Effective Deep Reinforcement Learning Setups for Multiple Goals on Visual Navigation

Luiz Ricardo Takeshi Horita[1,2], Denis Fernando Wolf[2], Valdir Grassi Junior[2]
Sidia Institute of Science and Technology[1]
University of São Paulo[2]
São Carlos, SP, Brazil
luiz.horita@sidia.com, {denis, vgrassi}@usp.br

***Abstract*—Deep Reinforcement Learning (DRL) represents an interesting class of algorithms, since its objective is to learn a behavioral policy through interaction with the environment, leveraging the function approximation properties of neural networks. Nonetheless, for episodic problems, it is usually modeled to deal with a unique goal. In this sense, some works showed that it is possible to learn multiple goals when using a Universal Value Function Approximator (UVFA), i.e. a method to learn a universal policy by taking information about the current state of the agent and the goal. Their results are promising but show that there is still space for new contributions regarding the integration of the goal information into the model. For this reason, we propose using the Hadamard product or the Gated-Attention module in the UVFA architecture for visual-based problems. Also, we propose a hybrid exploration strategy based on the ϵ-greedy and the categorical probability distribution, namely ϵ-categorical. By systematically comparing different architectures of UVFA for different exploration strategies, and applying or not the Trust Region Policy Optimization (TRPO), we demonstrate through experiments that, for visual topologic navigation, combining visual information of the current and goal states through Hadamard product or Gated-Attention module allows the network learning near-optimal navigation policies. Also, we empirically show that the ϵ-categorical policy helps to avoid local minimums during the training, which facilitates the convergence to better results.**

***Index Terms*—reinforcement learning, goal-driven navigation, visual navigation**

I. Introduction

Artificial intelligence (AI) is a subject that studies how to model intelligence as machine algorithms. Nonetheless, according to Naselaris et al. [1], besides all progress on AI achieved by intensive research on machine learning, it is still limited to the critical ingredients for intelligence (i.e. inference, prediction, reasoning), and there is a gap to be filled regarding the cognitive intelligence. Among all types of machine learning, Reinforcement Learning (RL) is the one that has the objective of learning a behavior, which is usually driven by a policy. RL had emerged among AI and neural research community by combining adaptive dynamic programming with learning-based methods in the mid-1980s, and had attracted plenty attention in early 1990s, after Tesauro [2] reached the level of an expert on playing backgammon using Temporal Difference (TD) learning on a neural network [3, 4].

Sidia - Institute of Science and Technology.

One way to develop and evaluate artificial cognitive intelligence is to adopt an instance of a certain class of problems that are solvable for humans, but still unsolvable for machines [5]. Among all problems where AI can be applied, autonomous navigation in robotics represents a promising research area due to its extensibility to several other applications [6]. Recent advances on deep learning allowed the arising of methods based in Deep Reinforcement Learning (DRL) [7, 8, 9], in which the image of the current robot state is taken as input, and the action is given as output in an end-to-end model.

Although DRL has been presenting promising results, there is still space for improvements. For example, most of existing methods tackle the navigation to a unique pre-defined goal in the environment, i.e., the agent must be able to reach this goal independently to its initial state. However, autonomous navigation should be a multi-goal learning problem, in which the agent must know how to reach any state of the environment from any other initial state. In spite of the fact this discussion is being guided by the robotics view point, it could also be applied for other multi-goal problems that can be modeled as Markov Decision Process (MDP).

Some recent works [9, 10] showed that when applying the idea of a Universal Value Function Approximator (UVFA) on a DRL for a topological environment, the multi-goal learning may be possible. Schaul et al. [11] formally introduced UVFA as an extension of the value function in RL that takes not only the current state of the agent, but also the goal state, to estimate the behavior policy that generalises over multiple goals. Also, the same authors tested different architectures types, and concluded that processing the state and goal data in two-stream network is the most promising approach for multi-goal learning. However, this problem has much to be explored yet. The works that employed UVFA tried few techniques to combine the two input data, e.g. most researchers concatenated the state and goal embedding vectors [8, 9], some authors merged the raw data as the network input [12], and fewer combined the embedding vectors with a dot product [11].

Thus, we present two main contributions in this paper. First, we propose two UVFA architectures that outperform the commonly adopted concatenation of vectors. One of the architectures apply the Hadamard product between the state and goal embedding vectors, and the other uses Gated-Attention module for combining information of the current

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

and the goal states to estimate the Universal Value Function (UVF). The second contribution regards the exploration strategy that allows reaching better policy in the end of training. The proposed strategy combines the ϵ-greedy and the categorical probability distribution, which we name ϵ-categorical. To validate our contributions, we systematically compare different UVFA architectures for different exploration strategies [13]. Experimental results shows that, for visual topologic navigation, combining the information of the current and goal states through Hadamard product or Gated-Attention module allows better convergence for multi-goal learning. Also, we empirically show that the ϵ-categorical policy helps avoiding local minimum during the training.

