Understanding Transparency and Interpretability in Deep Reinforcement Learning Applications

Shipra Dureja
Faculty of Computer Science
Otto-von-Guericke University
Magdeburg, Germany
shipra.dureja@st.ovgu.de

Kartik Mudgal
Faculty of Computer Science
Otto-von-Guericke University
Magdeburg, Germany
kartik.mudgal@st.ovgu.de

Tarun Gupta
Faculty of Computer Science
Otto-von-Guericke University
Magdeburg, Germany
tarun.gupta@ovgu.de

***Abstract*—In recent years, the combination of Reinforcement Learning (RL) with Deep Learning (DL) has led to remarkable results in a variety of applications such as robotics, or the full spectrum of Atari games. While deep RL agents are effective at maximizing rewards, they are highly sample inefficient and it is often unclear how or what they focus on to converge to the strategies they use. In this paper, we take a principled step towards increasing the sample efficiency in RL and explaining deep RL agents by using an example-based interpretability method - MMD-Critic. MMD-Critic efficiently identifies prototypes and combinations of prototypes and criticisms from the data, which in turn can be used to explain or train the agents instead of using the entire dataset, hence we seek to understand how the interpretable method could assist sample-efficiency. We complement the result obtained from MMD-Critic by using perturbation-based saliency maps to understand how an agent learns and predicts how its training will proceed. In particular, we apply MMD-Critic and Saliency Maps on the MNIST dataset and the Atari Space Invaders environment to show: how the MMD-Critic can be used to improve the sample efficiency of RL agents by training only on prototypes, how saliency maps can be used to interpret trained RL agents by visualizing which features do strong agents attend to, compared to weaker ones, and how such observations help us predict further training performance. We also evaluate the effects of different network architectures on the performance of the RL agents. Overall, our results suggest that prototypes improve sample efficiency by a great margin and saliency brings more transparency to neuron concentration, allowing to understand the locus of attention of models in an image.**

***Index Terms*—Reinforcement Learning, Interpretability, Saliency Maps, MMD-Critic, MNIST, Space Invaders, Prototypes, Criticisms**

I. Introduction

There are many real-world cases, where computer systems have to perform an optimization of a design or a configuration choice (e.g. the quality of a video stream, given an observed transmission rate). These kind of optimizations can be performed using traditional approaches such as linear programming, quadratic programming, the Newton method, and others, or meta-heuristic non-machine-learning methods methods like swarm intelligence or evolutionary algorithms. These methods are helpful if all parameters are known beforehand and we can build an accurate model of the process. However, this is not always the case for real-world scenarios where there exists a large amount of modeling uncertainty. Furthermore, these approaches are usually slow and inefficient when dealing with a huge space of possible solutions. They are unable to guarantee the optima, and learn from experience, which means that a failure to find an optimum can reoccur.

In recent years, there has been immense development in the field of machine learning (ML) to enhance optimization methods. Examples of such approaches include extensions to traditional optimizers with surrogate models or with reinforcement learning (RL). Though the training period for these approaches needs some effort investments to reach a good performance, once that is achieved, they can be faster, efficient, and more importantly, they can keep learning from experience. Some applications include self-driving cars, robotics, computer systems like databases where RL has been effectively applied for join optimization, resulting in better join costs than PostgreSQL's query cost estimator [1].

Nonetheless, for its practical adoption RL has two core challenges:

- First, RL requires a large amount of data to train on, making it a sample inefficient method. They might require a huge amount of interactions with the environment, to start deciding proper actions based on state. Such a great amount of interactions in the real-world often results in large implementation costs [2]. Even in complex environments in the digital world, e.g. playing the Space Invaders game, the low sample efficiency creates difficulties for learning a good policy. In most of the applications, the environment is dynamic instead of static as assumed in traditional RL algorithms, resulting in high variance [3]. Also, with a large action space, each discrete action has to be well explored, which requires many samples, causing difficulty for RL to be sample efficient.
- Second, most of these approaches have a trade-off between complexity and transparency, i.e. the more the complexity of the model increases, the less certain we are about the internal mechanisms of the model, making them hardly interpretable [4]. The interpretability becomes increasingly important in approaches like RL where the system learns autonomously, as it can be necessary to understand the underlying reasoning for their decisions [5]. While interpretability is well explored for standard ML methods, the particular domain of RL has yet many intricacies to be better understood. The difficulty lies in

their complexity, normally parameterized with thousands if not millions of parameters [6]. Another factor to be considered is the performance-interpretability trade-off. Developing a simple RL model could reduce its performance since its complexity also reduces.

Various approaches have been studied previously to alleviate the problem of sample inefficiency in RL, including the use of different exploration strategies such as Curiosity-driven exploration [7], the use of different optimization techniques for updating policy-like derivative-free optimization [8] and the use of neural networks, environment modeling to manually simulate/construct environment [3] as well as learning jump start models using transfer RL [9]. In the area of interpretable RL, a distinction is made between transparent RL algorithms and post-hoc explainability techniques [10]. Transparent algorithms include explanation through representation learning, simultaneous learning of the explanation and the policy and, explanation through hierarchical goals. Post-hoc methods for explainability include explanation through saliency maps [11], interaction data [12], among others.

In our project, we propose the application of two post-hoc interpretability methods to alleviate the problem of sample inefficiency and interpretability in RL. We select them as representatives of their classes of approaches, in order to study their role in improving RL applications. Specifically our work comprises of two contributions:

- First, we use MMD-Critic [13] to extract prototypes and criticisms and compare how alternative scenarios (using only prototypes, only criticisms and both prototypes & criticisms) for training the RL agent contribute towards the performance of the model while improving sample efficiency.
- Second, we use Saliency maps [11], which interprets the areas of interest of a frame by highlighting important features to compare between learned models, and thereby guides us to improve model performance.

The remainder of the paper is structured as follows: In Sec. II, we provide relevant background about RL with a concentration on Deep Q-learning (DQN), and the interpretable techniques for RL, focusing on saliency maps and MMD-Critic. Further in Sec. III, Prototype Design, we define our research questions. We also define the main procedure and components that we propose to tackle our research questions. In Sec. IV, we provide details about experiments concerning variations in model hyperparameters and architecture, for the approaches selected (MMD-Critic and saliency maps), with the corresponding evaluation results. Finally, we conclude in Sec. V, suggesting some directions for extending these methods to databases in future work.

II. Background

For the scope of our project, we conducted a scoped literature research using the Google Scholar database. We used a combination of keywords for searching and selecting research papers, surveys, journals, and conference papers to fetch some related and background work done in the same field. Our fundamental outlook was to obtain relevant work on 'interpretable reinforcement learning', 'explainable AI' and, 'machine learning'. Further, we made variations over the keywords using search keys like 'transparent' and 'understandable'. We considered the papers based on their relevance and citations. With clear learning environments and datasets being defined, we looked for the work that was more associated with particular applications as we wanted to concentrate on state of the art to carry interpretations further. We examined about 100 papers and narrowed our search to the papers of the last 10 years. We classified papers as surveys, current methods in the field, and application-specific state of the art. Surveys helped us to identify the papers and approaches that require more detailed understanding to explore different interpretable RL methods. In the following areas we introduce the reader to background knowledge based on our literature search, covering an introduction to basics of RL and a review of Interpretability Methods for RL.

