IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Revue d'Intelligence Artificielle
Vol. 33, No. 4, August, 2019, pp. 265-273
Journal homepage: http://iieta.org/journals/ria

Optimal Combination of Imitation and Reinforcement Learning for Self-driving Cars

Fenjiro Youssef*, Benbrahim Houda

National School of Computer Science and Systems Analysis (ENSIAS), Mohammed V University, Rabat 8007, Morocco

Corresponding Author Email: fenjiro@ensias.ma

https://doi.org/10.18280/ria.330402

Received: 12 May 2019
Accepted: 23 July 2019

Keywords:
deep reinforcement learning, behavioral cloning, supervised imitation learning, prioritized experience replay, expert's trust margin, simulation environment

ABSTRACT

The two steps in human intelligence development, namely, mimicking and tentative application of expertise, are reflected by imitation learning (IL) and reinforcement learning (RL) in artificial intelligence (AI). However, the RL process does not always improve the skills learned from expert demonstrations and enhance the algorithm performance. To solve the problem, this paper puts forward a novel algorithm called optimal combination of imitation and reinforcement learning (OCIRL). First, the concept of deep q-learning from demonstrations (DQfD) was introduced to the actor-critic (A2C) model, creating the A2CfD model. Then, a threshold was estimated from a trained IL model with the same inputs and reward function with the DOfD, and applied to the A2CfD model. The threshold represents the minimum reward that conserves the learned expertise. The resulting A2CfDoC model was trained and tested on self-driving cars in both discrete and continuous environments. The results show that the model outperformed several existing algorithms in terms of speed and accuracy.

1. INTRODUCTION

For the apprenticeship of any skills, a human being always began by learning from teachers, copying their behaviors and the way they are making things working, and once the expert level is reached, the knowledge and performance are then limited by the teacher mastery's level represented by his instructions and demonstrations. To go beyond, humans adopt explorations strategy [1-3] to discover new areas, methods, and skills to improve their expertise and outperform current expert capacities. If it is not well managed and controlled, this exploration process can sometimes diverge and yield to forget expert demonstrations, leading to a suboptimal performance that is lower than the capacities learned from the experts.

In artificial intelligence, we can find the same learning process, beginning by imitation learning (IL), where we leverage supervised learning regression algorithms capabilities to teach a given policy using expert labeled demonstrations as a training dataset [4], so the model tries to faithfully mimic the behavior of this expert, but the IL agent need large dataset of demonstrations for the training and lacks generalization abilities, since the large search space with many continuous variables represent the main constraint that IL face while solving real-world problems. Moreover, agent performance is capped and limited by the level of the expert skills learned through his demonstrations. Therefore, IL models can only achieve the tasks as good as the expertise level of the teacher, which leads most of the time to only a near-optimal policy.

At this stage, the Reinforcement learning algorithms can take over, since it has the capacity of self-learning and self-improvement, thus, RL models can discover through trial and error new strategies and maneuvers that can enhance the IL policy. RL agent learns optimal policies from rewards, a signal that encourages or discourages to do a given action while interacting with the environment; however, it suffers from many issues like the reward sparsity and slowness of the training process. In the last years, many attempts have been made to combine IL and RL and take advantage of the strengths of both models to improve performance and shorten the learning time by leveraging supervised learning capacities to initialize the RL neural network with expert demonstrations, using two main techniques:

- Pre-train DRL neural network using supervised learning IM: the resulting trained parameters that have proved their good performance in classification, allow initializing the DRL network with a good policy using Transfer learning [5].
- Use the DRL Prioritized experience replay [6] (PER) memory to inject expert demonstrations, and thus providing the model with data that contains good behaviors' transitions, labeled by experts in an early stage of the training phase. The PER filled with expert demonstrations can be used alone for a pre-training or during the Off-policy training with other stored transitions that came from the interaction of the agent with its environment. For this last case, a prioritized experience replay is used and a higher probability is assigned for the expert demonstrations transitions to encourage sampling from them more often.