II. Related Works

In RL, multi-goal learning can be interpreted by many ways, e.g. reaching a moving target that constantly changes its position [14], reaching any of two or more targets present in the environment [15], and reaching a target that can be at any position in the environment [8, 10]. For convenience, the term *multi-goal learning* will be referring to the later case, since it is the focus of this paper.

Naturally, learning multi-goal can be very challenging if compared to learning a unique goal. To address this problem, there were some recent attempts in leveraging any information about the target at the input data. For example, Zhu et al. [8] and Bruce et al. [9] concatenated embedding representations of the observed and the target view images as the robot state. This idea was formally introduced by Schaul et al. [11] as UVFA, a generalized idea of the value function that also takes the goal as argument. According to them, there are three main approaches to combine the agent's state and the goal. The first one, the simplest and the naivest one, is the *concatenated* architecture, which concatenates the data of both the state and goal in the input, which are processed by a unique network stream. The second option is the *two-stream* architecture, which assumes a factorized structure of the problem, processing the state and the goal by an embedding neural network separately before combining them with a function. Lastly, the *decoupled two-stream* has basically the same idea of the *two-stream* architecture, except the fact the networks are previously trained in a supervised way to estimate the UVF.

Experimentally, Schaul et al. [11] tested these approaches in applications where the map coordinates were taken as state vectors, demonstrating that the *two-stream* architecture learns UVFs much faster than the *concatenated*. Moreover, still stated by the authors, when using the same embedding network for state and goal, and apply a distance-based function (e.g., dot product or euclidean distance) for combining both embedding vectors, it allows the UVF estimator to learn that nearby states imply small distances in embedding space. Nonetheless, this assertion was empirically proved in the case where the state space is defined by euclidean space.

Besides Schaul et al. [11], there were some other works that also used the *two-stream* architecture for other games or other applications. For example, Andrychowicz et al. [16] applied this approach to control a manipulator arm to move an object from one place to the other, taking the joint angles as the state vector. Another example is the work published by Veeriah et al. [17], who also applied on games, but taking a sequence of 4 frames as the state representation. Adopting the same idea but for different type of data between the state and the goal, Chaplot et al. [18] proposed the multimodal fusion unit, which employs the Gated-Attention [19] scheme between textual instruction embedding and image representation. According to the intuition given by Dhingra et al. [19], the Gated-Attention module allows the textual instruction embedding act as a filter of information, which decides what parts from the image representation are important to the decision-making.

The UVFA approach may be the key point for multi-goal learning, however exploration policy might also impact the learning efficiency. Usually, ϵ-greedy strategy is used, where the ϵ factor adapts over the training time steps [8, 16, 18, 20]. The advantage of this technique is that it is possible to control the probability of exploration against the probability of exploitation. Another strategy that fewer authors employ is the *softmax* policy (or categorical policy), where the probability of an action being chosen is proportional to its respective action-value related to the others [11, 17]. Compared to ϵ-greedy, the categorical policy gradually adapts the probability of actions being explored according to the estimated action-values. The intuition behind this idea is that, in the beginning of the training, all the estimated action-values are nearly the same, so every action has approximately the same probability of being chosen. As the agent optimizes its policy, the probability distribution among the possible actions change so that the best action can be chosen more often than the others.

Papers on several applications are attempting to resolve multi-goal learning through the UVFA concept. In this paper, we focus in the scenario where the state-space and the goal information are given as images for visual navigation, and propose using the Hadamard product or the Gated-Attention module for UVFA. Also, we propose a hybrid exploration strategy that combines the ϵ-greedy and the categorical policies, so that we can keep a small probability of random exploration during the whole training process. By doing so, we might avoid the network to converge to a local minima.

III. Background

A Markov Decision Process model of the environment consists of a state space $S = \{s_1, s_2, ..., s_N\}$, an action space $A = \{a_1, a_2, ..., a_M\}$, and the reward function $R : S \times A \rightarrow \Re$ that rewards or penalizes the agent according to the action it takes in its current state.

