Appendix

A. 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 it's 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
```

B. 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.

C. 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)($\lambda = 1.0$).

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.

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.

QBNs. 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.

D. Quantitative analyses

Table 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 task. We have not included the detailed information about each interpretable reduction step, Section 4.1, 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 original MM.

According to Table 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 paper, Section 5.1, it is because minimal MMs integrate the decision points with unrelated states. This strongly obscures the ability to interpret agent's decision making process.

E. Qualitative analyses

Since Moore Machines are large, we uploaded all of them here: https://tinyurl.com/y96d8jub for better understanding of their details.

E.1. Atari: Case Studies

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 1a. 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 1b, 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 1c, gives no insights about the agent's behavior. Since semantically unrelated states are matched to each other, complexity is even increased. Figure 2 shows an example of this introduced complexity. All four frames in Figure 2 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 2a shows a frame early in the game, where there are lots of rewards available and ghosts are not out completely. Figure 2b 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 2d 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.

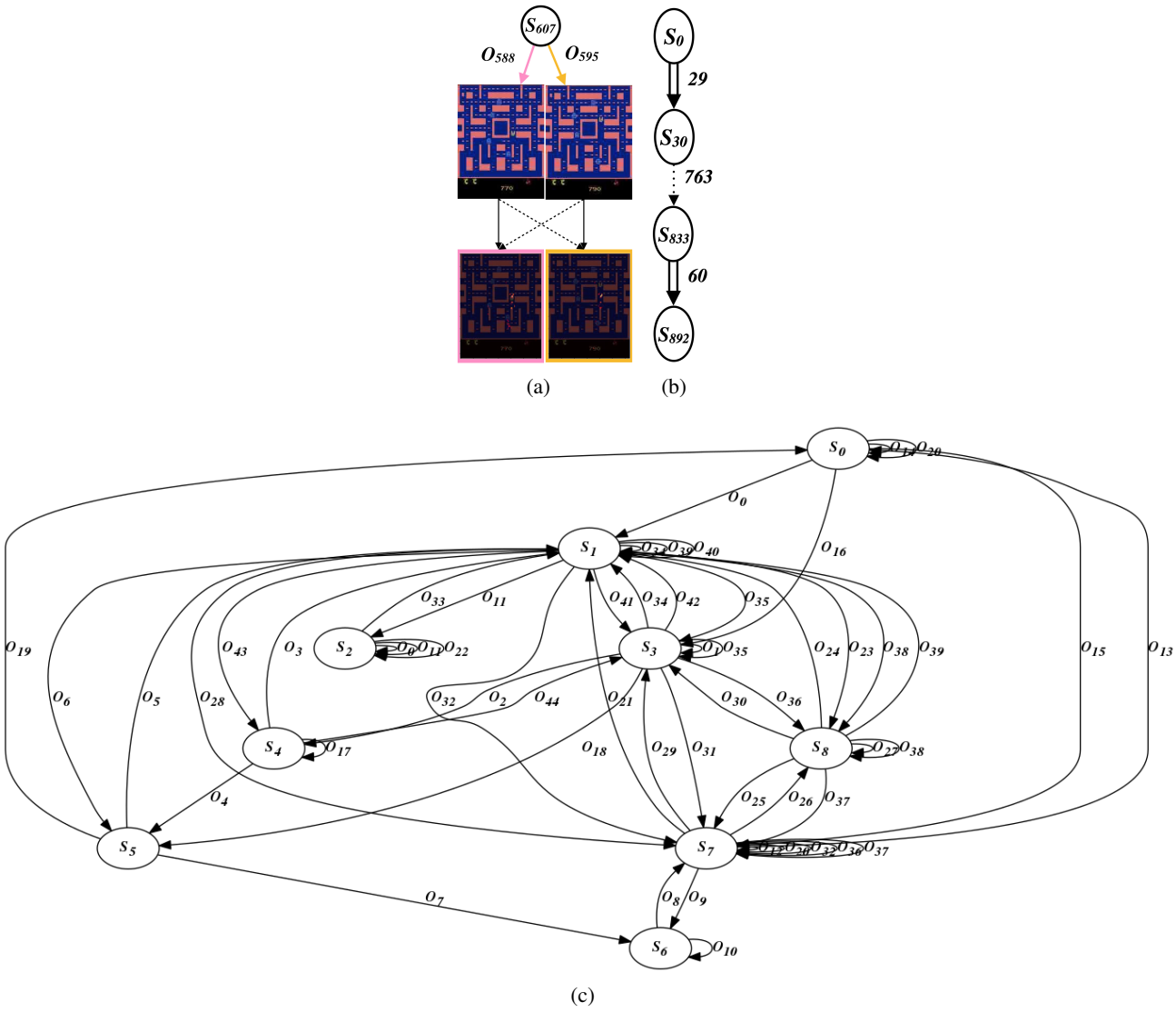


Figure 1. MsPacman: a) Differential saliency for a sample decision point. The first row shows a sample observation that occurs for each branch. The second row gives differential saliency for pairs of observations (dotted arrows indicate the baseline), b) pruned MM, c) minimal MM.