A. Reinforcement Learning

RL deals with training an agent that continuously interacts with its environment by performing actions in a given state. With every action, the agent collects a reward and arrives at a different state. This reward could be positive, negative, or zero. The main goal in RL is to learn a strategy or a control policy which would result in the greatest cumulative reward after a series of actions. The learning task can then be precisely defined as the one where the agent has to learn a policy π that maximizes $V^{\pi}(s)$ for all states s. We call such as policy as optimal policy and denote it by π^*. $V^{\pi}(s)$ is called the cumulative reward and is given by

$$V^{\pi}(s_t) = \sum_{i=0}^{\infty} \gamma^i r_{t+i} \tag{1}$$

where r_t is the reward at time t and γ is the discount factor which discounts the contribution of future rewards. The optimal policy can then defined as:

$$\pi^* = argmax_{\pi} V^{\pi}(s) \quad (\forall s\ ) \tag{2}$$

When the expected reward of every action is known, the agent can perform a sequence of actions that will eventually result in the greatest cumulative reward over time. The time-ammortized value of this total reward, for an action given a state, is called the Q-value and the strategy can be formalized as:

$$Q(s,a) = r(s,a) + \gamma max_a Q(s',a) \tag{3}$$

The Q-value of current state is dependent on Q-values of future states. If there are n future states, then we have:

$$Q(s,a) \rightarrow \gamma Q(s',a) + \gamma^2 Q(s",a) + ... + \gamma^n Q(s^n,a) \tag{4}$$

This is a recursive function which can then be written as:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma max_a Q(S_{t+1,a}) - Q(S_t, A_t)] \quad (5)$$

The above equation is called the Bellman equation, where α is the learning rate or step size which determines to which extent new information overrides the old one. However, when the environment has a large number of states, in each state, the agent has to choose amongst thousands of actions. So then, the Q-value cannot be inferred from already explored states. There are mainly two problems that occur: First, since the amount of memory required is directly related to the states, it exponentially increases as the number of states increases. Second, the amount of time demanded to explore each of the states to create the Q-table would be unrealistic.

Deep Q-learning can be selected as an alternative to the traditional approaches for approximating the Q-values. In DQN, the state-action pair is provided as input to the neural network to estimate the expected Q-value for this pair. Alternatively, only the state can be provided as input to DQN without explicitly providing the action, to predict the Q-values. The experiences are stored in a replay memory. At every step, the next action is determined by selecting the action associated with the maximum value, according to the output of the Q-network. This is a regression problem where the loss function is the mean squared error of the predicted Q-value and the target Q-value. We then update the network weights using the Bellman equation.

There are many alternative models to Deep Q-Learning, which can be used in deep RL. We focus in this study on Deep Q-Learning as it is a simple method that is still representative of the overall existing methods.

B. Interpretability in Reinforcement Learning

In ML, sometimes the selected method must explain why the model behaves the way it does. This might especially be important in fields where the results of the model have serious consequences such as finance and healthcare. The levels of interpretability of a model vary from the ability to understand how the model makes decisions, to perceiving the holistic view of the features and each of the learned components. This also includes identifying a set of important features and their interactions. ML requires explanations to provide significant confidence, trustworthiness, informativeness, transferability, and fairness to its audience [10]. These interpretations may differ with the sector of the audience it is addressing and their objective. It is important to consider the notion of the audience as the certainty and efficiency of a model are dependent on the goals and cognitive judgement of the audience [10]. We focus on users and developers as our target audience. Users seek to understand the situation and validate the decisions that were taken for a model. It is ideal for them to adopt techniques that can provide transparency, fairness, and trustworthiness. However, if the audience consists of developers, it is necessary to emphasize more the improvement of the performance or efficiency of the product, an understanding of the role of features, and of the overall functionalities. For such cases, interpretable methods need to attain the goals of informativeness and transferability. For example, it is useful for a developer to train one model for multiple applications without having to engage end users.

Transparent algorithms [10] have a transparent architecture and are inherently interpretable. Models built using these algorithms are called white-box models. Apart from this alternative, most interpretation techniques fall under either model-specific or model-agnostic methods [4]. While the former is only limited to a specific model class, the latter can be used on any machine learning model and are applied after the model has been trained. Surrogate models and saliency maps are examples of this type [14]. Sometimes, methods can also be classified as post-hoc explanation methods since they are applied after the model is trained. Additionally, the scope of interpretability may also vary, which can either be global or local. Global interpretable methods explain the entire general model behavior while the local model provides explanations for only a few decisions.

Like many ML algorithms, RL algorithms also suffer from a lack of explainability. This defect can be highly crippling. RL applications (defense, finance, medicine) need models that explain their decisions and actions to human users [15] that can be accepted by the audience. Deep RL models are complex to debug for developers since they rely on many factors such as environment, reward function, observations encoding, large DL models, and the algorithm used to train the policy. Generally, in machine learning, there is only a single model to explain at a time, however, it is more complicated in RL as we generally want to explain a policy or reasons as to why some action was taken by the agent in a particular state [10]. Thus, an interpretable model could help in fixing problems quicker and exponentially speed up new developments in RL methods.

Given the importance of interpretability for RL, there has been a drastic improvement in recent times in the field of explainable RL. Representational learning algorithms such as State Representation Learning (SRL) [16] aim to build a low-dimensional and meaningful representation of state space by processing high-dimensional raw observation data. Simultaneous learning is another field that uses reward decomposition to improve performance and interpretability of the RL models [17]. Methods based on hierarchical RL and sub-task decomposition [18] consist of a high-level agent dividing the main goal into sub-goals for a low-level agent. First, optimal sub-goals of the low-level agents are learned and then the high-level agents form a representation of the environment that is interpretable by human users. All the above methods fall under transparent algorithms. Apart from transparent algorithms, explanations using saliency maps [10] and explanations through interaction data are some of the recently used post-hoc methods. In the next sub-sections we consider specifically 2 post-hoc methods, saliency maps and MMD-Critic. These methods are representatives based either

on local or global explanations respectively.

1) Saliency Maps: When an RL algorithm learns from images or videos, it can be useful to know which feature of the images holds the most relevant information. These features can be detected using saliency methods that produce saliency maps. Saliency maps can be derived using two different methods. Gradient-based saliency maps [11] are calculated by taking the derivative of the input for the output of interest. Unfortunately, these kinds of saliency maps are not interpretable and produce poor results. Several variants have emerged which try to overcome these drawbacks to obtain more meaningful saliency maps. These variants include Guided Backpropagation [19], Excitation Backpropagation [20] and DeepLift [21]. Gradient-based methods are efficient to compute and have a clear mathematical framework. However, changing an input in the direction of the gradient tends to make the input images unrealistic [11] thereby making it difficult to interpret.

Perturbation-based saliency maps have emerged as an alternative to gradient-based methods. The idea behind them is very simple: identify the part of the input image that is to be altered so that the output changes. Inducing perturbations can be as simple as randomly replacing a part of the input image with a gray square or using masked interpolations between the original image and a random image. Saliency maps are widely used to gain insights into convolutional neural networks. In RL, saliency maps have been recently used to interpret agents playing in the OpenAI Gym environment Atari 2600 games with Asynchronous Actor-Critic [11].