Given the definition of an MDP environment, the problem that RL tries to solve is to find out the best action at a given state s_t that maximizes the expected return in future. In other words, to maximize the accumulated discounted rewards of future states. Formally, the objective function that determines

this expected return is the value function $V(s)$:

$$V(s_t) = \max_{\pi} \mathbb{E}\left[\sum_{\tau=t}^{\infty} \gamma^{\tau} R(s_\tau)\right] \tag{1}$$

where π is the complete decision policy that guides the whole sequence of actions, $\mathbb{E}$ is the expected accumulated reward, and $0 \leq \gamma \leq 1$ is the discount factor that reduces the importance of future values. Considering the recurrence relation we have the Bellman Equation

$$V(s) = \max_{a}\left[R(s,a) + \gamma \sum_{s'} \cdot V(s')\right] \tag{2}$$

which can also be rewritten as the action-value function (also known as the Q-value function)

$$\begin{aligned} Q(s,a) &= R(s,a) + \gamma \sum_{s'} V(s') \\ &= R(s,a) + \gamma \sum_{s'} \max_{a'} Q(s',a') \end{aligned} \tag{3}$$

where $V(s) = \max_a Q(s,a)$. This function represents the performance metric (or quality value) of taking the action a in the state s.

Now, considering that the policy is responsible for choosing the action, we can specify the optimal policy as

$$\pi^*(s) = arg\max_{a}\left(R(s,a) + \gamma \sum_{s' \in S} V^*(s')\right) \tag{4}$$

which can also be written as $\pi^*(s) = arg\max_a (Q(s,a))$.

A. Universal Value Function Approximation

The value function represents the core of RL, since it addresses the temporal credit assignment, i.e., the problem of assigning long-term rewards to an action, which is essential in the moment of choosing an action. However, the way how classical methods models the value function assumes there is only one goal to be achieved. Aiming to generalize the value function for cases that multiple goals can be present, Schaul et al. [11] proposed the idea of a *Universal Value Function* (UVF), which can be thought of as a large set of value functions optimal for different goals unified into a unique function that generalises over states and goals.

For better understanding, assume an MDP and a set of goals $G = g_1, g_2, ..., g_L$. For each goal $g \in G$, there is a defined reward function $R_g(s,a)$ and a discount function γ_g. Thus, for each goal, there is an optimal policy $\pi^*_g : S \mapsto A$ that guides the agent to take the actions that maximizes the value function

$$V_{g,\pi_g}(s) = \mathbb{E}\left[R_g(s, a_{\pi_g}) + \gamma_g \sum_{s'} V_{g,\pi_g}(s')\right] \tag{5}$$

or the action-value function

$$Q_{g,\pi_g}(s,a) = R(s,a) + \gamma_g \sum_{s'} \mathbb{E}\left[Q_{g,\pi_g}(s', a'_{\pi_g})\right] \tag{6}$$

where the actions are chosen according to π_g.

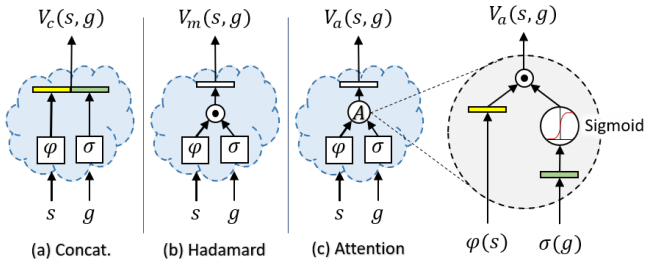


Fig. 1. Diagram of UVFA architectures. Combine embedding vectors by **(a)** concatenation, **(b)** Hadamard product, and **(c)** Gated-Attention module.

Considering the idea of using a parameterized function to approximate the value function, the goal descriptor can be included as an argument of this function. Thus, the functions $V(s,g;\mathbf{w}) \approx V^*_g(s)$ or $Q(s,a,g;\mathbf{w}) \approx Q^*_g(s,a)$ for all $g \in G$ would approximate the optimal value function over potentially large state and goal spaces.

IV. UVFA Architectures

For UVFA, one could adopt any parameterized function approximator that receives the state s and goal g data as input, and outputs the estimated $V(s,g)$. With respect to this, the advantage of working with neural networks is that it allows breaking the architecture in network blocks and combine them in different ways. A naive solution would be to concatenate the state and goal data in a unique input for the network, but it has already been proved it is not the best solution [11]. Instead, processing each data in parallel network streams ($\varphi(s)$ and $\sigma(g)$) and combining the resulting embedding vectors into a single stream by a function $H(\varphi(s), \sigma(g))$ seems to be the most promising approach.

Figure 1(a) shows the simplest and most commonly adopted approach, which combines the embedding vectors by concatenation. The following architecture in Figure 1(b), which is proposed in this paper, merges the processed data from the state and goal by Hadamard product, i.e. by element-wise product. Finally, the last combination approach, in Figure 1(c), that is also a contribution of this paper, uses the Gated-Attention [19] module.

