs and make it appear as if there is a decision that controls loop exit, when there is not. Thus, we

simply unroll such loops before applying sequence reduction.

3.3.1.3 Parallel Reduction

There are often multiple transition arcs between two states with different observation labels (e.g. between S_5 and S_6 in Figure 3.3a). This can also occur for self-transitions. We merge these arcs into an abstract arc labeled by the number of observations.

3.3.1.4 Startup and Termination Reduction

In some MMs, there is a period of state transitions that corresponds to a warm-up or a termination period in the environment where actions have no impact. If desired, these parts of an MM can be replaced with a macro arc represented by two parallel lines with the number of transitions in that period. This reduction requires minimal human annotation of the steps where episodes "meaningfully" begin and end, which is usually straightforward. For example, in Atari games, there is usually a warm-up period before actions impact the game and the machines can have arbitrary structure that is not important to the game playing behavior. It is straightforward for a human to mark the time when this warm-up period ends.

The result of these reductions is shown in Figure 3.3b and often result in orders of magnitude smaller visualizations, e.g., going from Figure 3.1b to Figure 3.1c. Note that interpretable reductions do not change the behavior of the agent, nor the control flow, but are only for improved visualization. The states remaining in the reduced diagram are *decision points* (e.g. state S_2 is the single decision point in Figure 3.3b), where the next state, and hence future behavior, depends on the observation. The decision points are the key states in the machine that dictate how its behavior is influenced by observations. It is these points where we can gain the most insight about an MM.

3.3.2 Differential Attention for Decision Points

Given an MM decision point, we are interested in understanding how the raw observations (e.g. image pixels) influence its decisions. In particular, for a pair of outgoing branches labeled by discrete observation $\hat{o}_1$ and $\hat{o}_2$, we would like to answer the question: *"What*

features of the raw observations are most influential to selecting the $\hat{o}_1$ branch versus $\hat{o}_2$?". To help answer this, we consider pairs of raw observations o_1 and o_2 that occur at the decision point during MM execution (e.g. o_{342} and o_{267} in Figure 3.1c), such that $\hat{o}_i = E_o(o_i)$. That is, o_1 and o_2 are example observations that cause the machine to differentiate between the branches. We then produce a *differential attention map* $S(o_1, o_2)$ that highlights the parts of o_1 and o_2 that are most responsible for preferring branch $\hat{o}_1$ over $\hat{o}_2$. To compute $S(o_1, o_2)$, we focus on the set $F(o_1, o_2)$ of discrete features produced by E_o that differ between o_1 and o_2. As described below, for each $f \in F(o_1, o_2)$, we first produce an attention map $S[f](o_1, o_2)$ that highlights the parts of o_1 and o_2 that "explain" the difference in value of f. $S(o_1, o_2)$ is then just the average of the individual maps.

We compute each $S[f](o_1, o_2)$ via a straightforward, but novel, adaptation of the *Integrated Gradient (IG)* attention approach [21]. Originally, IG was used to compute attention maps that explain the decision of a classifier on a single image/observation o. Fundamental to the approach is the notion of a *baseline* image o_b, which is used as a reference that is assumed to not excite the classifier. In an image domain, the baseline is often a constant or noise image. Let $f(o)$ be the classifier response for observation o (usually the largest class-specific input to the softmax layer), noting that $f(o_b)$ will be small, and let $IG_i[f](o, o_b)$ be the corresponding attention value produced by IG for feature/pixel i. The key property of IG, which makes it a meaningful notion of attention, is the relation $\sum_i IG_i[f](o, o_b) = f(o) - f(o_b)$. Thus, the attention value of pixel i can be viewed as its additive contribution to the difference in classification responses for O over O_b. For space reasons we refer the reader to the original paper [21] for details of the IG computation.

Figure 3.2 shows how we adapt IG to compute a differential attention map $S[f](o_1, o_2)$ by treating o_2 as the baseline and letting the response function $f \in F(o_1, o_2)$ be the continuous features computed by the encoder just before the discretization step. That is, differential attention is given by $S[f](o_1, o_2) = IG[f](o_1, o_2)$. This means that differential attention satisfies $\sum_i S_i[f](o_1, o_2) = f(o_1) - f(o_2)$, which has the interpretation that each attention value $S_j[f]$ can be viewed as an additive contribution to the difference in response for o_1 and o_2, i.e. the preference of o_1 over o_2.

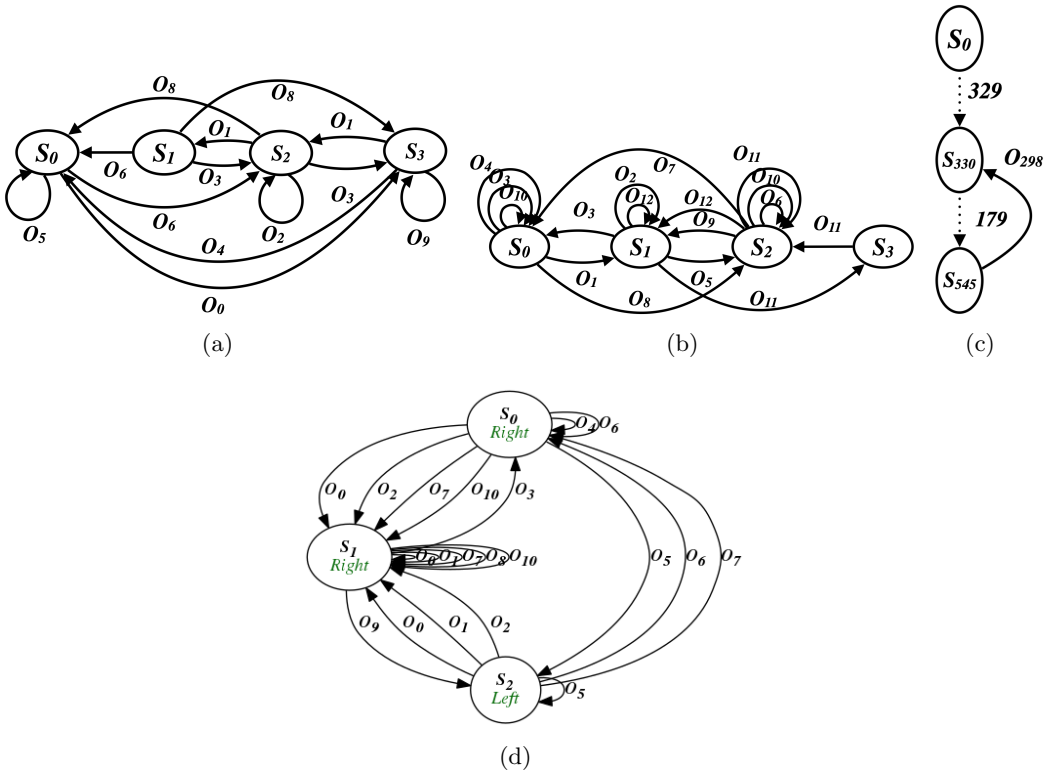


Figure 3.4: a) Pong minimal MM, b) Bowling minimal MM, c) Bowling pruned MM, d) Acrobot minimal MM.