2) MMD-Critic: One of the most popular kind of interpretable methods are example-based explanations, in which, particular instances of the dataset are taken to explain the behavior of ML models when facing the underlying data distribution [4]. MMD-Critic [13] is one such example-based method that uses prototypes and criticisms to explain the model. Prototypes are those data instances that are representative of the entire data and on the contrary, criticisms are those instances that are not well represented by the set of prototypes, in other words, they are different from the rest of the data. MMD-Critic selects those datapoints as prototypes that minimize the discrepancy between the distributions of prototypes and the data distribution. The data points from regions that are not well explained by the prototypes are selected as criticisms. MMD-Critic works by first selecting the number of prototypes and criticisms needed by the developer for the application. Then uses a greedy search algorithm to select prototypes and criticisms based on the above criteria. A kernel function that weights the data points according to their proximity is used to estimate the data density. Based on the density estimates, maximum mean discrepancy (MMD) is used to measure the discrepancy between two distributions. The value for the MMD is given by:

$$MMD^2 = \frac{1}{m^2}\sum_{i,j=1}^{m} k(z_i, z_j) - \frac{2}{mn}\sum_{i,j=1}^{m,n} k(z_i, x_j) + \frac{1}{n^2}\sum_{i,j=1}^{n} k(x_i, x_j) \tag{6}$$

where k is a kernel function that measures the similarity of two points, n is the number of data points (set x) and, m is the number of prototypes expected (set z) in our original dataset. The goal of MMD-Critic is to minimize MMD^2. The closer MMD^2 is to zero, the better the distribution of the prototypes fits the data. MMD-Critic can be used for interpretability in three ways:

- by helping to better understand the data distribution
- by building an interpretable model
- by making a black box model interpretable [4]

The first and second uses from above can be applied by building a model similar to a k-nearest neighbor approach, where a prototype from the set of prototypes that is closest to a new data point is selected such that it yields the greatest value of the kernel function. The prototype itself can then be returned as an explanation for the prediction. The third method of using MMD-Critic for interpretability is to predict the outcome of the prototypes and criticisms using a trained model and then analyzing where the predictions went wrong and identify the examples that the model fails to classify. Hence, pointing towards limitations of the model.

III. Prototype Design

Building on our motivation, and our presentation of the methods that we focus on, we can now define the questions that will guide our research. We intend to address two major **research questions** for our project:

- *To what extent does a representative interpretability method based on prototypes and criticisms (here, MMD-Critic) help us to understand and improve Deep RL agents for basic and advanced problems?*
- *To what extent does a representative interpretability method based on feature importance(here, Saliency Maps) help us to understand and improve Deep RL agents for basic and advanced problems?*

We chose the established problems as our focus is on interpretable methods rather than applications. We carry out the experimentation in two distinct learning tasks. The first learning task selected is MNIST learning task, which consists of images from the National Institute of Standards and Technology (NIST) database. MNIST is a small subset from the NIST database which is a collection of handwritten digits from 0-9. It is widely used for learning techniques and pattern recognition methods. Fig. 1 shows example images from the MNIST dataset. It contains 60,000 training images and 10,000 test images. Each image is of 28x28 pixel resolution, which has been size-normalized and centered. This task was selected

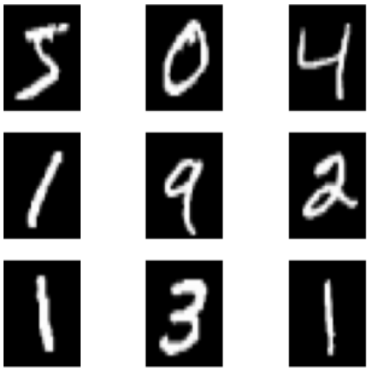

Fig. 1: Example images from the MNIST dataset

Fig. 2: Example image from the Atari Space Invaders Environment

due to its simplicity and common adoption in ML work. The second task is the game Space Invaders, a popular Atari console game. Fig. 2 shows an example frame from the game. The gameplay is discretized into time-steps. At each time step, the agent chooses an action from the set of possible actions for each state (there are 6 actions such as: move left, move right, shoot while staying still, etc.). The emulator then applies this action to the current state and brings the game to a new state. The game score is updated and the reward is returned to the agent.

Generally, RL problems have large Q-tables, the DQN algorithm aids in alleviating this issue. Thus, DQN is used to approximate the Q-function, replacing the need for a look-up table to store the Q-values. Pragmatically, the algorithm uses two deep neural networks to stabilize the learning process. One is the target network that gives the optimal Q-value which is used as the ground truth at every step. The other is the policy network that at each state predicts the Q-value associated with the highest Q-value of an action performed in the next state. This associated Q-value is then compared with the Q-value calculated from the observed reward, the difference is used to update the weights of the policy network. Unlike the policy network, the weights of the target network are updated only after a fixed number of predefined training steps, by copying the weight from the online network. For game environments of the Atari console [11] such as Pong, Frostbite, Enduro, Space Invaders, Breakout, MsPacman, or benchmark environments such as USPS or MNIST, a DQN with a two-network strategy has been commonly used to train RL agents. The architecture usually contains three convolutional layers which are proven to produce efficient results when dealing with images. For our implementation, the architecture for the MNIST environment is inspired by the DQN architecture of the stable baselines3 implementation in PyTorch. We adapt the network and made suitable changes as per our requirements. For Space Invaders, we designed our own architecture based on a similar architecture.

We seek to study post-hoc interpretable techniques to explain the results of both learning tasks. We consider saliency maps and MMD-Critic as follows:

A. Saliency Maps

When an RL agent trained using the DQN network interacts with an environment, images or image features can be extracted at each time step t. Let I_t be the image at time t. Then, we can define the perturbation of pixel (i,j) as $\phi(I_t, i, j)$, which is a blur, localized around (i,j). This blur is constructed using the Hadamard product $\odot$, which interpolates between the original image and the Gaussian blur, $A(I_t, \sigma_A = 3)$, of that image. The interpolation coefficients are given by image mask M(i, j) $\in (0,1)^{m \times n}$ corresponding to a 2D Gaussian centered at μ = (i, j) [11]. The perturbation is then given by:

$$\phi(I_t, i, j) = I_t \odot (1 - M(i, j)) + A(I_t, \sigma_A) \odot M(i, j) \quad (7)$$

If $\pi_u(I_{1:t})$ are the logits of a policy π, then the saliency metric at the location (i,j) can be defined as:

$$S_\pi(t, i, j) = \frac{1}{2} \| \pi_u(I_{1:t}) - \pi_u(I'_{1:t}) \|^2 \quad (8)$$

where

$$I'_{1:k} = \begin{cases} \pi(I_k, i, j) & \text{if k=t} \\ I_k & \text{otherwise} \end{cases} \quad (9)$$

where the difference between the policy of the non-perturbed image and that of the perturbed image approximates the directional gradient of $\pi_u(I_{1:t})$ in the direction of I'$_{1:t}$ [11]. Saliency can then be taken as the square of this directional gradient. Constraining the approximated gradient to be directed in the localized perturbation region overcomes the drawback of gradient-based methods that sometimes do not point towards a meaningful visual direction. In practice, instead of constructing a saliency map for every pixel, generating the saliency maps for a few pixels reduces the computational costs. We calculated the saliency maps for every k = 5 pixels in our setup. Once the saliency maps are extracted, we upsample them and convert them to RGB so that it matches to the input frames size. Fig. 3 showcases the prototype design for saliency maps at a high-level. In general, saliency maps help in the interpretation of the areas of interest of a frame by