Here we decided to use the Hadamard product instead of the dot product, which has already shown to be effective by School et al. [11]. The dot product has proved to be effective for the case where the state-space was defined by euclidean coordinates of a map. For this paper, where the state-space is defined by images, using dot product does not seem to make much sense as this operation provides the notion of distance. For this reason, we propose using the Hadamard product. Moreover, the result of a dot product between two vectors is a single value, which might represent implicitly how close we are to the goal, but will not tell anything about what is the best action to take in the current state.

Regarding the Gated-Attention, it basically receives an input and a key signals. The input signal is processed to the

output, and the key signal is used to tune the processing. Mathematically, the output of this module can be written as

$$\mathbf{v}_{output} = \mathbf{v}_{input} \odot Sigmoid(\mathbf{v}_{key}) \tag{7}$$

where $\odot$ is the Hadamard product.

The intuition behind this type of module is that the key signal serves as a filter for the input signal. In the context of UVFA, the input signal would be the state embedding of the agent, while the key signal would be the goal embedding. By doing so, the goal embedding would be the key to model the value function of the given state accordingly, making the agent to learn a goal-driven policy.

V. ϵ-Categorical Exploration Approach

Regarding the exploration policy, the most common strategy is the ϵ-greedy, which makes the agent to explore with probability $\epsilon(t)$ and exploit the learned policy with probability $(1-\epsilon(t))$, where $\epsilon(t)$ is a probability that decreasingly changes over the time steps. The difficulty of using ϵ-greedy is to set the best $\epsilon(t)$ so we can get an efficient policy convergence to near-optimality. On the other hand, categorical policy leverages the proportion of estimated action-values $Q(s,a)_{\forall a \in \mathbb{A}}$ to compute the probability of choosing each action a in a given state s. Different from ϵ-greedy, categorical policy naturally changes the probability as the agent gains confidence about the policy. Nonetheless, depending on the complexity of the problem, this strategy may lead the convergence to a local minimum.

Algorithm 1 ϵ-Categorical Exploration Policy

Input: s
Output: a

```
Get random value x between [0, 1]
if x < ε then
    Get random action a from action space.
else
    Draw action a according to probability defined by the
    action-values Q(s, a).
end if
return a
```

Alternatively, we propose to combine both strategies in order to mitigate the drawback of both. Algorithm 1 presents how this exploration policy works. In practice, the probabilities of an action being chosen during the training process is given by

$$p(a|s) = \begin{cases} 1/N_a & \text{with prob. } \epsilon \\ \frac{Q(s,a)}{\sum_{a' \in \mathbb{A}} Q(s,a')} & \text{with prob. } (1-\epsilon) \end{cases} \tag{8}$$

where N_a is the number of actions in the action-space.

VI. Experiments

A. Framework

For experimental evaluation, a topological environment was created over an urban simulated area. To make it closer to reality, each node of this environment consists of a situation where the driver must take a decision, i.e., when an intersection is reached. Therefore, instead of a node representing the whole intersection, each intersection may have more than one node. In this way, decision making becomes agent-centered instead of being related to the coordinates of the environment. Moreover, each node of the topological environment is associated with a wide snapshot (field of view 120º) of the front view of the vehicle.

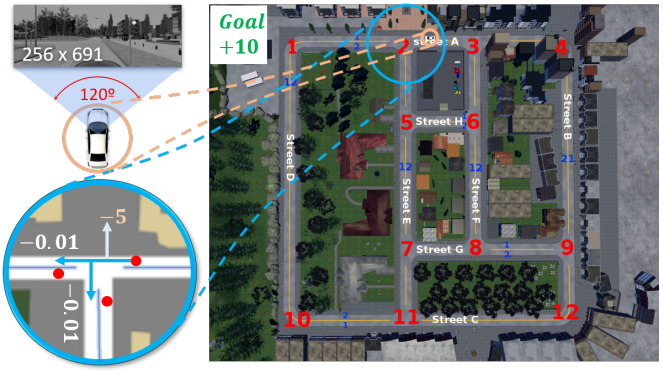


Fig. 2. The emulated topologic environment built over the Town 02 from CARLA. In the map, numbers in red indicate the street intersections, while blue numbers indicate lanes of each street. The nodes were created for each position where a street lane met an intersection (illustrated in the zoomed blue circle), and each node was associated to an image correspondent to the front view of the vehicle (illustrated in the upper left).

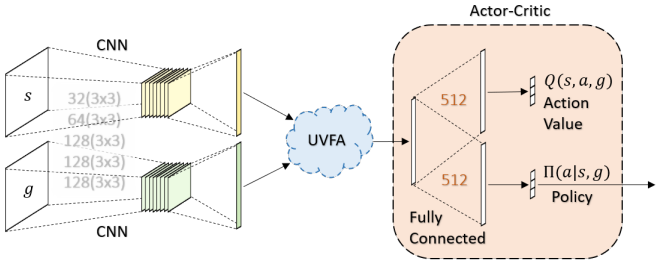


Fig. 3. Diagram of the actor-critic agent architecture.