3.3.3 Functional Pruning

After exploring the reduced MM graphs and attentions in Atari, we made two high-level observations. First, as shown in our experiments the differential attention results often indicated that observations were not being used in a strategically meaningful way, e.g., branching on observations that were extremely similar. Second, the graphs beyond different branches at a decision point were often very similar and appeared to address similar situations. We hypothesized that many decision points may not be strategically meaningful, but rather just an artifact of learning. That is, even if a branching decision is not required at a certain point in a game, the inclusion of a decision point that conditions on an arbitrary part of the observation will not hurt the performance as long as good behavior is learned for each of the resulting branches. In such situations, the choice of

which branch to traverse may be arbitrary and any one of them may work. Note that the usual attention-based tools [6, 12, 7] will simply indicate the agent's attention and can lead a human to incorrectly infer strategic relevance.

Detecting unnecessary decision points cannot be done by just graph analysis—rather, empirical analysis of modified machines is required. To do this, we conduct a simple form of *functional pruning* (described in detail in Section 3.3.3.1), to identify and prune unnecessary branches at decision points. Our approach considers each MM decision point, and prunes each of the branches one at a time, in order of the least to most frequently visited based on multiple MM runs. When a branch is pruned, we empirically estimate the performance of the resulting MM, noting that when the machine would have previously taken the pruned branch, we force it to take the most frequent branch. If the empirical performance does not degrade beyond a threshold, we permanently remove the branch and move on to the next pruning step. *The intent of function pruning is not to preserve logical equivalence.* Rather, the intent is to test whether observations were strategically important or just an artifact of learning. After gaining the insights, one could decide to use the original machine or attempt to improve it.

We found greedy pruning to be effective for the MMs considered in this work. For example, in Figure 3.1 we see that functional pruning resulted in pruning all but one branch from the single decision point in the MM, leaving an open-loop policy. In such cases, when an MM can be functionally pruned to the point of removing all decision points and leaving an open loop policy, we say that the MM is a *pruned open-loop policy*, which generalizes the notion of open-loop policy to include machines that condition on observations, but in ways that are not strategically necessary. As an analogy, consider a pruned open-loop policy for a human driving to a store along one of two equally good routes. The human may measure the temperature and decide between the routes depending on whether the temperature was an even or odd value. The observation had no real impact on the quality of the policy, since a fixed route could have been selected, but behavior is still seen to depend on observations.

3.3.3.1 Details on functional pruning

In this method, we perform forward parsing of the MM from the start-state to the final state. During parsing, we only consider outgoing transactions from a decision point

for the purpose of pruning. Over here, each transition is weighted as per its visitation frequency which is empirically estimated by multiple runs of the MM. At each decision point, we consider each transition for pruning in the least to most frequency order. Once a transition is pruned, the overall MM is evaluated for decay in performance. During evaluation, if we encounter the pruned observation transaction, we transact through the most frequent branch of the decision point. If there is a decay in performance, the transition is restored and other candidates are considered. This simple and greedy functional pruning method is able to keep the performance, while removing unnecessary branches from the MM. Since sequence of actions at two branch may be identical, or different than one another, this type of MM minimization may change the behavior of the agent, after functional pruning. Algorithm 1 shows the pseudo-code of functional pruning.

Algorithm 1 Functional Pruning

```
Output: Pruned MM
DecisionPoints = []
PrunedBranches = []
for node in (Nodes in MM) do
  if node is decision point then
    DecisionPoints.append(node and frequency)
  end if
end for
for DP in DecisionPoints do
  leastFreqBranch = the least frequent branch
  mostFreqBranch = the most frequent branch
  new MM = prune leastFreqDP from MM
  for node in new MM do
    if node is in leastFreqBranch then
      node = mostFreqBranch
    end if
  end for
  performance = record the performance of new MM
  if performance unchanged then
    PrunedBranches.append(leastFreqBranch)
  end if
end for
for PB in PrunedBranches do
  remove PB from MM
end for
Return MM
```

3.4 Experiments and Results

The only prior approach to compare against is the recent MM minimization approach for analysis of MMs [13]. As described earlier, we have found it very difficult to gain insights from minimized MMs. To further illustrate this point, we provide more qualitative

comparisons of the minimization approach to our approach in Atari games. Overall, the minimization approach does not reveal the insights of our new approach.

We consider 7 deterministic Atari environments: Bowling, Boxing, Breakout, MsPacman, Pong, SeaQuest and SpaceInvaders, and 3 stochastic discrete-action classic control tasks: Acrobot, CartPole, and LunarLander. For each experiment, we follow the choices of prior discretization work [13] for pre-processing, RPN architecture, QBN architectures, and training via A3C [15] reinforcement learning. Detailed information on these choices along with hyperparameter choices are provided in Section 3.4.1 and Section 3.4.2. We considered two sets of QBN sizes for each environment, shown in Table 3.1. From the table we see that the agent performance remains the same after reductions for all domains, except for Acrobot and LunarLander, where functional pruning reduced performance within the specified tolerance. The table also gives the number of states and observations, N_h and N_o, after the reductions. Interestingly, in Atari games, no decision points are left after functional pruning, i.e. all of the policies were "pruned open-loop" policies. The following case studies illustrate how our approach is useful for gaining insights and revealing unexpected properties of the decision logic.

3.4.1 Training Details

We used A3C with Adam optimizer ($lr = 1e - 4$) to train our Recurrent Policy Network(RPN). Also, we used discount= 0.99 and calculated policy loss using Generalized Advantage Estimation (GAE)($\lambda = 1.0$).

3.4.1.1 Atari

In this case, RPN comprises of 4 convolutional layers (kernel size 3, strides 2, padding 1, and 32, 32, 16, 8 filters respectively) with intermediate ReLU activations. The last convolutional layer has ReLU6. This is followed by a GRU Cell having 32 hidden units. The output of GRU is consumed by a "policy" and "value" linear network having 'action-space' and '1' unit, respectively.

3.4.1.2 Continuous Control Tasks

RPN is composed of 2 linear layers having 16 and 8 units, respectively. First layer's activation function is ELU and second layer's is ReLU6. Rest of the architecture is same as for Atari.

3.4.1.3 Quantized Bottleneck Networks

Each QBN comprises of 'n' layer encoder and 'n' layer decoder. At the bottleneck, we used the same $TernaryTanh$ operator to quantize the encoded representation as done in prior work. This quantized representation is fed to the decoder. Also, We used Tanh as intermediate activation's for encoder and decoder. We used $n = 2$ and $n = 3$ and apply RelU6, Tanh activation to last layers of Q_o and Q_h, respectively. In each QBN's encoder, for the first layer there are $8 \times$ *QBN SIZE* neurons, and for the second layer there are $4 \times$ *QBN SIZE* neurons. In the final layer there are *QBN SIZE* neurons. For QBN's decoder, the order is reversed. We use Adam optimizer ($lr = 1e - 4$) and max norm of the gradients was set to 5.

3.4.2 Pre-processing

In Atari games, input images are pre-processed. This has been done by applying a wrapper over OpenAI gym environments which gray-scales and resizes input images from 210×160 to 80×80 shape. Also, We use deterministic Atari environments with $frameskip = 4$. For Pong and SpaceInvaders, we changed the action space to [Noop, RightFire, LeftFire] and [Noop, Fire, Right, Left],respectively. These pre-processing steps are done in order to ease policy training and interpretation.

In classic control tasks, we do not exert any pre-processing on input features and action spaces.

3.4.3 Quantitative analyses

Table 3.1 provides the results of original MM, minimal MM, and our approach, for two QBN pairs. For either pair, our results are consistent, and agent with the pruned MM performs the same as the original agent, except for Acrobot and LunarLander. In those

two environments, we have a small drop in performance, but agent is still able to solve the problem. We have not included the detailed information about each interpretable reduction step (Section 3.3.1), since their purpose is to make visualization simpler. The underlying MM would not change by applying interpretable reductions, and basically, everything is the same as the original MM. Thus, there is no point in providing details such as the ones presented in Table 3.1.