For both techniques, DRL algorithms can produce a suboptimal policy with inferior performance than the IL policy that it was initialized with if the planning problems are not resolved until optimality is reached, which calls into question the main reason behind the IL and RL's combination. This issue could be due mostly to three points:

- High exploration rate [3, 7] that is adopted in such algorithm to encourage discovering new strategies that can outperform IL initialization.
- Transitions similarity of the expert demonstrations, which

is due to the coherent expert's behavior when dealing with similar situations, so we can have in the demonstrations many of the same transitions that could harm the model performance as it shrinks the agent training to a smaller state-space.

- The difference between the two distributions of the demonstrations' transitions and the RL agent's transitions.

In this paper, we address this issue exposed above by proposing a new algorithm OCIRL "Optimal Combination of Deep Imitation and Reinforcement learning", which allows DRL models that rely on intrinsic reward function (tailored for self-driving car environment), to preserve the learned skills from expert demonstrations, and evolve the DRL policy only when it performs the desired tasks at the same level or better than the IL expert policy, by applying a threshold on cumulative reward to condition the update of the DRL target network each N steps. This threshold is estimated from an already trained IL network with the expert's demonstration that receives the same input image as the DRL network and uses the same DRL intrinsic shaped reward function. We applied the OCIRL concept to A2C [8] algorithm, stating by adapting the DQfD [8, 9] to the actor-critic model that we call A2CfD, and then add the threshold condition that is the object of our paper, the new model is called A2CfDoC which stands for A2C from Demonstrations optimal Combination, and we trained and tested it on Self-driving cars virtual Simulator OpenAI Carraccing-v0 which is a discrete environment and CARLA a continuous one.

We obtain good results for both discrete and continuous environments that outperform IL (behavioral cloning [10]), RL (DQfD [8] and A2C [11]), during the training and testing. We got a better policy that gets better results than IL and RL, even for a complex and continuous environment like CARLA, which difficulty is near ground truth.

2. BACKGROUND

Our approach is built upon Reinforcement learning [1, 12] concept and Markov Decision Process MDP, a sequential decision-making problem based, defined by a 5-tuple: A set of states and actions (S,A), reward model R, state transition probability matrix P (from all states s to all their successor s') and discounted factor $\gamma \in [0,1]$, to give more importance to recent rewards compared with future rewards.

The policy function π maps the possible states S of the agent to its selective action a, $\pi: S \rightarrow p(A = a|S)$. The RL agent tries to learn π^*, the optimal policy, so as to maximize the cumulative reward by taking the best actions.

The Actor-Critic (AC) algorithms [12, 13] are hybrid RL methods that combine both, the policy gradient method and the value function method. They are composed of the Actor, the policy function that generates the actions to execute and the Critic that assesses the actor's actions. The high variance introduced by the Actor is lowered by the Critic which balances the equation and higher the likelihood of convergence of the policy gradient methods.

With Prioritized experience replay [6] (PER), we store agent experience (s_t,a_t,r_t,s_{t+1}) in a memory D queue ranked using the criterion TD-error, by giving the highest probability to the transitions with the highest |TD-errors|, since they have the highest potential of learning progress. During the train, mini-batches of transitions are sampled from the PER (s, a, r, s') as input to the DRL network.

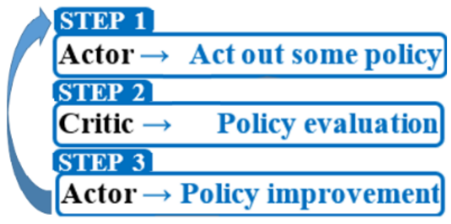
STEP 1
Actor → Act out some policy
STEP 2
Critic → Policy evaluation
STEP 3
Actor → Policy improvement

Figure 1. Actor-critic algorithm steps

3. RELATED WORK

In the last decade, Artificial Intelligence supported by the breakthroughs of deep learning in the areas of computer vision and control has been focused on two main concepts, imitation learning, and reinforcement learning to enhance the capacities and accuracy of the models.