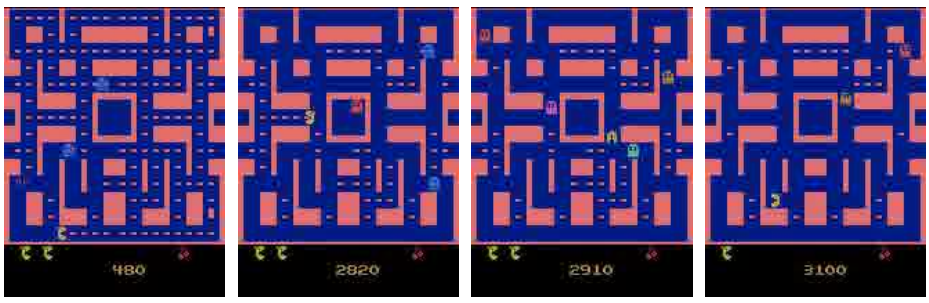
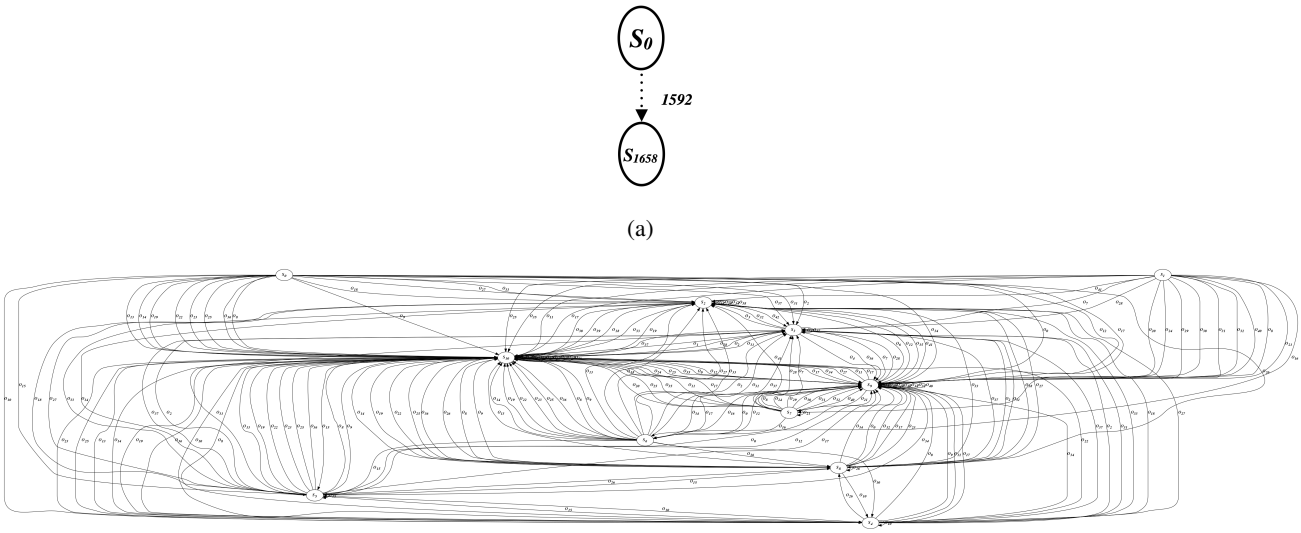


Figure 2. Different observations as branches of decision point S_3 in minimal MM.

Breakout. QBN pair of (64, 100) is considered in this game as well. In Figure 3a, pruned MM, and in Figure 3b, 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.