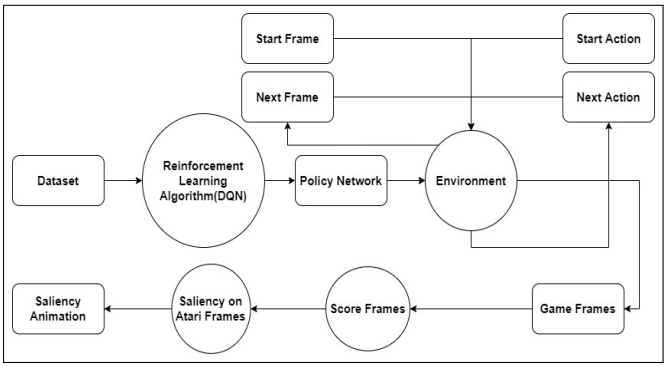

Fig. 3: Prototype Design for Saliency Maps

highlighting important features. For the MNIST dataset, these saliency maps help us to identify the features which removing from the image makes the agent change its learning policy. In the case of Space Invaders, we generate saliency videos by taking consecutive frames and analyzing the strategies and the features that the agent learns. We strive to understand at the end whether saliency maps are able to help us distinguish models of different performance.

B. MMD-Critic

Models can be interpreted using MMD-Critic as follows: MMD-Critic takes as input the number of prototypes, the number of criticisms, and the kernel function to weigh the proximity between two data points. The kernel function helps us to measure the difference between two data distributions by defining a witness function. Once these design decisions are considered, the data is fed to the MMD-Critic algorithm. The MMD criteria select the prototypes that minimize the discrepancy between the data distribution and the distribution of prototypes. Points that are not explained well by prototypes are considered as criticisms. Note that both prototypes and criticisms are data points from the dataset. Fig. 4 showcases the prototype design for MMD-Critic at a high-level. For the MNIST dataset, prototypes and criticisms are images of the digits. These prototypes and criticisms are then used to train a DQN followed by predictions on test set to get the performance. If the prototypes and criticisms are chosen such that they explain the data distribution well, then trained on just prototypes and criticism can achieve comparable performance to that of a model trained on the complete dataset. We seek to evaluate if sample efficiency can be increased by training on prototypes and/or criticisms compared to random data points.

IV. Evaluation

We evaluate the interpretable models for the benchmark environments using variations in network architectures for training RL agents. The evaluation goal is to validate if interpretable models lead to models that are more transparent and less prone to sample inefficiency. The variations of these networks contribute to quantitative and qualitative analysis of both the interpretation techniques.

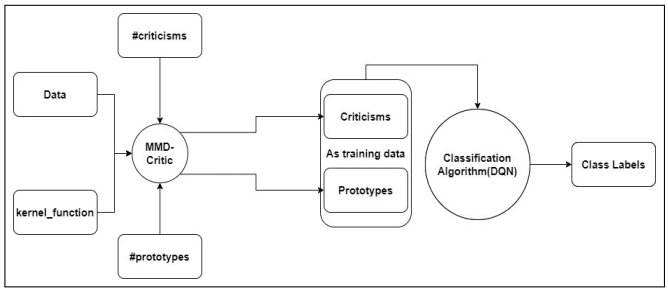

Fig. 4: Prototype Design for MMD-Critic

A. Experiments

All experiments are conducted based on OpenAI Gym [22] specifications. OpenAI Gym is a toolkit to develop and compare RL algorithms by fixing environment implementations.

1) Implementation: Environment: For MNIST, we created a wrapper class based on OpenAI Gym specifications. This class acts as the environment for our agent. This was done, as OpenAI Gym doesn't have a pre-defined environment for MNIST. The states and rewards are provided by the wrapper class. The learning problem for the MNIST dataset can be formalized as:

- State s: a sequence of observations which is a matrix representing the input image or the extracted features of the input image.
- Action a: depends on the input, classifies the image among one of the 10 classes.
- Reward r: reward returned by the environment which is either 1 (for correct classification of the digit) or 0 (for incorrect classification).
- Learning task: Our task is to learn a policy for the agent such that it classifies each image correctly.

For Space Invaders, we used a pre-defined environment from OpenAI Gym API. The states and rewards are provided by the API. The learning problem for Atari Space Invaders can be formalized as:

- State s: A sequence of observations, which depending on the game state, is a matrix representing the image frame of the game.
- Action a: An integer in the range of [1, 6]. where each integer represents the actions FIRE (shoot without moving), RIGHT (move right), LEFT (move left), RIGHTFIRE (shoot and move right), LEFTFIRE (shoot and move left), NOOP (no operation) respectively.
- Reward r: Reward returned by the environment which is the score in the game at the current time step.
- Learning task: Our task is to learn a policy for the agent to enable the agent to make a good choice of action for each state.

2) Implementation: Model:

a) MNIST: The original dimension of an image from MNIST is (28, 28, 1) which is width, height, number of channels respectively. In the preprocessing step, the images are upscaled to dimension (64, 64, 1) which is again the

TABLE I: Hyperparameters for DQNs

	MNIST DQN	Space Invaders DQN
Learning Rate (α)	$1e^{-4}$	$5e^{-4}$
Discount Factor (γ)	0.99	0.99
Initial Exploration Rate (ϵ_{start})	1.0	1.0
Exploration Decay (ϵ_{decay})	0.1	0.996
Final Exploration Rate (ϵ_{end})	0.05	0.01

width, height and, number of channels respectively which then becomes the input of the network. The network architecture for the MNIST environment is shown in Fig. 10 in the Appendix A. Our model consists of three convolutional layers. The first layer consists of 32 filters of kernel size 8 and stride of 4 followed by the second layer consisting of 64 filters of kernel size 4 and stride 2. The last convolutional layer consists of again 64 filters with a kernel size of 3 and a stride of 1. We use the ReLu activation function after each convolutional operation and zero padding for all the layers. This is followed by a flattening of output after convolution followed by one fully-connected layer. The fully-connected layer has n output units where n is 10, which is the dimension of the MNIST action space. Both target and policy networks have the same architecture. The network hyperparameters are mentioned in Table I the target network is updated after every 10,000 time steps.

b) Space Invaders: The original dimension of an image from Space Invaders is (210, 160, 3) which is width, height, and number of channels respectively. The preprocessing consists of first grayscaling the images, then downsampling the images by a factor of 2, and finally cropping the game space to an (80, 80) square which is width and height, and then normalizing the values to [0, 1]. The policy network architecture for Atari Space Invaders is shown in Fig. 11 in Appendix A. This network also consists of three convolutional layers where the number of filters double after each layer starting from 32 and reaching 128 in the last layer. The first layer has a kernel size of 8 and stride of 4 followed by the second layer with kernel size 4 and stride 2. The last convolutional layer consists of a kernel size of 3 and a stride of 1. We used the ReLu activation function after each convolutional operation, padding of 1 for the first layer and zero padding for all other layers. This is followed by a flattening of output after convolution followed by two fully-connected layers. Here, the last fully connected layer has n output units, where n is 6, which is the dimension of the Space Invaders action space. Similar to the MNIST network architecture, both the target and the policy networks share the same architecture. The network hyperparameters are mentioned in Table I, the target network is updated after every 10,000 time steps.