Imitation learning (IL) [4] most adopted form is Behavioral Cloning (BC) [14] that uses supervised learning with the goal to make a neural network (convolutional net for vision plus a Fully connected net for control) behave as close as the observed expert trajectory, by optimizing the error of a loss function. Many BC models have proven their efficiency, especially in navigation and autonomous driving, it was used in many models ALVINN [15], DAVE [16] and DAVE2 [10], but gives poorer predictions on unseen situations. This issue was addressed by Dagger [17] that keeps its training Dataset growing by appending the newly visited states labeled which require the intervention of an expert during all the training, which is very constraining. Other extensions of Dagger like AggreVaTe [18], SEARN [19], SMILE [20] that demonstrated high performance on ground-truth. All these models suffer from a limitation of the performance since the IL agent can be only as good as the expert's demonstrations, also it lacks generalization, when facing unknown environment, however these two points turn out to be the great strengths of the Deep Reinforcement learning models that have made great steps toward end-to-end control algorithms, proving their capacities to manage new situations, by learning through the process of trial and error and exploration to favor certain actions over others. DQN, Double and Dueling DQN were one of the initial and more important steps and comes after a wide list of techniques that enhanced the model like the prioritize experience replay, multi-step learning and actor-critic deep policy (A3C [21], PPO [22], ACKTR [23]). The results of DRL were a great success for artificial intelligence researchers, the generalization capacity was good but it still lacks the speed of learning.

Recently, many papers address this subject and try to combine both algorithms IL and DRL like THOR [24], LOKI [25] and DQfD [8], these algorithms try to leverage expert demonstrations using imitation supervised learning, to speed up the training and allow DRL models to bypass the slowness of the beginning of the learning phase, due to the exploration of the environments through trial and error. So, it initialize the DRL policy network by injecting the expert skills in an early stage of training, which leads to a faster convergence than executing a from scratch policy gradient, and let to the RL training phase, the task of fine-tuning the model, so as to overtake the skills gained through mimicking the demonstrations.

Most of the time, these hybrid approaches are made by adding a supervised imitation learning term to the DRL TD loss function, so the policy will tend to behave the same as in the demonstrations and not forget the IL learning it receives in the pre-training phase [8].

These implementations have made steps in the right direction, but it remains that we must ensure that the resulting combined algorithm will outperform the expertise level learned using IL, and that's exactly the problem we have addressed in this paper.

4. OPTIMAL COMBINATION OF IL AND RL

4.1 Problem definition

The objective of combining IL with RL is to outperform the expert skills learned through the demonstrations without any drop in the performance of the new model regarding the expert demonstrations and ensuring a continuous accumulation of knowledge and skills. Also, the new model must deal with real-world environment, consequently it has to manage continuous action spaces.

4.2 OCIRL concept and architecture

The solution that we propose in this paper combines two main principles to stabilize the improvement of the learning and prevent it from divergence:

Expert's trust margin: The experience replay (ER) memory of the DRL model is filled by expert demonstrations. After training the DRL model off-line based on the PER expert demonstrations data set, we get an expert network, with which we instantiate two DRL agents (same network and same weights) and both interact with two environment instances that have the same seed. We call the two networks respectively, the Explorer network and the expert critic network.

The Explorer network is initialized by the expert network but adopts an exploration strategy to discover better actions and behaviors, and the expert critic network is used as the expert reference network that conditions the evolution of the learning improvement through his assessment of the Explorer network behavior, with the goal to constrain the model to stay in high-reward regions. Both networks interact with their respective environments, generate their actions and calculate their cumulative rewards using a shaped intrinsic reward function, which values are respectively R_{Expert} and $R_{Explorer}$. The learning improvement is checked each N steps to trigger the update of the Expert Critic Network by the Explorer network's weights in an asynchronous mode. N represents the number of steps we wait before updating the Expert critic network in an asynchronous mode with the weights of the primary network, in order to keep the stability of the neural network during the training.