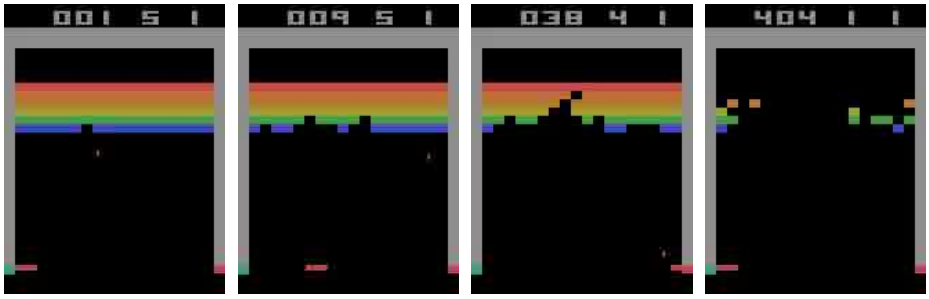
Figure 4 shows four semantically different observations that are turned into branches of a state in the minimal MM. This set of various observations are the outgoing branches of decision point S_9 in the corresponding minimal MM.



(a)

(b)

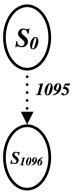
Figure 3. Breakout: a) pruned MM, b) minimal MM.



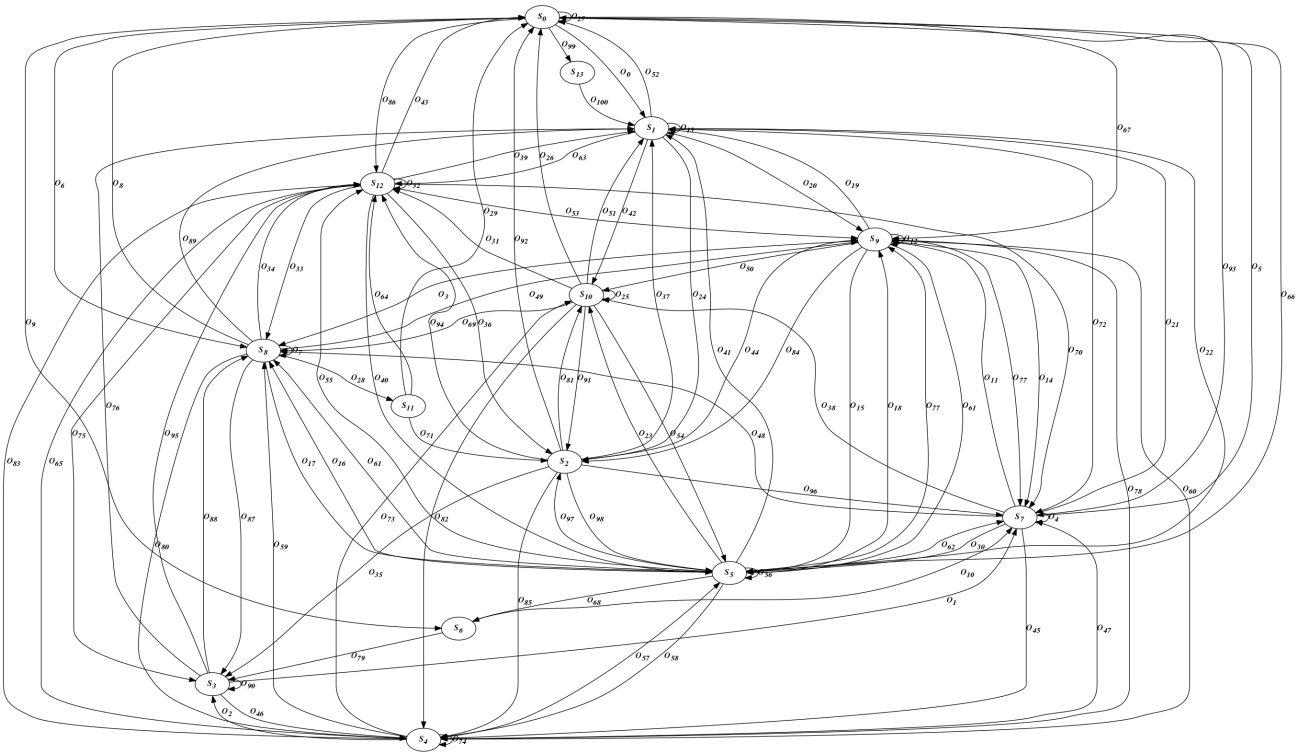
(a) (b) (c) (d)

Figure 4. Different observations as branches of decision point S_9 in minimal MM.