According to Table 3.1, in minimal MM, except for one case, CartPole(4,4), the number of decision points and number of states are equal. Also, the number of decision points increases in many cases comparing to the original MM, which makes explainability capability more complex. As pointed out in the Section 3.4.4, it is because minimal MMs integrate the decision points with unrelated states. This strongly obscures the ability to interpret agent's decision making process.

3.4.4 Atari: Case Studies

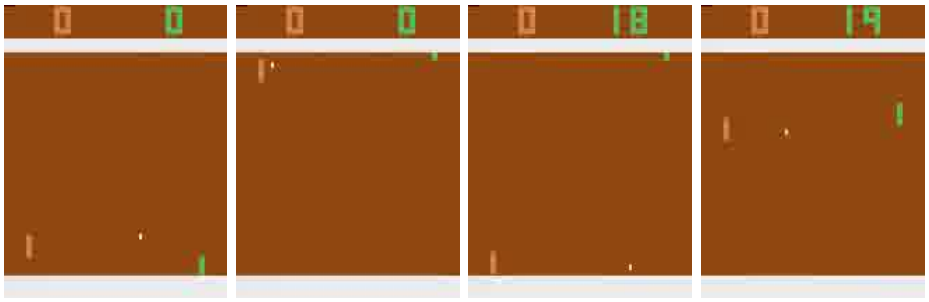
3.4.4.1 Pong and Bowling

The RPN for Pong achieved a maximum score of 21. The policy displays repetitive behavior by performing a "kill shot" against the opponent to win each point, though the behavior is not exactly the same across all shots. Figure 3.1b shows a view of the original large MM. Figure 3.1c shows the graph after the interpretable reductions, which is quite small with only one decision point at state S_{197}. This key decision point is the starting point of 3 possible loops, depending on the branch taken, and is entered once per round (each round is one point). This raised the questions of *"What basis is the machine using to decide which loop to enter?"* and *"Is there a strategic reason for the branching decision?"*.

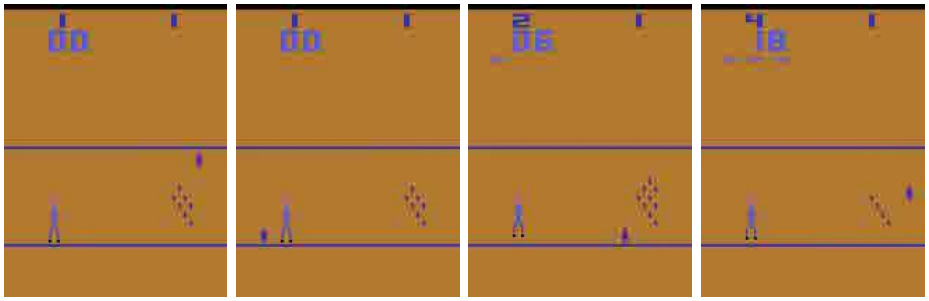
To investigate, we computed the differential attention for the decision of choosing S_{275} over S_{352}, as shown in Figure 3.2. It is striking that the sample observations associated with the branches are almost identical. The differential attention indicates that the ball region and tips of the paddles are the most critical factors in deciding between the branches. Close inspection reveals that the appearance and location of the ball in the two observations are subtly different. To understand this, we observed that these differences are due to the fact that at the beginning of each round the starting position

Table 3.1: MM results for control tasks and Atari environments.

Game	QBN Sizes		Original MM				After Pruning				After Minimization			
	N_h	N_o	Decis. Points	States	Obs.	Perf.	Decis. Points	States	Obs.	Perf.	Decis. Points	States	Obs.	Perf.
Acrobot	4	4	9	12	38	-77.1	2	4	3	-80.7	3	3	11	-80
	8	8	79	42	200	-77.1	2	4	5	-79.9	3	3	7	-86
CartPole	4	4	6	7	18	500	2	4	5	**500**	2	3	8	500
	8	8	7	8	58	500	2	4	5	**500**	3	3	11	500
Lunar Lander	32	32	195	1426	1150	180.48	184	1387	980	172.2	41	41	92	165
	32	64	249	1389	1437	204.93	230	1321	1327	197.35	19	19	73	147
Bowling	32	50	5	608	525	60	**0**	530	437	**60**	5	5	19	60
	64	100	5	630	552	60	**0**	546	488	**60**	3	4	12	60
Boxing	32	50	**0**	1274	1270	100	**0**	1274	1270	**100**	19	19	109	100
	64	100	**0**	1097	1095	100	**0**	1097	1095	**100**	14	14	101	100
Breakout	32	50	4	2479	2466	404	**0**	2365	2345	**404**	8	8	28	404
	64	100	5	1659	1608	404	**0**	1598	1540	**404**	11	11	43	404
MsPacman	32	50	43	728	716	3060	**0**	612	597	**3060**	21	21	70	3060
	64	100	34	895	876	3060	**0**	776	758	**3080**	9	9	45	3060
Pong	32	50	**0**	383	369	21	**0**	383	369	**21**	3	3	12	21
	64	100	2	384	369	21	**0**	271	268	**21**	4	4	10	21
SeaQuest	32	50	46	2167	2233	2580	**0**	1679	1577	**2580**	16	16	140	2580
	64	100	18	2244	2261	2580	**0**	1834	1883	**2580**	17	17	135	2580
Space Invaders	32	50	102	1700	1709	1820	**0**	1314	1350	**1820**	30	30	40	1820
	64	100	35	1914	1852	1820	**0**	1832	1802	**1820**	11	11	27	1820



(a)



(b)

Figure 3.5: Four observations that enter a state in the corresponding minimal MM. a) Frames of Pong for in-going observations to S_2, b) Frames of Bowling for in-going observations to S_0.

of the ball is minutely different for even versus odd rounds, which translated to the small difference observed at the decision point. Intuitively, this difference did not appear to have a strategic value. Indeed, after functional pruning (Figure 3.1d), we see that all branches were removed except for the one through S_{275}, leaving an open-loop strategy with no loss in performance. The even versus odd branching, was an unnecessary artifact of the RNN learning process.

For Bowling, the original large MM had 630 discrete states and 552 discrete observations. Our interpretable reductions revealed only 5 of those states corresponded to true decision points. Further, Figure 3.4c shows the result of functional pruning to get an MM with no loss in performance. Similar to Pong we end up with a pruned open-loop policy. Again the observations used at decision points were not strategically relevant and rather artifacts of learning.

3.4.4.2 Comparison to Prior Work

We now compare to the minimization approach of [13]. For Pong, our approach was able to isolate a single decision point (Figure 3.1d) where behavior depended on observations in an understandable way, which was ultimately determined to be non-strategic. Meanwhile, Figure 3.4a shows the minimal MM produced by prior work for the same initial MM. This minimal MM merges the key decision point with semantically unrelated states (e.g. from multiple macros), obscures insights, and gives no understanding of how memory and observations are used. To highlight this, Figure 3.5a demonstrates four frames of Pong which enter S_2 in the corresponding minimal MM. This is an evidence of minimal MMs merging states that are semantically distinct. Further, it is unclear how the equivalent of functional pruning could be done using the minimal MM, due to the merging of decision points. Also, it is unclear how to uncover the key insight identified by our approach by starting with the minimal MM.