3) Implementation: Model Variations:

a) By Architecture: To qualitatively interpret the results of saliency maps, we introduce variations in the network architecture demonstrated above. Our base model is hereafter called the default model. We injected the default model with small variations to generate two more policy network architectures. For the first, we added dropout units to the default model and named it the dropout model. We created another model by introducing maxpooling layers in our default model and named it the maxpool model.

For the MNIST environment, in the dropout model, we added dropout units after the first and second convolutional layers with moderate dropout probabilities of 0.2 and 0.3 respectively. The maxpool model was created by introducing a maxpool layer of kernel size of 2 and a stride of 2 after the second convolutional layer and a dropout unit with a dropout probability of 0.4 after the third convolutional layer.

In the Space Invaders environment, we create the dropout model by adding dropout units after each successive convolutional layer with a dropout probability of 0.2, 0.3, 0.4 respectively. The maxpool model was created by adding 2 maxpool layers, one after the first convolutional layer with a kernel size of 2 and a stride of 1 and the other after the second convolutional layer with a kernel size of 4 and a stride of 2.

We apply saliency maps on all the model variations to provide a qualitative and quantitative evaluation of the performance and effectiveness of these interpretation techniques. We **hypothesize** that *the saliency of the default model offers a better explanation than both the variations for both environments.*

Further, we also **hypothesize** that *the dropout model will performs better than the maxpool model since maxpool layers down-samples the image features by combing the features at the risk of losing important information although maxpool layers act as a regularizer*. The dropout model might not bring any benefit to our RL use but could be comparable to the default model as it prevents overfitting of the policy however, for the environment in use, we do not encounter the issue of overfitting.

b) By Training Data: For MMD-Critic, we choose to have variations in the interpretable model rather than network architecture by creating variations on the training data. The variations we employ are analyzed to identify the learning capabilities of the model on three separate accounts:

- Training only on Prototypes
- Training only on Criticisms
- Training both on Prototypes and Criticisms

We formulate **3 hypotheses** for these variations:

- *Training only on Prototypes should be able to provide comparable or higher performance than training on the entire dataset.*
- *Training only on Criticisms should be able to provide comparable or higher performance than training on the entire dataset.*
- *Training both on Prototypes and Criticisms should be able to provide comparable or higher performance than training on the entire dataset.*

B. Results

1) Saliency Maps:

a) MNIST: In the images of the MNIST dataset, the most important features are in the center. For the default model, in the saliency, we see that the majority of the time the neurons are acting in coordination, neuron firing is concentrated around the central region where the important features are associated with the digit in the image are present. Further, from the saliency maps of the dropout model, we noticed that for a particular digit image, the network initially has a more diverse locus of focus, but as it encounters more images of the same digit the network concentration starts to improve and converge towards the saliency of the default model. Finally, for the maxpool model, we witness that the network has the most diverse locus among the three models throughout all the images provided for saliency generation. The neuron concentration is unable to align to the central part of the image because of the inherent design of how a maxpool function works. It is unable to capture all the important feature pixels from the image. The saliencies for all 3 models can be seen in Fig. 5.

These results from the saliency maps from all three models can be quantified with the rewards received. When all three models are used to collect rewards for the entire set of 10,000 images over 10 times, the average rewards received are 2842.1, 2800, 2334 for default, dropout, and maxpool models respectively. For all the three models based on saliency maps and the rewards received we can infer that:

- The default model is the best out of all three because of the concentrated saliency on important features and better rewards. Hence, we accept our first hypothesis of default model performing better than the variations.
- The dropout model looks like starting to converge towards default from both saliency and the rewards and starts diverging from the maxpool model with the limitation of images being of low dimension but since the distinction of training and testing data does not change for MNIST, we do not expect the dropout model to converge towards the default model and provide better generalization even after a longer training time. We can say that the downsampling in maxpool hurts the performance whereas dropout probably starts to generalize better. Hence, we can accept the second hypothesis as the dropout model performs better than the maxpool model.

It is worth noting that the acceptance of these hypotheses for MNIST is limited to training the agent for 1 million frames. It is probable that with further training the default model might overfit and the dropout model starts generalizing better. However, with the low-dimension of images, we can presume that the maxpoool model will not converge towards either dropout or default model.

b) Space Invaders: For the default model, in the saliency maps, we see that the majority of the time the neurons are acting in coordination, neuron firing is concentrated around the area where the important features i.e., the aliens in the image are present. Further, from the saliency maps of the dropout model, we noticed that for a particular frame, the network initially has a more diverse locus but as it encounters more images the network concentration starts to improve and converge towards the saliency of the default model, as in the case for MNIST. Finally, for the maxpool model, we see that the network has the most diverse locus among the three models throughout all the images provided for saliency generation. The neuron concentration is unable to align to the most important part of the image i.e., the aliens.

We were not able to quantify these results from the saliency maps from all three models with the rewards received because of the limited model training due to time-constraints. When all three models are used to collect rewards for 100 games played until completion the average rewards received are 285.0 for all default, dropout, and maxpool models.

Saliency becomes an important factor here to interpret the models as there is no strong distinction between the models based on rewards. Based on the visual evidence of the saliency maps, we can say that the maxpool model focuses on less or unimportant portions of the frame. We can infer that the default model is the best out of all the three because of the concentrated saliency on important features and better rewards. So, given the observation, the training is then done for 2 million frames for all the three models, leading us to the following conclusions:

- We confirm our hypothesis that the default model performs better than the variations.
- While training for 300,000 frames, there is divergence seen in the neuron concentration of the default and the dropout model. The default model has a better concentration as compared to the dropout model. However, when trained further up to 2 million frames, the divergence reduces. We observed that the dropout model starts to converge towards the default of saliency, dropout probably starts to generalize better. Hence, we can conclude our second hypothesis is accepted.

It is probable that with further training the default model might overfit and the dropout model starts generalizing better but nothing concrete can be said about the maxpool model. The saliencies for all default, dropout and maxpool models can be seen in Fig. 6, Fig. 7 and Fig. 8 respectively.

2) MMD-Critic:

a) MNIST: For every combination discussed in the experiment section, the network was trained for 60,000 timesteps. When we run the model for the entire training dataset, the test accuracy achieved is 69.85%. The number of prototypes and criticisms chosen are 1500, 2000, 3000, 4000, 4500, and 6000, which add up to 10% of the dataset. A few sample prototypes and criticisms are shown in Fig. 12 and Fig. 13 in Appendix A, respectively. We trained the RL agent on these prototypes and criticisms individually as well as combined. The test accuracies for each permutation is available in the Table II with random datapoints as well as in Fig. 9 and for individual comparisons with random datapoints refer to Fig. 14 in Appendix A.