Other games (Boxing, SeaQuest, SpaceInvaders). Similar scenario happens in all other environments as well. As an example, we consider Boxing. Figure 5 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 6 shows four different observations as branches of S_5 in the minimal MM. Figure 6a is where agent is hitting the opponent, Figure 6b agent is being hit, and in Figure 6c 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 7 and Figure 8 demonstrate similar properties for SeaQuest. And Figure 9 and Figure 10 illustrate them for SpaceInvaders.

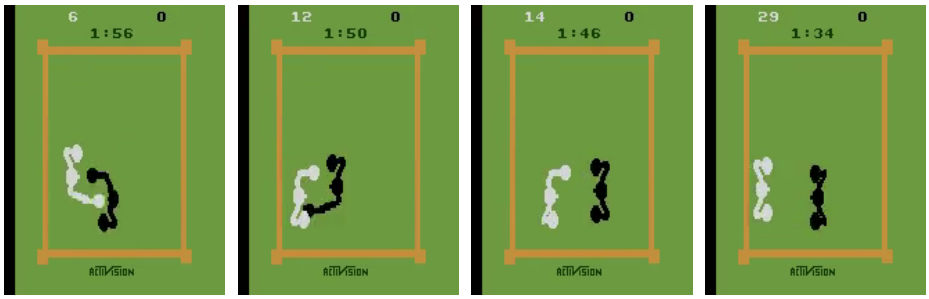


(a)



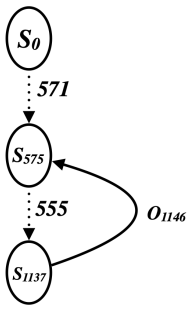
(b)

Figure 5. Boxing: a) pruned MM, b) minimal MM.

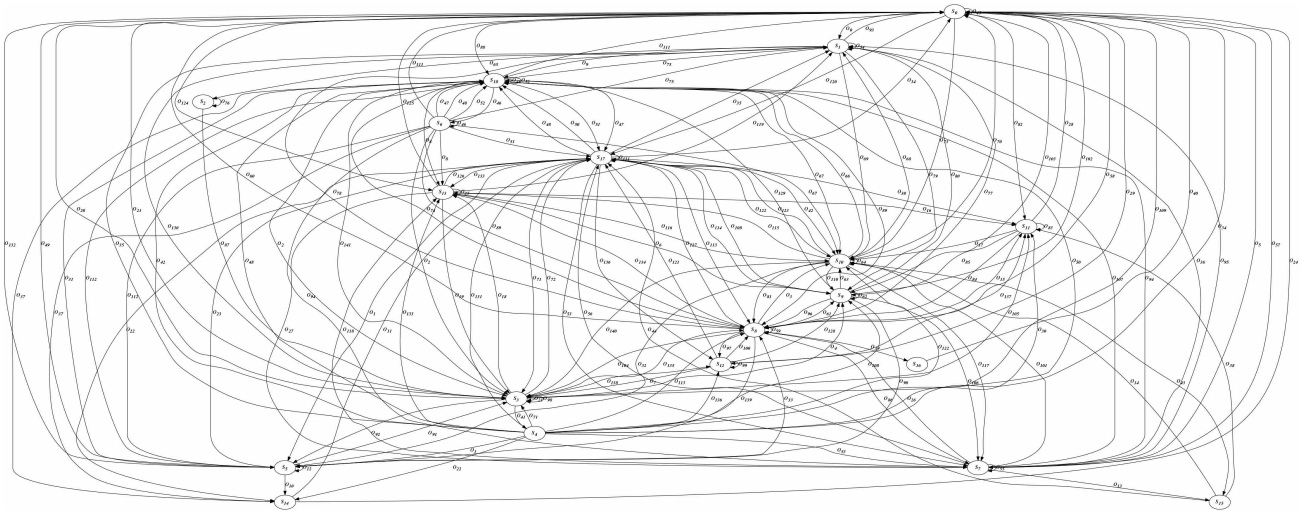


(a) (b) (c) (d)

Figure 6. Different observations as branches of decision point S_5 in minimal MM.