A similar comparison holds for Bowling where our approach resulted in the open-loop policy (Figure 3.4c). Rather, the minimal MM of the same initial policy is shown in Figure 3.4b, which resembles the tightly coupled minimal MM of Pong. Figure 3.5b shows frames that lead to decision point S_0, which appear to be semantically very different from a human perspective. This is also the case for other states, which makes it very difficult to extract meaningful insights from the minimal MM. Again, it is completely unclear how we could start with the minimal machine and gain the realization that the observations play no significant strategic role, which our approach revealed.

3.4.4.3 MsPacman

The original MM for MsPacman with QBNs of size (64, 100) has 34 decision points before accounting for the warm-up and termination periods. After the reduction for warm-up and termination the MM is left with 19 decision points. We observed that most of the branches in the MM are visited only once and any single state transition is covered no more than 4 times during an episode. This indicates that little high-level generalization of the strategy is occurring. We considered the differential saliency at all decision points and show one example in Figure 3.6a. The saliency primarily focuses on the middle of the map at a location which distinguishes the presence of Pacman

and less attention at a location distinguishing the presence of a ghost. By applying the functional pruning, we are able to remove all "non-strategic" branches from all decision points throughout the MM. This emphasizes that these salient features simply serve as arbitrary landmarks with no strategic value. This left us with an open-loop controller with no drop in performance. In fact, performance improved by 20 points.

The pruned MM, depicted in Figure 3.6b, gives the insight that agent's policy is a pruned open-loop policy. The edges with two parallel lines indicate the start-up and termination phases of the game, which is an interpretable reduction introduced in the paper. On the other hand, the minimal MM, shown in Figure 3.6c, gives no insights about the agent's behavior. Since semantically unrelated states are matched to each other, complexity is even increased. Figure 3.7 shows an example of this introduced complexity. All four frames in Figure 3.7 are outgoing observations of a decision point, S_3, in the minimal MM. As it can be seen, these observations are semantically distinct to humans. Figure 3.7a shows a frame early in the game, where there are lots of rewards available and ghosts are not out completely. Figure 3.7b and c show a frame in the middle of the game. In b, Pacman is not being threatened by any ghosts, but in c there is a very close ghost to it. Figure 3.7d shows a frame from almost end of the game where rewards are sparse. Although these frames encode very different semantics, but minimal MM treats them semantically related, which is misleading in terms of interpretation.

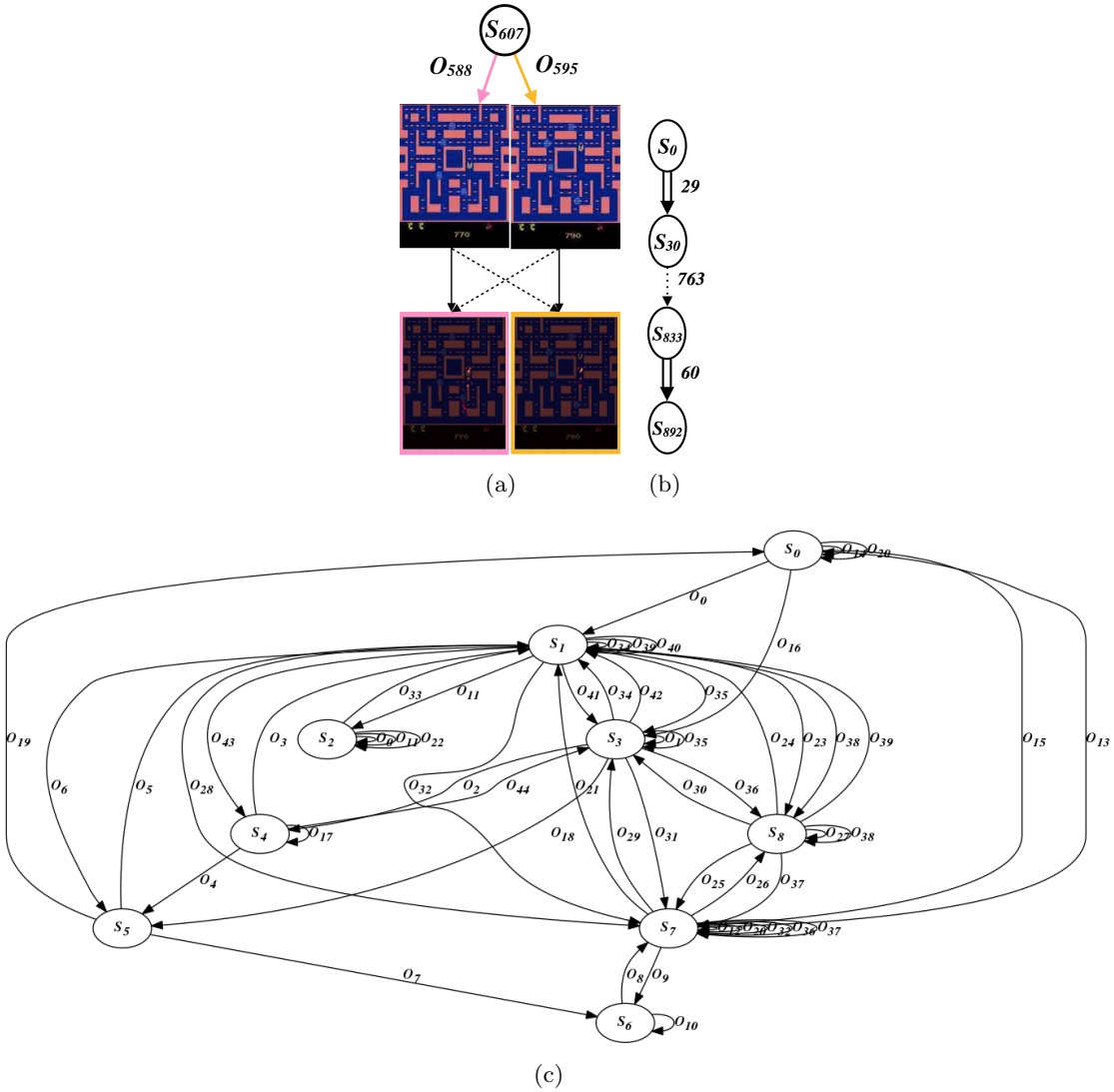
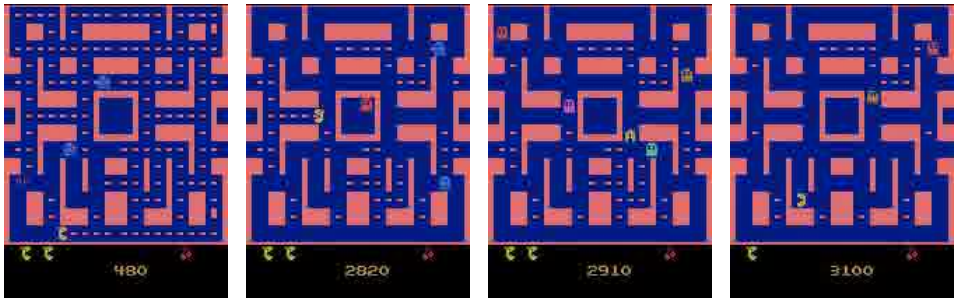


Figure 3.6: MsPacman: a) Differential saliency for a sample decision point. The first row shows a sample observation that occurs for each branch. The second row gives differential saliency for pairs of observations (dotted arrows indicate the baseline), b) pruned MM, c) minimal MM.