The *CARLA Simulator* was used to build this environment [21]. More specifically, the *Town 02* map was used to obtain data for simulation. Figure 2 illustrates how the environment was modeled. In total, the topological environment comprises with 32 nodes and 96 edges, in which each node is associated to an image, taken at its respective position in the map with a front-camera of the vehicle.

For the MDP formulation, the state-space is given by gray-scaled images of 256×691 resolution, and its action-space is defined by three possible choices the agent can do in a node (i.e. turn left, go straight, or turn right). Moreover, for multi-goals learning, we have the goal-space also defined by gray-scaled images. With respect to the reward policy, the agent gets $r_{step} = -0.01$ for each action taken, $r_{break} = -5.0$ if the action makes the vehicle leave the navigable area, and $r_{goal} = +10.0$ if it reaches the goal. See illustration of rewards in the zoomed blue circle of Figure 2.

Finally, to solve the MDP problem, we use the actor-critic method with experience replay strategy for better sample

efficiency [22]. Figure 3 illustrates the agent's architecture, in which the state and goal images are processed by two identical convolutional networks in parallel, and then combined with an UVFA structure. After merging the state and goal embedding data, the resulting vector is given to the actor-critic head, which estimates the action-values and the actions probabilities (the policy) with two fully connected layers. Furthermore, the Trust Region Policy Optimization (TRPO) heuristic proposed by Schulman et al. [13] was adopted to allow more effective optimization of the neural network. Basically, TRPO limits the learning step size according to the average Kullback-Leibler (KL) divergence between new policies and old policies.

B. Results

Aiming the evaluation of all the three UVFA architectures mentioned in Section IV, and the exploration policies (i.e. ϵ-greedy, categorical, and ϵ-categorical), all combinations were executed. Each experiment was limited to 5000 training episodes, in which the initial and goal states are randomly chosen at every new episode, asserting these states are not the same. Also, the experience replay ratio was set to 4, meaning that four off-policy iterations are executed at each training iteration [22]. Although performances can vary substantially from run to run, each UVFA architecture and exploration policy combination was run once due to time constraints.

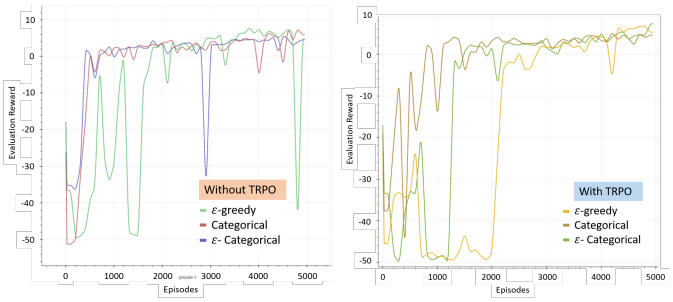
Table I shows the success rates of all the experiences. Success rate was computed from 100 random test episodes, in which an episode is considered successful if it reaches the respective goal in less than 10 action steps. Considering only the final experimental results, it is clear that the best performances concerning this metric were obtained by using the Gated-Attention module for both categorical and ϵ-categorical policies. But, if analyzing the convergence process during the whole training execution, we can have a better understanding of the impact of each parameter or configuration over the performance.

Figures 4, 5, and 6 show the learning curves concerning the averaged evaluation accumulated rewards for every UVFA architectures, exploration policies and TRPO. The evaluation consisted of 50 testing episodes after each 100 training episodes.

Analyzing the learning curves of the UVFA that combines embedding vectors by concatenation (Figure 4), we can see that the network optimizes almost monotonically when using TRPO, reducing considerably the learning variance. Also, in Figure 4(b), it is possible to see that the categorical policy allows faster convergence, due to its better exploration

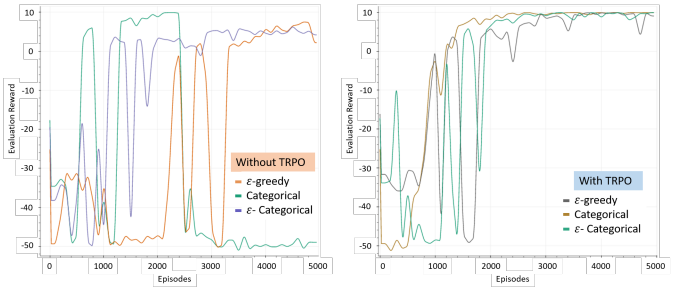
TABLE I
SUCCESS RATES OBTAINED AFTER 5000 TRAINING EPISODES.