(a)

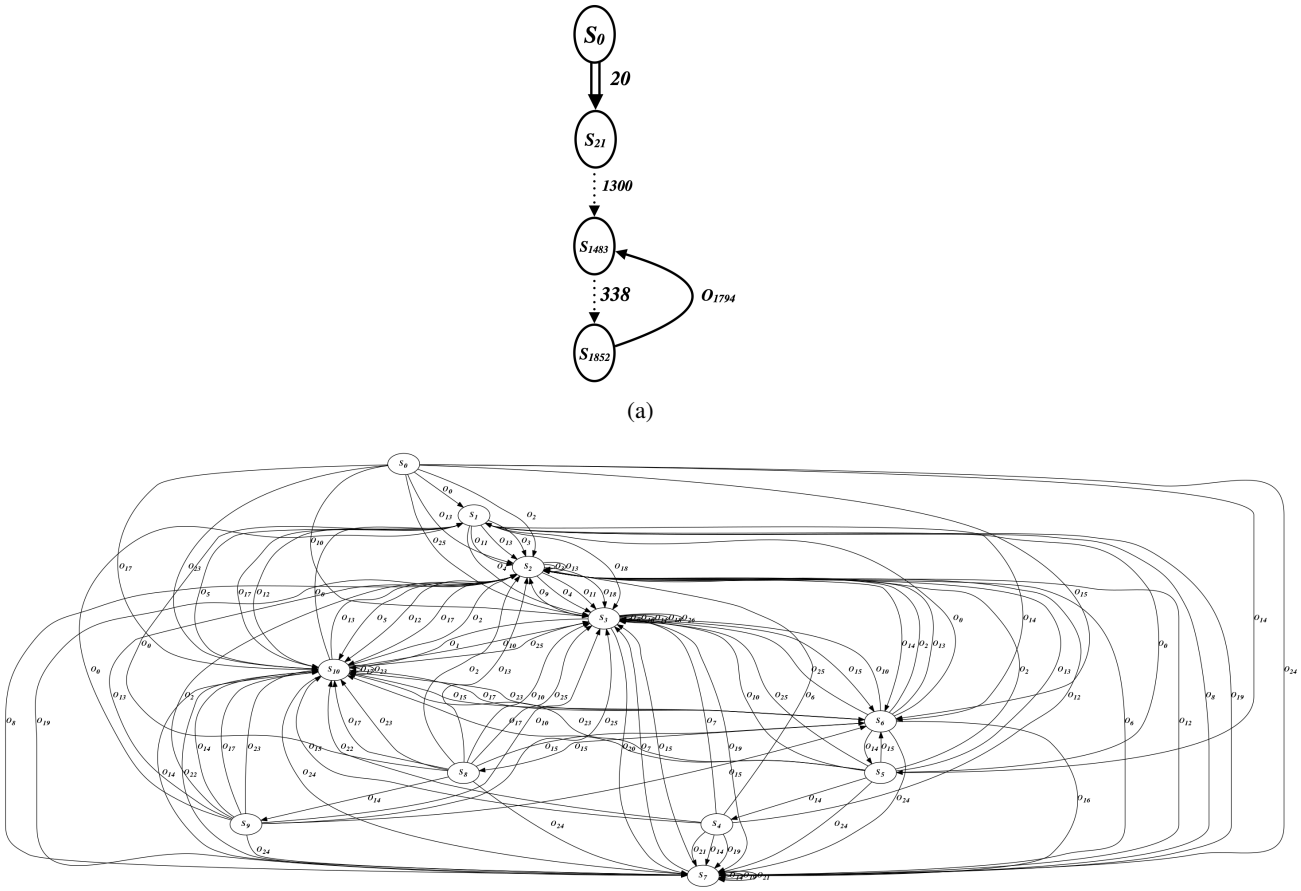


(b)

Figure 7. SeaQuest: a) pruned MM, b) minimal MM.