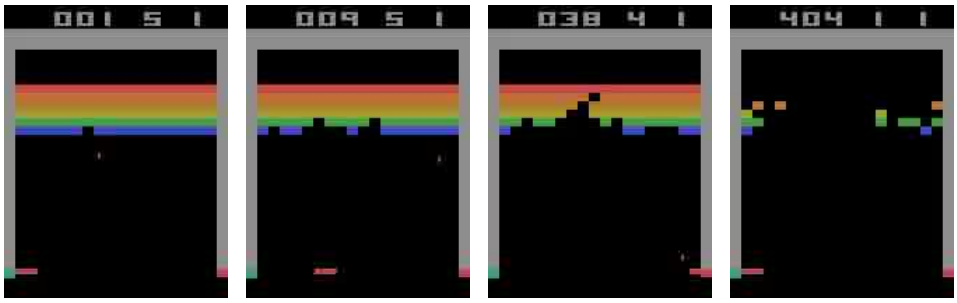
(a) (b) (c) (d)

Figure 3.7: Different observations as branches of decision point S_3 in minimal MM.

3.4.4.4 Breakout

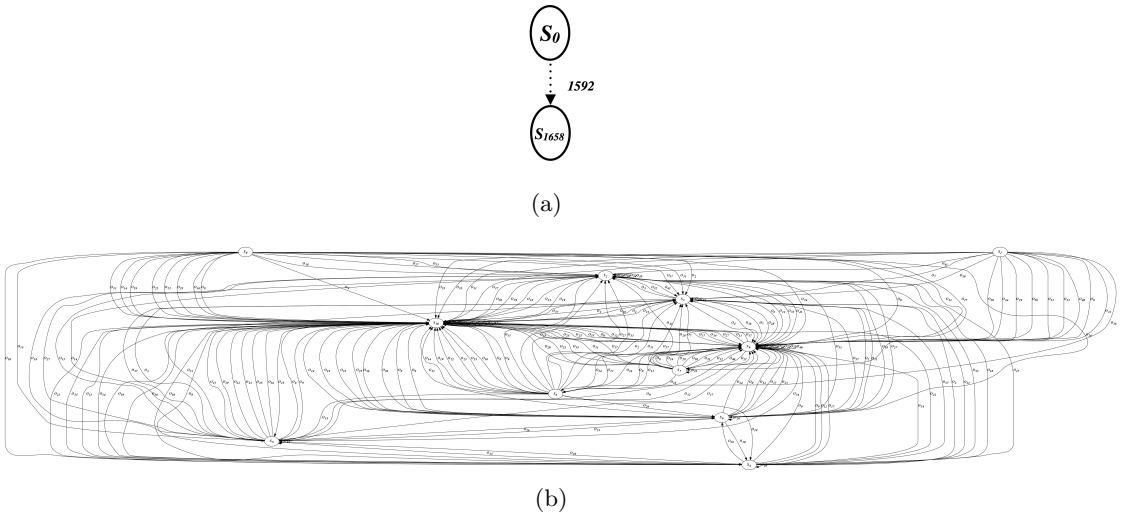
QBN pair of (64, 100) is considered in this game as well. In Figure 3.9a, pruned MM, and in Figure 3.9b, minimal MM can be seen. Breakout's policy turned out to be a pruned open-loop policy, but this could not be possibly understood by looking at the minimal MM. In fact, it is hard to get any explanation about policy based on minimal MM.

Figure 3.8 shows four semantically different observations that are turned into branches of a state in the minimal MM. This set of various observations are the outgoing branches of decision point S_9 in the corresponding minimal MM.



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

Figure 3.8: Different observations as branches of decision point S_9 in minimal MM.



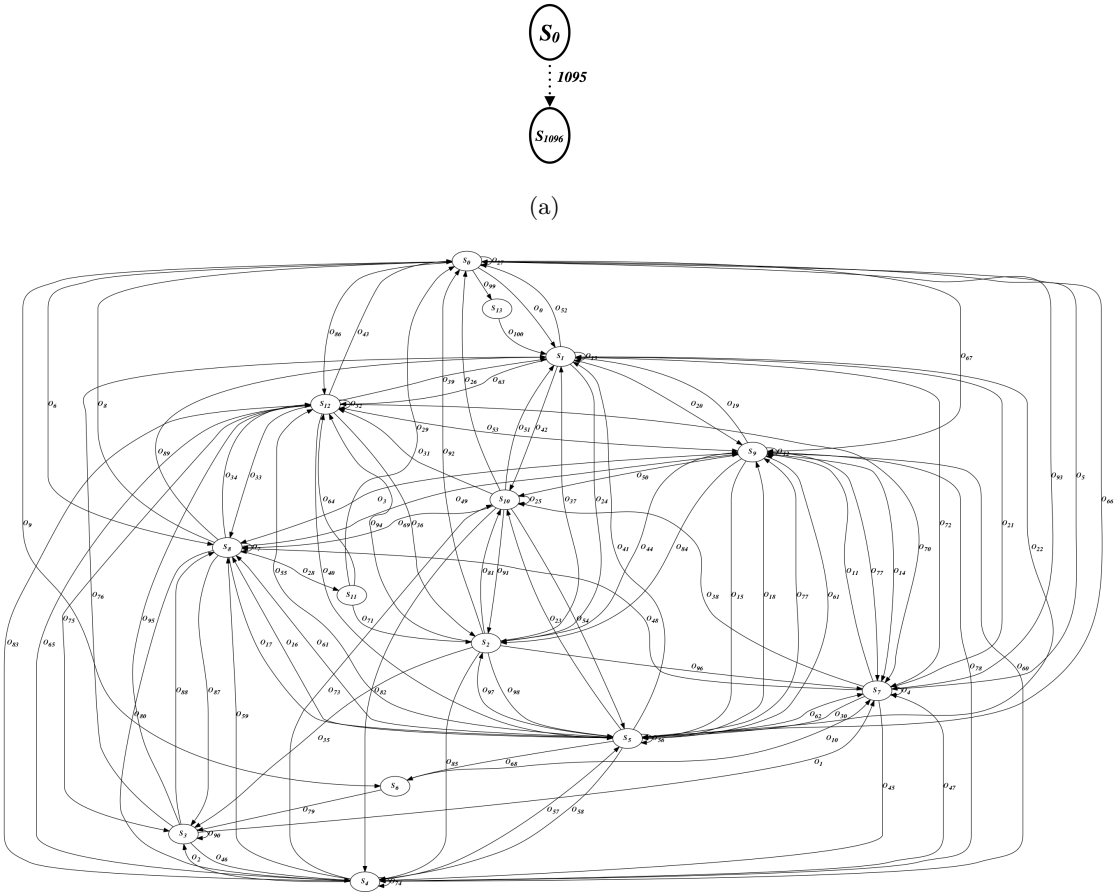
(a)

(b)

Figure 3.9: Breakout: a) pruned MM, b) minimal MM.