For the first case, when trained only on prototypes that range from 2.5 - 10% of the dataset, the test accuracies were higher than the test accuracy on the entire dataset. Hence, we accept

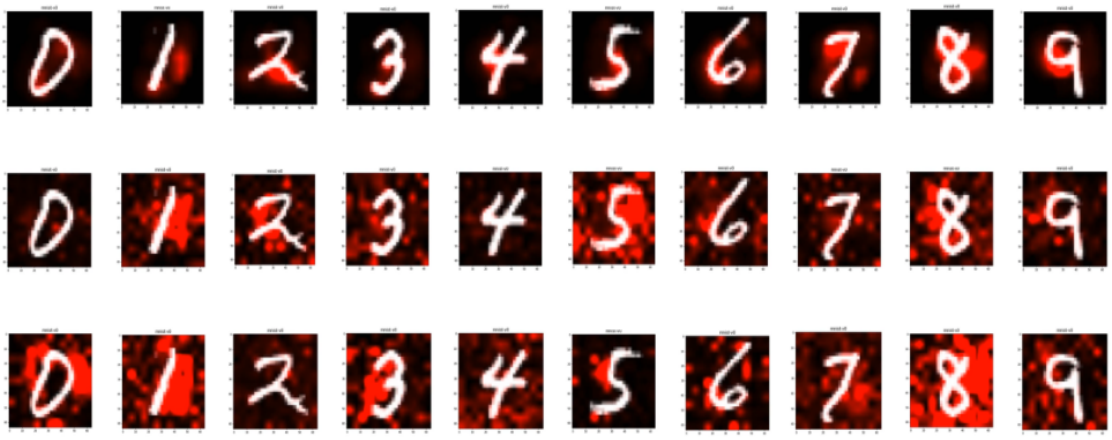

Fig. 5: Comparison of saliency for MNIST in default (top), dropout (middle) and maxpool (bottom) models

TABLE II: Accuracies associated with Number of data points according to their types

Number of Datapoints	Prototypes Accuracy (%)	Criticisms Accuracy (%)	Combined Accuracy (%)	Random Accuracy (%)
1500	71.97	50.46	69.14	63.73
2000	74.38	55.00	72.76	64.60
3000	75.02	58.65	70.06	62.65
4000	76.11	56.37	76.97	61.02
4500	72.51	59.19	74.84	64.86
6000	75.25	60.18	70.47	64.92
9000	Not Applicable	Not Applicable	74.09	65.34
12000	Not Applicable	Not Applicable	73.42	64.24

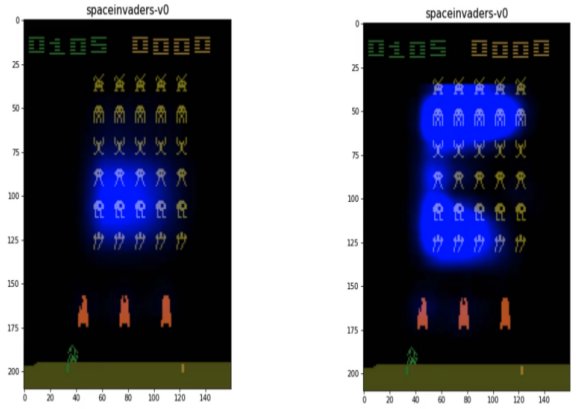

Fig. 6: Saliency for Space Invaders in default model at 300,000(left) and 2 million(right) frames

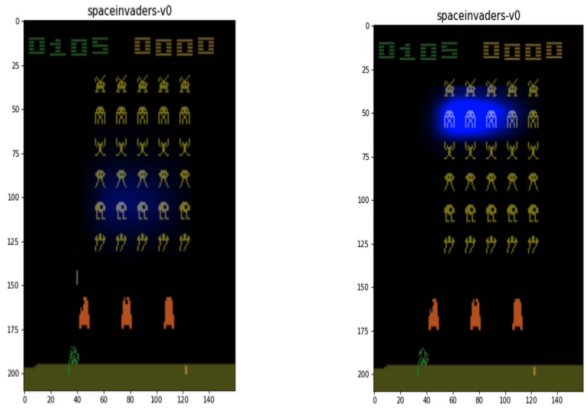

Fig. 7: Saliency for Space Invaders in dropout model at 300,000(left) and 2 million(right) frames

our first hypothesis, which proves that using prototypes that are representative of the entire dataset improves sample efficiency. For the second case, when trained only on criticisms which also ranges from 2.5 - 10% of the dataset, the test accuracies were lower than the test accuracy on the entire dataset. Hence, we reject our second hypothesis, which demonstrates that using only criticisms that consist of data points that are not well represented by the set of prototypes does not represent well the entire dataset and hence, does not improve the sample efficiency. For the third case, for the combination of prototypes & criticisms that range from 2.5 - 20% of the dataset, the test accuracies are higher than the test accuracy on the entire dataset but lower than test accuracies of only prototypes, except on one occasion. Hence, we accept our third hypothesis, but with a grain of salt that using the combination might not always provide the best performance among the three cases.

To provide a baseline for all three cases, we trained the model on the same number of data points in that particular instance by selecting the same number of datapoints randomly and calculating the test accuracy for every possible case. We discovered that when trained only on prototypes and combination of prototypes and criticisms, the performance was better than the random baseline but not for the case when trained only on criticisms. The code for all the implementations is

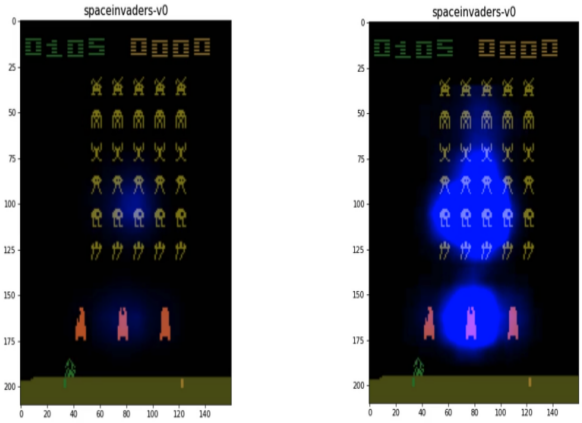

Fig. 8: Saliency for Space Invaders in maxpool model at 300,000(left) and 2 million(right) frames

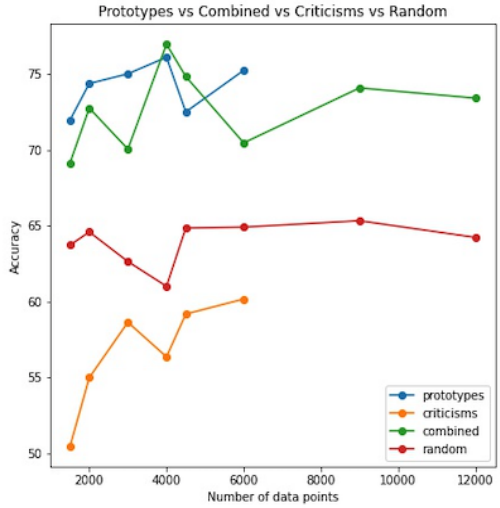

Fig. 9: Test Accuracies for Prototypes vs Criticisms vs Prototypes & Criticisms Combined vs Random

available on GitHub [1,2].