	E-greedy		Categorical		E-categorical	
	with TRPO	without TRPO	with TRPO	without TRPO	with TRPO	without TRPO
Concat.	38%	62%	47%	66%	65%	43%
Hadamard	**80%**	63%	85%	4%	94%	68%
Attention	64%	1%	5%	**98%**	**99%**	14%



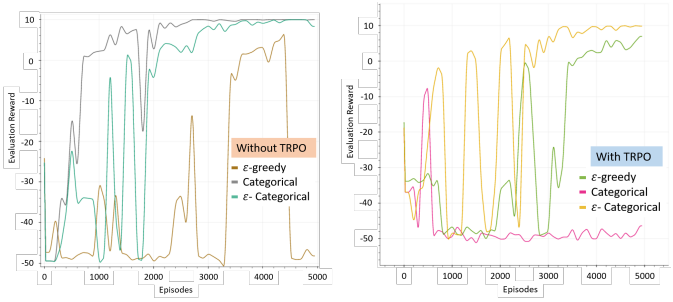
(**a**) (**b**)

Fig. 4. Comparative learning curves for different exploration policies and TRPO scenarios for concatenated UVFA.



(**a**) (**b**)

Fig. 5. Comparative learning curves for different exploration policies and TRPO scenarios for Hadamard UVFA.

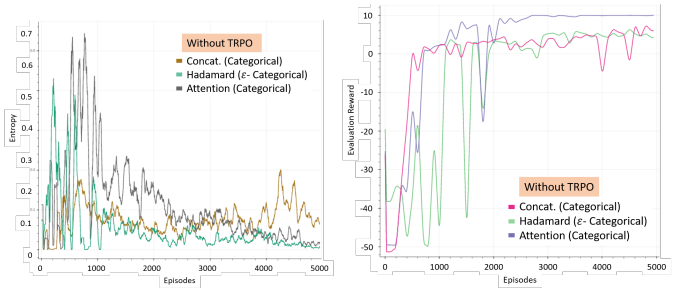


(**a**) (**b**)

Fig. 6. Comparative learning curves for different exploration policies and TRPO scenarios for Gated-Attention UVFA.

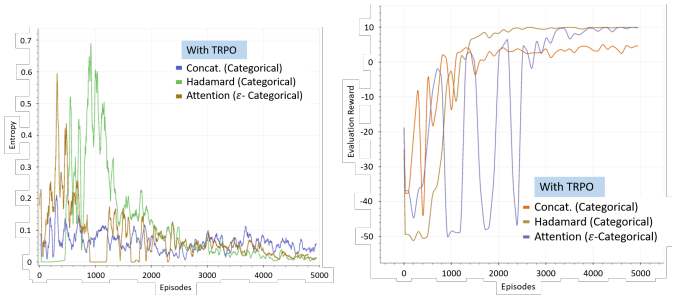
balancing. On the other hand, since ϵ-categorical insert some level of randomness for exploration, the convergence becomes slower. Lastly, because ϵ-greedy has a hard-coded exploration-exploitation ratio, it is difficult to get faster optimization.

Compared to the concatenation approach, Hadamard product provides higher learning variance, which hinders the optimization, as shown in Figure 5(a). A possible reason for this is that, every time the network is updated, even small changes in the parameters of the both input convolutional networks may cause greater impact on the UVFA output, since there is a multiplication between each elements of the embedding vectors. However, TRPO showed to be enough to control this learning variance, as shown in Figure 5(b). Regarding the impact of exploration policies over the architecture with Hadamard product, we can see that when using TRPO all



(**a**) (**b**)

Fig. 7. Comparative performance of different UVFA architectures in the case of **not** using TRPO.



(**a**) (**b**)

Fig. 8. Comparative performance of different UVFA architectures in the case of using TRPO.

of them could reach a similar result in the end. Nonetheless, we can still see the categorical policy offers more monotonic convergence. Howsoever, ϵ-categorical still allows the agent to learn near-optimal universal policy for visual navigation, i.e. reach a random goal in the environment with accumulated reward close to the goal reward r_{goal}.

In turn, the third UVFA architecture, which uses the Gated-Attention module to filter the state embedding with the goal embedding, showed a particular convergence behavior that was not observed in the other architectures. In this case, TRPO slowed down the convergence, and was not enough to control the learning variance. Moreover, as shown by the pink curve in Figure 6(b), the categorical exploration policy got stuck in a local minimum. In navigation control, due to sparse rewards, the agent could not learn how to reach the goal, but only to maintain the vehicle in the navigable area. On the other hand, even though it presented large learning variance until around 3000 training episodes, ϵ-categorical could converge to near-optimal solution, as shown by the yellow curve in Figure 6(b).