3.4.4.5 Boxing, SeaQuest, SpaceInvaders

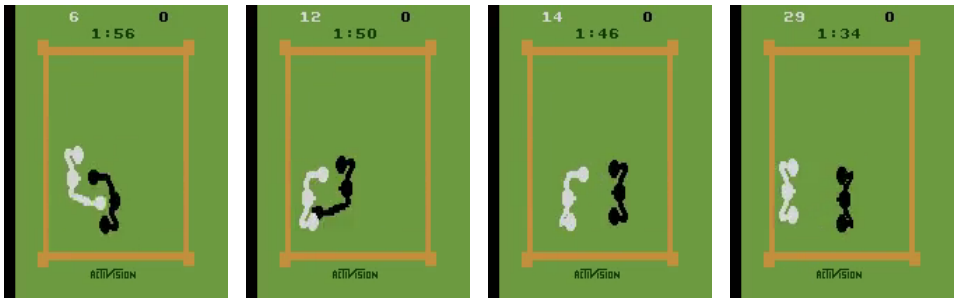
Similar scenario happens in all other environments as well. As an example, we consider Boxing. Figure 3.10 shows the pruned MM and minimal MM, where minimal MM looks tangled and very hard to decode. Pruned MM looks like a straight path, which shows that at key decision points, policy does not strategically rely on observations, instead it relies on memory. Figure 3.11 shows four different observations as branches of S_5 in the minimal MM. Figure 3.11a is where agent is hitting the opponent, Figure 3.11b agent is being hit, and in Figure 3.11c and d there is a distance between the two players. This shows how different each observation is, in terms of interpretation. But minimal MM counts them as semantically relevant states which is misleading. Figure 3.12 and Figure 3.13 demonstrate similar properties for SeaQuest. And Figure 3.14 and Figure 3.15 illustrate them for SpaceInvaders.



(a)

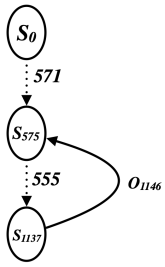
(b)

Figure 3.10: Boxing: a) pruned MM, b) minimal MM.

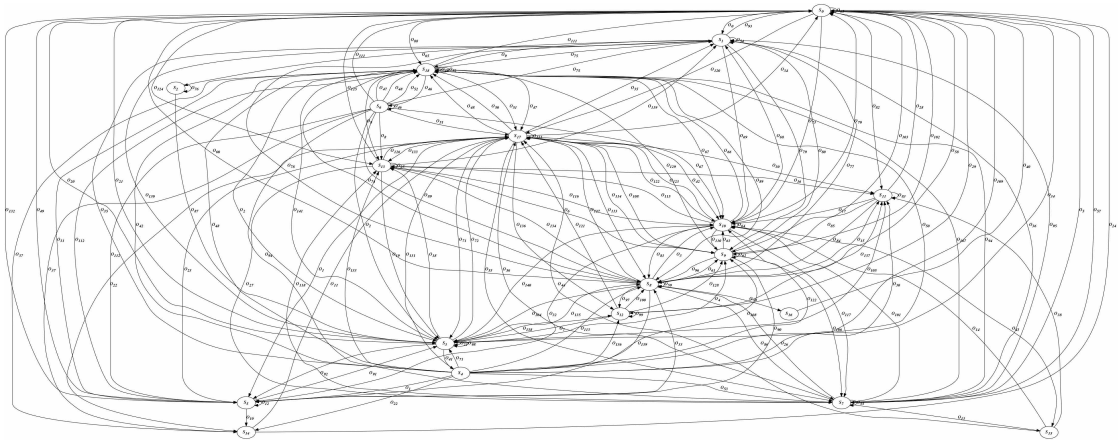


(a) (b) (c) (d)

Figure 3.11: Different observations as branches of decision point S_5 in minimal MM.



(a)

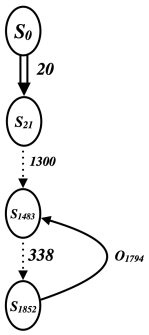


(b)

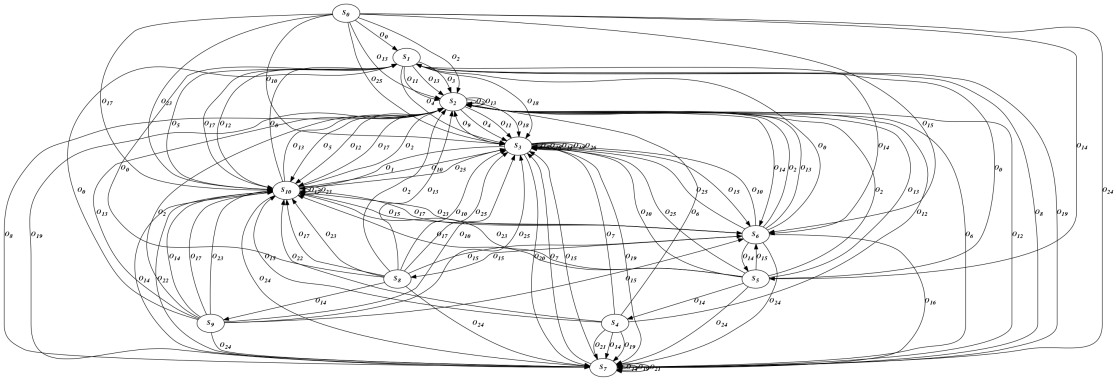
Figure 3.12: SeaQuest: a) pruned MM, b) minimal MM.

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

Figure 3.13: Different observations as branches of decision point S_6 in minimal MM.

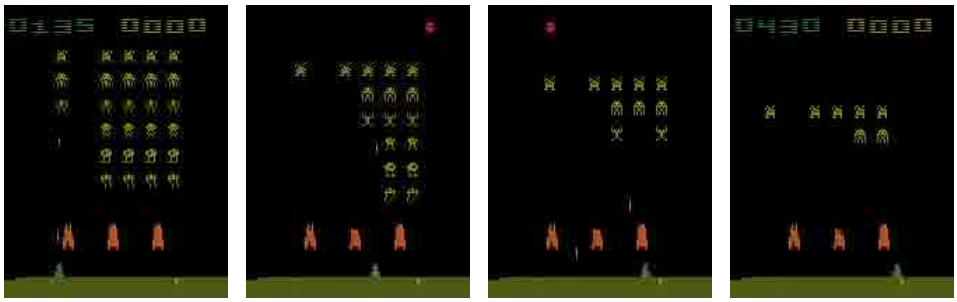


(a)



(b)

Figure 3.14: SeaQuest: a) pruned MM, b) minimal MM.



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

Figure 3.15: Different observations as branches of decision point S_6 in minimal MM.

3.4.4.6 Overall Results

From the qualitative comparisons of Atari games, one could see that they all provide similar distinctions which shows our approach's advantage over [13]. Table 3.1, gives information about the MMs before and after functional pruning for each game. Note that 0 decision points indicates an open-loop policy. We see that before functional pruning there is only one case of open-loop policies: Boxing. All other MMs have at least one decision point. This initially makes one to believe that observations are a key part of the overall MM strategy. However, in each such case, we found that it was rare to find a decision point where observations provide strategic values at a decision point. This was confirmed by our most striking finding. After functional pruning, each of the games resulted in open-loop MMs (i.e. zero decision points). Thus, in all cases, Atari RPNs produced MMs that were pruned open-loop policies. Since Moore Machines are large, we uploaded all of them here: https://tinyurl.com/y96d8jub for better accessibility and understanding their details.