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

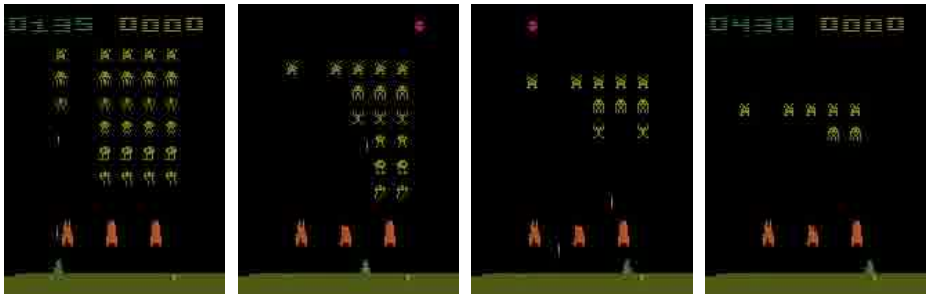
Figure 8. Different observations as branches of decision point S_6 in minimal MM.



(a)

(b)

Figure 9. SeaQuest: a) pruned MM, b) minimal MM.



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

Figure 10. Different observations as branches of decision point S_6 in minimal MM.

E.2. Stochastic Classic Control Tasks

LunarLander. 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.

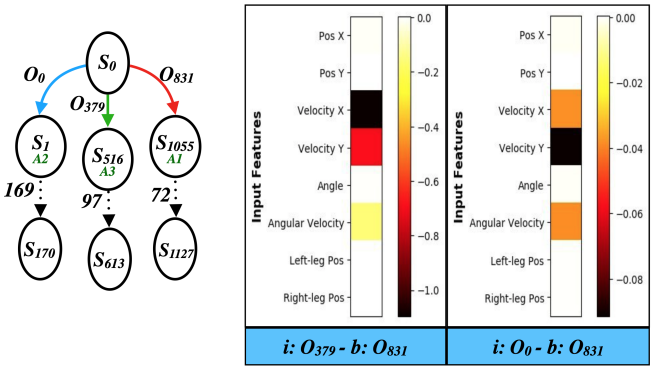


Figure 11. Pruned MMs for LunarLander. Next to the pruned MM it the saliency of features for first decision point. For each saliency map, input image and baseline image are shown.

F. Interpreting another RL policy

To study the behavior of a more recent RL policy, we apply the same steps of our method to a policy learned by the R2D2 algorithm (Kapturowski et al., 2018). To be consistent with the A3C algorithm, which is studies in the main paper, we have replaced the LSTM module of R2D2 with a GRU network. R2D2's policy outperforms A3C in every environment we studied, and consequently, the policy's behavior is different from A3C's. However, after applying our method and studying the resulting MMs, we obtained insights that align with those extracted from the A3C policy. Table 1 provides detailed quantitative results for the R2D2 learning algorithm case.

Table 1. MM results obtained from the R2D2 RL algorithm.

Game	QBN Sizes		Original MM				Functional pruning				Minimal MM			
	N_h	N_o	DP	States	Obs.	Perf.	DP	States	Obs.	Perf.	DP	States	Obs.	Perf.
Bowling	32	50	3	781	761	19	**0**	619	511	**19**	7	7	12	19
	64	100	2	718	709	18	**0**	640	501	**18**	4	4	10	18
Boxing	32	50	**0**	1320	1307	100	**0**	1320	1307	**100**	15	15	119	100
	64	100	**0**	991	980	100	**0**	991	980	**100**	15	15	121	100
Breakout	32	50	2	3391	3312	758	**0**	3110	3091	**758**	14	14	37	758
	64	100	3	2019	1971	763	**0**	1671	1620	**763**	19	19	39	763
Pacman	32	50	21	1201	1171	10k	**0**	982	921	**10k**	27	27	81	10k
	64	100	25	1181	1151	10k	**0**	901	811	**10k**	19	19	82	10k
Pong	32	50	**0**	291	277	21	**0**	291	277	**21**	3	3	15	21
	64	100	**0**	309	291	21	**0**	309	291	**21**	4	4	14	21
SeaQuest	32	50	28	2819	2781	381k	**0**	1872	1794	**381k**	23	23	191	381k
	64	100	19	2711	2618	390k	**0**	1896	1813	**390k**	21	21	205	390k
Space Invaders	32	50	63	1802	1769	51k	**0**	1562	1523	**51k**	28	28	59	51k
	64	100	42	2182	2001	52k	**0**	1301	1232	**52k**	9	9	29	52k