We have compared the impact of exploration policies and TRPO over each UVFA architecture. Now, Figures 7 and 8 show the best learning curves of each UVFA architecture, concerning the estimated policy entropy and the averaged evaluation reward. One fact that is easily observable on the item (b) of both figures is that, the UVFA architecture that uses concatenation could not reach near-optimal policy. This explains why this architecture shows low performances in Table I. The same convergence behavior can be seen for the architecture with Hadamard product when not applying TRPO (Figure 7(b)). On the other hand, with categorical exploration policy and TRPO, Hadamard product allows fast convergence to near-optimality, as shown in Figure 8(b).

The only combination approach that allowed UVFA to converge to near-optimal navigation policy in both cases (using TRPO or not) was the Gated-Attention, despite of its high learning variance when using TRPO and ϵ-categorical strategy. The last observation is about the policy entropy curves in item (a) of Figures 7 and 8. In general, it is possible to see that the entropy curves decrease at a faster rate when using TRPO, including the case of using the Gated-Attention module. Although the evaluation reward curve (the purple curve in Figure 8(b)) presents much less variance when not using TRPO, the entropy curve shows the opposite. Therefore, even though the evaluation reward varies a lot during the training, by looking at the entropy curve we can empirically suggest that the network can learn the near-optimal universal policy.

VII. Discussion

In this paper, we propose two architectures for UVFA that improves the multi-goal learning when the state-space and goal-space are defined as images. Experimentally, we show that our architectures present better convergence properties and reach better universal policy if compared with the architecture based on the concatenation of state and goal embedding vectors.

Also, as another contribution, we propose an exploration strategy called ϵ-categorical, which is the combination of the ϵ-greedy with the categorical (or softmax) policy. In fact, categorical policy can provide more efficient exploration strategy, allowing faster learning. However, in some cases it might get stuck in a local minimum. In contrast, ϵ-categorical provides a way to avoid this kind of situation by inserting a certain probability ϵ of randomness in the categorical policy. As one can expect, and what is shown empirically in this paper, this small randomness increases the learning variance, but still helps the agent to learn near-optimal solution for multi-goal learning.

Regarding the TRPO, in most cases it indeed improves the convergence, as proposed by Schulman et al. [13]. This is because, it provides theoretical foundations that guarantee monotonic improvement of the policy during training. However, since TRPO provides a constraint to the learning step size according to the KL divergence, at some level, it can slow down the network convergence.

Schaul et al. [11] proposed the UVFA concept and validated it in applications where the state-space was defined by euclidean coordinates. After this, some other authors tried to apply the same idea in other applications in which the state-space is given by images, but keeping the simple concatenation of vectors to combine the state and goal information. In this paper, we propose different techniques to rethink the UVFA architecture to improve performance on this kind of visual-based application. Here, we applied the proposed architectures

for visual-navigation of a vehicle, but the same line of thought could be applied for several other MDP problems. We hope that the approach proposed serves as an inspiration for further studies to define the best way of modeling the UVFA architecture, given the application's domain.

Concerning the exploration strategy, as far as we know, this is the first time the concept of here presented as ϵ-categorical is proposed and tested for multi-goal learning. Since this technique was tested only in a controlled environment, we expect it will serve as a jumping-off point for future works on training agents in more challenging environments. Also, we hope more sophisticated exploration policies can be derived from ϵ-categorical, such that the ϵ can be changed during the training according to reasonable criteria.

ACKNOWLEDGMENT

This research was financed in part by the Brazilian National Council for Scientific and Technological Development - CNPq (grant 465755/2014-3), by the Coordination of Improvement of Higher Education Personnel - Brazil - CAPES (Finance Code 001 and 88887.136349/2017-00), and the São Paulo Research Foundation - FAPESP (grant 2014/50851-0). Not less important, Sidia, an Institute of Science and Technology located in Brazil that contributes to mobile-related projects of globally leading companies of the industry, also supported in part this work.