3.4.5 Stochastic Classic Control Tasks

3.4.5.1 Acrobot

This control task includes two joints and two links, where the joint between the two links is actuated. Initially, the links are hanging downwards, and the goal is to swing the end of the lower link up to a given height. The state vector gives the sin and cos of the two rotational joint angles and the joint angular velocities. The actions involved are -1,0 or +1 torque. We share pruned state machine for QBN sizes (4, 4) in Figure 3.16b. We applied our differential attention approach to decision point S_2 and found that the most important observation features were 'sin of joint angle 1" and "joint 2 velocity". The Figure 3.16b shows the scatter plot of the decisions at S_2 versus these features. The plot reveals that for positive sin values, a torque of -1 is applied by starting in S_2 forcing link 1 and link 2 to rise against gravity. This torque is applied until it transitions to S_3 via O_1 which corresponds to a positive velocity for joint 2 as shown in the scatter plot. This happens when link 2 cannot go further up against gravity under the current momentum. In this state, it applies +1 torque to supplement the momentum provided by the pull of gravity. It transitions back to S_2 via O_0 only when the joint velocity of link 1 is positive,

indicating that it cannot go further up. This loop between state S_2 and S_3 generates enough momentum to eventually reach the goal.

In contrast, we compare against the minimal MM (Figure 3.4d) extracted via prior work [13]. This MM, is fully connected and merges many different types of observations. Through various previous attempts, we were unable to elicit a clear description of the machines operation. Rather, our reduction approach was able to lead to the above relatively clear understanding of the machine.

3.4.5.2 CartPole

We use the standard OpenAI CartPole environment with randomized initial states. Table 3.1 shows that the number of decision points is 6 before functional pruning and reduces to 2 after. The pruned MM is shown in Figure 3.16a, which has the same simple structure as for Acrobot. The machine primarily transitions between S_2 (left movement) and S_3 (right movement) with self-loops at those states between transitions. The figure shows the differential attention computed over the features at S_2 and S_3 with the key features being "pole-velocity" followed by "pole-angle". The scatter plots for the decisions against these features show that at S_2 there is a clear threshold of positive pole-velocity that transitions to S_3 to take the "right" action, and otherwise continues with "left". This is an intuitive strategy for reducing the velocity. Similar insights are gained via the scatter plot at S_3, but here both pole-velocity and pole-angle play a role in the decision. Again, our approach was able to produce an MM that has meaningful interpretation.

3.4.5.3 Lunar Lander

This task involves landing a rover using actions that fire one of three thrusters: main, left, right, and no-op. The MMs for both QBN sizes were significantly larger than for the above tasks, likely due to increased complexity and stochasticity. Functional pruning was ineffective, leaving 184 and 230 decision points for the two QBN sizes, which indicates the observations have a strong strategic role. We analyzed many of the decision points and found that the "distance to the ground" is usually the most salient feature across decision points. One exception is the initial decision point. At this decision point

there were three branches with different observations leading to states with different fire actions *main engine*, *right engine* and *left engine*. The attention comparison of these observations shows prime differences in x-velocity and y-velocity, thereby opposite direction engines are fired to stabilized the rover. This decision point does not attend to rover coordinates (x,y position) and leg-positions, which are more relevant when the lander is closer to the ground. While it is difficult to articulate a concise description of such a machine, we see that this analysis approach is able to provide insights about the decisions that may help build confidence or identify concerns.

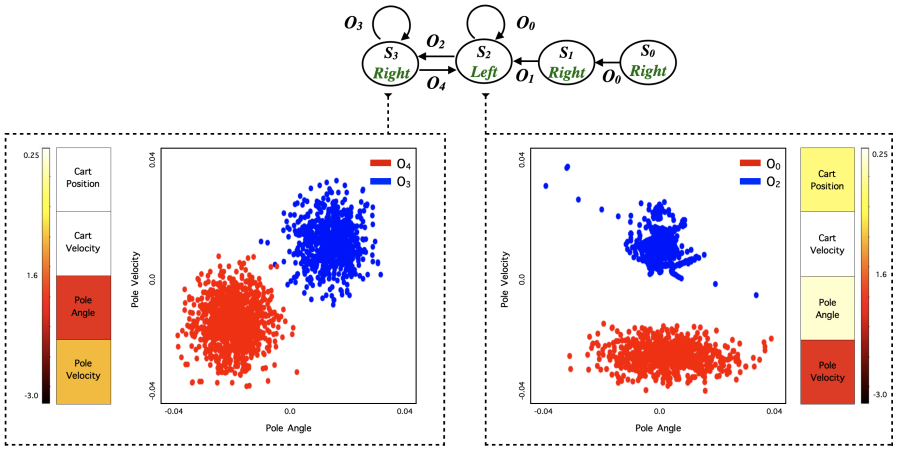
Regarding the minimal MM, due to its big size, it was not possible to be added here. Minimizing MM adds a lot of unexpected and unfavorable complexities to the MM, which makes it uninterpretable, while giving no information on agent's behavior. Functional pruning does not provide much insights into the MM of LunarLander, but our proposed differential saliency tool gives very interesting insights. To the best of our knowledge, this level of insight has not been given in any prior work.

3.5 Summary

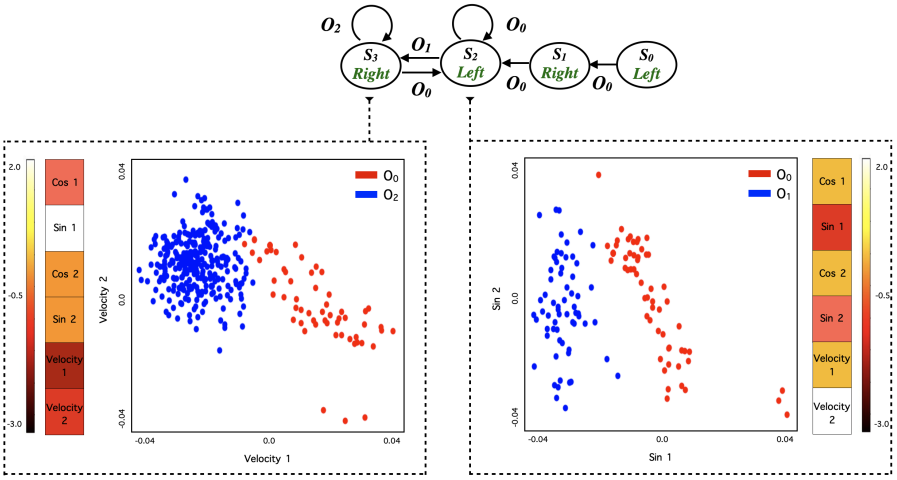
Our analysis is the first to provide such detailed insights into the decision making of RPNs for deterministic Atari games. Indeed, the observation that all of the considered policies resulted in pruned open-loop controllers was unexpected apriori and not apparent from prior work. For example, prior work on attention analysis of Atari policies, even for the deterministic setting, leaves one to believe that observations play key roles in decision making. It is tempting to discount the above insights, due to the deterministic setting, since 1) it is clear that there exist open-loop controllers for any deterministic domain, and 2) prior work has shown that search is able to find effective open-loop plans for some Atari games [2, 14]. However, these points do not imply that an RPN would necessarily learn an open-loop controller and indeed we observed that they do not. It is reasonable to expect that RPNs would meaningfully use observations to get a more general policy, but we instead saw the role for observations was very different. This demonstrates the importance of developing a variety of tools to reveal insights that rely less on human assumptions and interpretation.

Our preliminary experience with larger and more complex environments shows that sometimes our approach does not reveal easily analyzable MMs. This was apparent

in stochastic Atari experiments, where we observed that the discrete sequences produced across different episodes have some overlap, but are dominated by large numbers of disjoint states. Some potential explanations for this are: 1) An inadequacy of our approach–e.g., the quantization process and/or reduction steps/analysis may need to be improved; 2) Our approach may be identifying the fundamental nature of the learned RNN policies. That is, the policies may effectively use large numbers of trajectories to encode large numbers of effectively (pruned) open-loop patterns. As new observations are encountered the machines then map to encoded patterns in a nearest neighbor style. This second possibility would suggest the need for improved RPN training approaches to support data efficiency and interpretability.