V. Conclusion & Future Work

In this work we analyzed the role of model variations and saliency maps in predicting the performance of RL models on two environments, and considered the use of prototypes and criticisms to improve sample efficiency on a single environment. From our results we can conclude that prototypes improve sample efficiency by a huge factor and saliency brings more transparency to neuron concentration and diversity of locus in the image.

Given that the distinction of training and testing data does not change for MNIST, we do not expect the dropout model to converge towards the default model and provide better generalization even after a longer training time. However, in the case of Space Invaders, the training and the testing data comes from the OpenAI Gym environment and there is always some distinction, we expect the dropout model to converge towards the default model and provide better generalization after longer training time. For MMD-Critic on MNIST, we prove that using prototypes improves sample efficiency by achieving better performance when training on a fraction of the entire dataset. This strengthens the expectation of using prototypes to reduce the training time while not compromising on the performance. In sum our study validates the usefulness of both methods, emphasizing to a higher degree on the use of prototypes and criticisms.

Looking ahead it might be interesting to explore the behavior of other RL algorithms (DDQN, A2C, A3C) in terms of interpretability, transparency, related to their comparative performance. Further studies of MMD-critic concerning its sample efficiency can also be recommended. It is also intriguing to know and explore the use of criticisms to explain the model's behavior in case of divergence from the model generalization. In the original literature of MMD-Critic, there was no universal method mentioned to find the optimal number of prototypes and criticisms for any given problem but in the scope of the environments we used, there is a possibility to further explore the optimal number by using regression techniques. To further cement our hypotheses, longer training of the models as per current practices would be beneficial to understand overfitting in RL agents. Finally, we can suggest that the applications of these techniques to improve RL can impact best practices in RL applications for DBMSs.

[1]https://github.com/tarunlnmiit/idrl

[2]https://github.com/tarunlnmiit/idrl_mmd

References

[1] Marcus, Ryan, and Olga Papaemmanouil. "Deep reinforcement learning for join order enumeration." In Proceedings of the First International Workshop on Exploiting Artificial Intelligence Techniques for Data Management, pp. 1-4. 2018.

[2] Yu, Yang. "Towards Sample Efficient Reinforcement Learning." In IJCAI, pp. 5739-5743. 2018.

[3] Chen, Shi-Yong, Yang Yu, Qing Da, Jun Tan, Hai-Kuan Huang, and Hai-Hong Tang. "Stabilizing reinforcement learning in dynamic environment with application to online recommendation." In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 1187-1196. 2018.

[4] Molnar, Christoph. Interpretable machine learning. Lulu. com, 2020.

[5] Puiutta, Erika, and Eric MSP Veith. "Explainable reinforcement learning: A survey." In International Cross-Domain Conference for Machine Learning and Knowledge Extraction, pp. 77-95. Springer, Cham, 2020.

[6] Brown, Tom B., Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan et al. "Language models are few-shot learners." arXiv preprint arXiv:2005.14165 (2020).

[7] Singh, Satinder, Andrew G. Barto, and Nuttapong Chentanez. Intrinsically motivated reinforcement learning. MASSACHUSETTS UNIV AMHERST DEPT OF COMPUTER SCIENCE, 2005.

[8] Beyer, Hans-Georg, and Hans-Paul Schwefel. "Evolution strategies–a comprehensive introduction." Natural computing 1, no. 1 (2002): 3-52.

[9] Finn, Chelsea, Pieter Abbeel, and Sergey Levine. "Model-agnostic meta-learning for fast adaptation of deep networks." In International Conference on Machine Learning, pp. 1126-1135. PMLR, 2017.

[10] Heuillet, Alexandre, Fabien Couthouis, and Natalia Díaz-Rodríguez. "Explainability in deep reinforcement learning." Knowledge-Based Systems 214 (2021): 106685.

[11] Greydanus, Samuel, Anurag Koul, Jonathan Dodge, and Alan Fern. "Visualizing and understanding Atari agents." In International Conference on Machine Learning, pp. 1792-1801. PMLR, 2018.

[12] Caselles-Dupré, Hugo, Michael Garcia-Ortiz, and David Filliat. "Symmetry-based disentangled representation learning requires interaction with environments." arXiv preprint arXiv:1904.00243 (2019).

[13] Kim, Been, Oluwasanmi Koyejo, and Rajiv Khanna. "Examples are not enough, learn to criticize! Criticism for Interpretability." In NIPS, pp. 2280-2288. 2016.
[14] Adadi, Amina, and Mohammed Berrada. "Peeking inside the black-box: a survey on explainable artificial intelligence (XAI)." IEEE access 6 (2018): 52138-52160.
[15] Gunning, David, and David Aha. "DARPA's explainable artificial intelligence (XAI) program." AI Magazine 40, no. 2 (2019): 44-58.
[16] Bengio, Yoshua, Aaron Courville, and Pascal Vincent. "Representation learning: A review and new perspectives." IEEE transactions on pattern analysis and machine intelligence 35, no. 8 (2013): 1798-1828.
[17] Juozapaitis, Zoe, Anurag Koul, Alan Fern, Martin Erwig, and Finale Doshi-Velez. "Explainable reinforcement learning via reward decomposition." In IJCAI/ECAI Workshop on Explainable Artificial Intelligence. 2019.
[18] Kawano, Hiroshi. "Hierarchical sub-task decomposition for reinforcement learning of multi-robot delivery mission." In 2013 IEEE International Conference on Robotics and Automation, pp. 828-835. IEEE, 2013.
[19] Springenberg, Jost Tobias, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. "Striving for simplicity: The all convolutional net." arXiv preprint arXiv:1412.6806 (2014).
[20] Zhang, Jianming, Sarah Adel Bargal, Zhe Lin, Jonathan Brandt, Xiaohui Shen, and Stan Sclaroff. "Top-down neural attention by excitation backprop." International Journal of Computer Vision 126, no. 10 (2018): 1084-1102.
[21] Shrikumar, Avanti, Peyton Greenside, and Anshul Kundaje. "Learning important features through propagating activation differences." In International Conference on Machine Learning, pp. 3145-3153. PMLR, 2017.
[22] Brockman, Greg, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. "Openai gym." arXiv preprint arXiv:1606.01540 (2016).