REFERENCES

[1] T. Naselaris, D. S. Bassett, A. K. Fletcher, K. Kording, N. Kriegeskorte, H. Nienborg, R. A. Poldrack, D. Shohamy, and K. Kay, "Cognitive Computational Neuroscience: A New Conference for an Emerging Discipline," *Trends in Cognitive Sciences*, vol. 22, no. 5, pp. 365–367, 5 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1364661318300433

[2] G. Tesauro, "TD-Gammon, a Self-Teaching Backgammon Program, Achieves Master-Level Play," *Neural Computation*, vol. 6, no. 2, pp. 215–219, 3 1994. [Online]. Available: http://www.mitpressjournals.org/doi/10.1162/neco.1994.6.2.215

[3] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement Learning: A Survey," *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, 5 1996. [Online]. Available: https://jair.org/index.php/jair/article/view/10166

[4] A. Gosavi, "Reinforcement Learning: A Tutorial Survey and Recent Advances," *INFORMS Journal on Computing*, vol. 21, no. 2, pp. 178–192, 5 2009. [Online]. Available: http://pubsonline.informs.org/doi/10.1287/ijoc.1080.0305

[5] A. Lieto, M. Bhatt, A. Oltramari, and D. Vernon, "The role of cognitive architectures in general artificial intelligence," *Cognitive Systems Research*, vol. 48, pp. 1–3, 5 2018.

[6] T. T. Mac, C. Copot, D. T. Tran, and R. De Keyser, "Heuristic approaches in robot path planning: A survey," *Robotics and Autonomous Systems*, vol. 86, pp. 13–28, 12 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921889015300671

[7] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real Deep Reinforcement Learning: Continuous Control of Mobile Robots for Mapless Navigation," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 31–36.

[8] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 3357–3364.

[9] J. Bruce, N. Sünderhauf, P. Mirowski, R. Hadsell, and M. Milford, "Learning Deployable Navigation Policies at Kilometer Scale from a Single Traversal," 7 2018. [Online]. Available: http://arxiv.org/abs/1807.05211

[10] P. Mirowski, M. K. Grimes, M. Malinowski, K. M. Hermann, K. Anderson, D. Teplyashin, K. Simonyan, K. Kavukcuoglu, A. Zisserman, and R. Hadsell, "Learning to Navigate in Cities Without a Map," in *Proceedings of the 32nd International Conference on Neural Information Processing Systems*. Montreal, Canada: Neural Information Processing Systems Foundation, Inc., 2018, pp. 2424–2435.

[11] T. Schaul, D. Horgan, K. Gregor, and D. Silver, "Universal Value Function Approximators," in *32nd International Conference on Machine Learning*, vol. 37, 6 2015, pp. 1312–1320. [Online]. Available: http://proceedings.mlr.press/v37/schaul15.html

[12] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. P. Abbeel, and W. Zaremba, "Hindsight Experience Replay," pp. 5048–5058, 2017. [Online]. Available: http://papers.nips.cc/paper/7090-hindsight-experience-replay

[13] J. Schulman, S. Levine, P. Moritz, M. Jordan, and P. Abbeel, "Trust Region Policy Optimization," in *International Conference on Machine Learning*. Proceedings of Machine Learning Research, 2015. [Online]. Available: http://proceedings.mlr.press/v37/schulman15.pdf

[14] L. Pack Kaelbling, "Learning to Achieve Goals," in *International Joint Conference on Artificial Intelligence*, 1993, pp. 1094–1099.

[15] A. Arleo, F. Smeraldi, and W. Gerstner, "Cognitive navigation based on nonuniform Gabor space sampling, unsupervised growing networks, and reinforcement learning," *IEEE Transactions on Neural Networks*, vol. 15, no. 3, pp. 639–652, 2004.

[16] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. P. Abbeel, and W. Zaremba, "Hindsight Experience Replay," pp. 5048–5058, 2017. [Online]. Available: http://papers.nips.cc/paper/7090-hindsight-experience-replay

[17] V. Veeriah, J. Oh, and S. Singh, "Many-Goals Reinforcement Learning," 6 2018. [Online]. Available: http://arxiv.org/abs/1806.09605

[18] D. S. Chaplot, K. M. Sathyendra, R. K. Pasumarthi, D. Rajagopal, and R. Salakhutdinov, "Gated-Attention Architectures for Task-Oriented Language Grounding," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 4 2017. [Online]. Available: http://arxiv.org/abs/1706.07230

[19] B. Dhingra, H. Liu, Z. Yang, W. W. Cohen, and R. Salakhutdinov, "Gated-Attention Readers for Text Comprehension," 6 2016. [Online]. Available: http://arxiv.org/abs/1606.01549

[20] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous Methods for Deep Reinforcement Learning," 2 2016. [Online]. Available: https://arxiv.org/abs/1602.01783

[21] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An Open Urban Driving Simulator," 11 2017. [Online]. Available: http://arxiv.org/abs/1711.03938

[22] Z. Wang, V. Bapst, N. Heess, V. Mnih, R. Munos, K. Kavukcuoglu, and N. de Freitas, "Sample Efficient Actor-Critic with Experience Replay," 11 2016. [Online]. Available: http://arxiv.org/abs/1611.01224