(a)



(b)

Figure 3.16: Pruned MMs for control tasks. a) CartPole, and b) Acrobot. In each case, we show attention of features for decision points. Also, we show scatter plots of continuous observations during an episode at each decision point for the two most salient features, where color indicates the discrete machine observation.

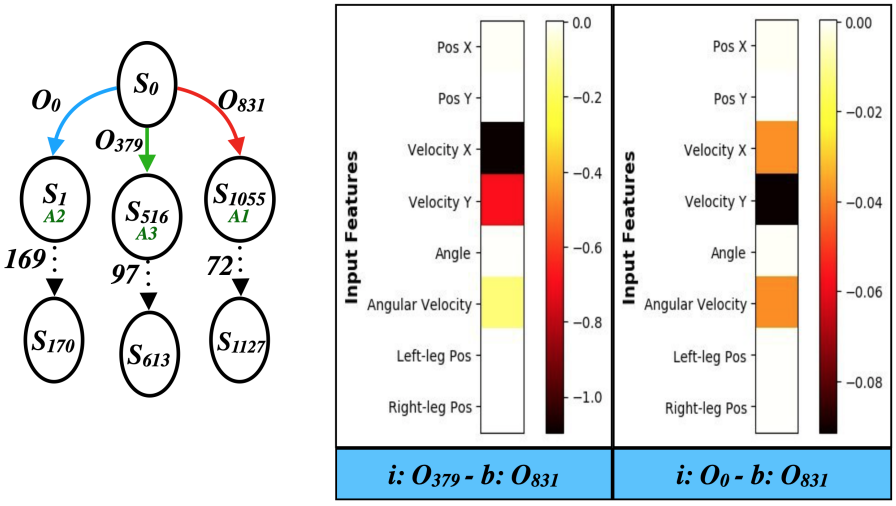


Figure 3.17: Pruned MMs for LunarLander. Next to the pruned MM it the saliency of features for first decision point. For each saliency map, input image and baseline image are shown.

Chapter 4: Conclusion

We have introduced an approach to extract pruned open-loop controllers from recurrent policy networks in order to provide insights into how they utilize observations and memory. The pruned open-loop controllers show that observations do not play a strategically important role to the intelligent agent. We test our approach on seven games from the Atari 2600 environment and three classic continuous control tasks. In all case studies, we demonstrate the effectiveness of our approach and how it contradicts prior works on decision-making agents' interpretability. Our results indicate that prior methods on understanding the behavior of memory-based agents could be misleading and false. Thus, we need to be very careful about the explanations we provide for such intelligent agents. Explanations should be well-defined and well-articulated because humans are very good at misinterpreting them. The misinterpretation may come from many different sources, for example, prior human knowledge on how a task should be done. Eventually, if we want to build trust in such intelligent agents using provided explanations, they need to be sound and trustworthy.

4.1 Future work

In the end, there are a few important questions unanswered with our approach that are an open area of investigation. First, we would like to study how our method could be extended to more complicated domains beyond Atari 2600 games and control tasks. As a starting point, one could investigate the effectiveness of our method on environments with higher stochasticity. It may not result in a pruned open-loop controller, but it gives a very good understanding regarding the importance of given observations at decision points. Also, the tools proposed and provided to analyze finite-state machines could be helpful to extract more information and insight from such complex policies.

Bibliography

[1] Akanksha Atrey, Kaleigh Clary, and David Jensen. Exploratory not explanatory: Counterfactual analysis of saliency maps for deep {rl}. In *International Conference on Learning Representations*, 2020.

[2] 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, 2013.

[3] A. L. Cechin, D. Regina, P. Simon, and K. Stertz. State automata extraction from recurrent neural nets using k-means and fuzzy clustering. In *23rd International Conference of the Chilean Computer Science Society, 2003. SCCC 2003. Proceedings.*, pages 73–78, Nov 2003.

[4] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation, 2014.

[5] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling, 2014.

[6] Samuel Greydanus, Anurag Koul, Jonathan Dodge, and Alan Fern. Visualizing and understanding Atari agents. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1792–1801, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[7] Piyush Gupta, Nikaash Puri, Sukriti Verma, Dhruv Kayastha, Shripad Deshmukh, Balaji Krishnamurthy, and Sameer Singh. Explain your move: Understanding agent actions using focused feature saliency. In *International Conference on Learning Representations*, 2020.

[8] Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. In *2015 AAAI Fall Symposium Series*, 2015.

[9] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning, 2016.

[10] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[11] Sandy H Huang, Kush Bhatia, Pieter Abbeel, and Anca D Dragan. Establishing appropriate trust via critical states. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3929–3936. IEEE, 2018.

[12] Rahul Iyer, Yuezhang Li, Huao Li, Michael Lewis, Ramitha Sundar, and Katia P. Sycara. Transparency and explanation in deep reinforcement learning neural networks. *CoRR*, abs/1809.06061, 2018.

[13] Anurag Koul, Alan Fern, and Sam Greydanus. Learning finite state representations of recurrent policy networks. In *International Conference on Learning Representations*, 2019.

[14] Marlos C Machado, Marc G Bellemare, Erik Talvitie, Joel Veness, Matthew Hausknecht, and Michael Bowling. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *Journal of Artificial Intelligence Research*, 61:523–562, 2018.

[15] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016.

[16] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[17] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[18] Alex Mott, Daniel Zoran, Mike Chrzanowski, Daan Wierstra, and Danilo J. Rezende. Towards interpretable reinforcement learning using attention augmented agents. *CoRR*, abs/1906.02500, 2019.

[19] Matthew Olson, R. Khanna, Lawrence Neal, Fuxin Li, and Weng-Keen Wong. Counterfactual state explanations for reinforcement learning agents via generative deep learning. *Artificial Intelligence*, In Press, 2021.

[20] Marvin C Paull and Stephen H Unger. Minimizing the number of states in incompletely specified sequential switching functions. *IRE Transactions on Electronic Computers*, pages 356–367, 1959.

[21] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3319–3328. JMLR. org, 2017.

[22] Peter Tiňo, Bill G Horne, C Lee Giles, and Pete C Collingwood. Finite state machines and recurrent neural networks—automata and dynamical systems approaches. In *Neural networks and pattern recognition*, pages 171–219. Elsevier, 1998.

[23] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Thirtieth AAAI conference on artificial intelligence*, 2016.

[24] Abhinav Verma, Vijayaraghavan Murali, Rishabh Singh, Pushmeet Kohli, and Swarat Chaudhuri. Programmatically interpretable reinforcement learning. In *International Conference on Machine Learning*, pages 5045–5054. PMLR, 2018.

[25] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Van Hasselt, Marc Lanctot, and Nando De Freitas. Dueling network architectures for deep reinforcement learning. *arXiv preprint arXiv:1511.06581*, 2015.

[26] Gail Weiss, Yoav Goldberg, and Eran Yahav. Extracting automata from recurrent neural networks using queries and counterexamples, 2017.

[27] Gail Weiss, Yoav Goldberg, and Eran Yahav. Learning deterministic weighted automata with queries and counterexamples, 2019.

[28] Zhao Yang, Song Bai, Li Zhang, and Philip H. S. Torr. Learn to interpret atari agents. *CoRR*, abs/1812.11276, 2018.