APPENDIX A
NETWORK MODEL ARCHITECTURES, SAMPLE PROTOTYPES AND CRITICISMS & INDIVIDUAL ACCURACY COMPARISONS FOR MMD-CRITIC

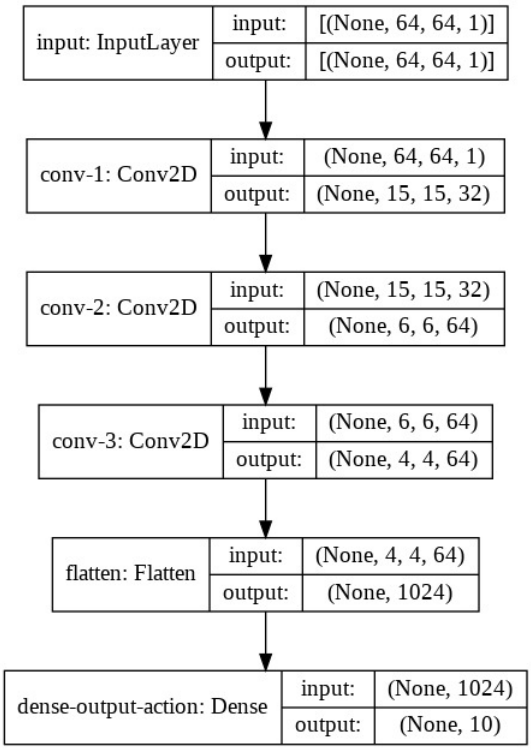

Fig. 10: Policy Network for the MNIST Dataset

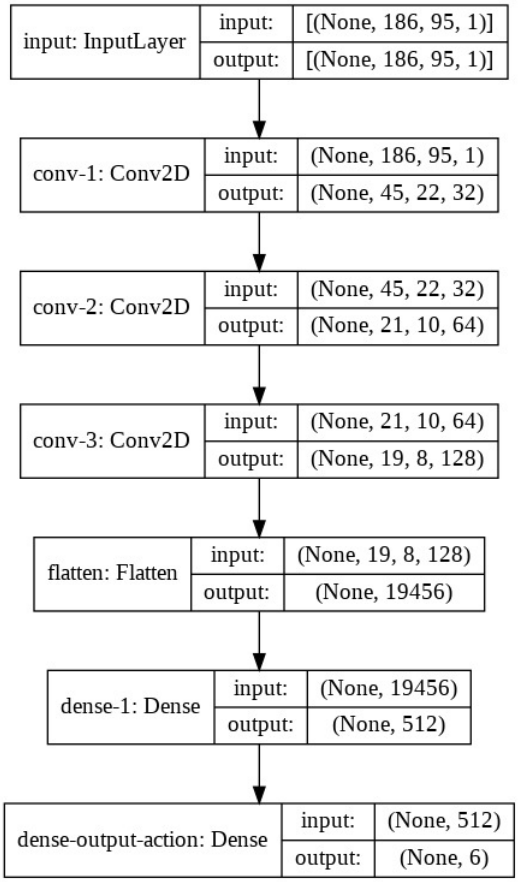

Fig. 11: Policy Network for the Atari Space Invaders Environment

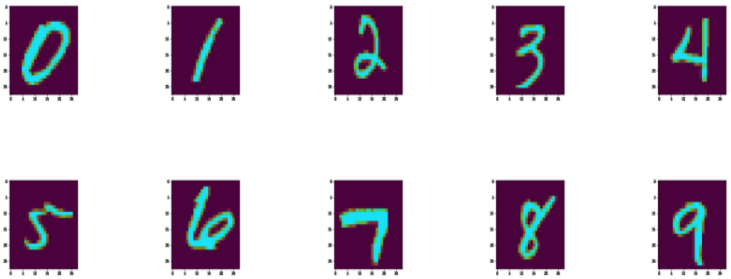

Fig. 12: Sample Prototypes for each of the digit in the dataset

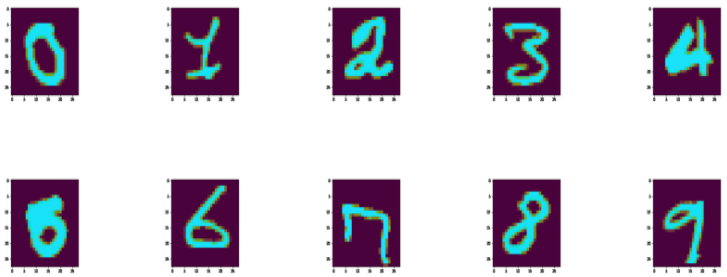

Fig. 13: Sample Criticisms for each of the digit in the dataset

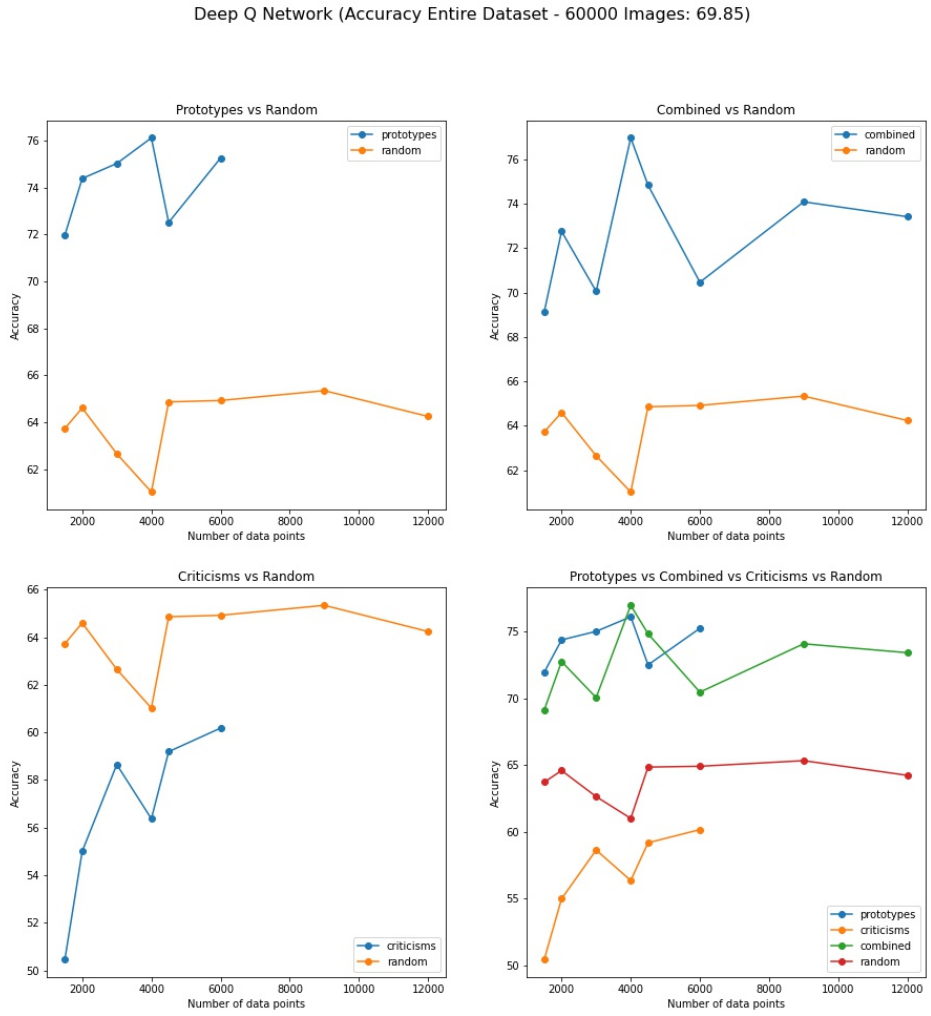

Fig. 14: Test Accuracies for Prototypes vs Random (top-left), Criticisms vs Random (bottom-left), Both Prototypes and Criticisms Combined vs Random (top-right) and Prototypes vs Criticisms vs Prototypes and Criticisms Combined vs Random (bottom-right)