The computed R_{Expert} is used as a critic signal that will condition the validation of the explorer network performance by the following constraint:

$$| R_{Explorer} - R_{Expert} | < \beta_{Expert} \qquad (1)$$

This condition prevents from bad learning, with β_{Expert} as a significant threshold and a hyperparameter that we call the Expert's trust Margin, and are tweaked through experimentations. β_{Expert} value depends on the shaped reward function and on the rewards values the agent gets through its interaction with the environment. This condition was adopted to fix a boundary region in order to avoid any performance drop of the OCIRL model compared with the imitation learning network. At the end, the resulting policy is the Expert critic network that has progressed through the training while obeying the constraint relating to the Expert's trust margin.

Expert gradient clipping is a customized version of Gradient clipping [22] that limits the magnitude of the update step of the policy network's parameters to maintain training stability, it's a threshold value to keep the gradients from getting too high. For the OCIRL algorithm, the approach adopted is Gradient Value Clipping instead of Gradient Norm Scaling by taking into account the β_{Expert} value. The higher the value of β_{Expert} is, the higher is the risk margin that the expert critic network skills level will get lower than the initial expert level. So, to moderate and balance this risk, we apply in this case an intelligent clipping by tightening its margin depending on $\frac{1}{\beta_{Expert}}$ value.

$$\varepsilon^{Clip}(\beta_{Expert}) = \alpha . \frac{1}{\beta_{Expert}} \qquad (2)$$

$$J_{clip}(Q) = E_t(Min(J(Q), Clip(J(Q), 1 - \varepsilon^{Clip}(\beta_{Expert}), 1 + \varepsilon^{Clip}(\beta_{Expert})) \qquad (3)$$

The correlated combination of these two mechanisms Expert's trust region and Expert Gradient clipping based on the β_{Expert} value will balance the risk margin of divergence by limiting the magnitude of the network update when it is too high.

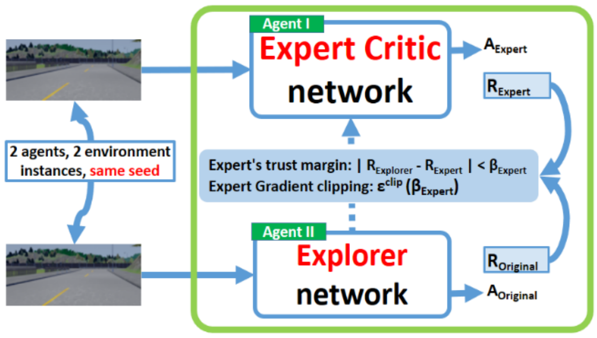
Agent I
Expert Critic network
A_{Expert}
R_{Expert}
2 agents, 2 environment instances, same seed
Expert's trust margin: $| R_{Explorer} - R_{Expert} | < \beta_{Expert}$
Expert Gradient clipping: $\varepsilon^{clip}(\beta_{Expert})$
Agent II
Explorer network
$R_{Original}$
$A_{Original}$

Figure 2. OCIRL architecture

```
Algorithm 1 OCIRL
1:  Inputs: $D^{PER}$: Prioritized experience replay memory, initialized with demonstration data set, $\theta_{Expert}$: weights for Expert critic network (random), $\theta_{Explorer}$: weights for Explorer network (random), $\tau$: frequency at which to update Expert critic network, k: number of pre-training gradient updates
2:  for steps t ∈ {1, 2, ... k} do
3:  Sample a mini-batch of n transitions from $D^{PER}$
4:  Calculate loss J(Q) using the Expert critic network
5:  Perform a gradient descent step to update $\theta$
6:  if t mod $\tau = 0$ then $\theta_{Explorer} \leftarrow \theta_{Expert}$ end if
7:  end for
8:  for steps t ∈ {1, 2, . . .} do
9:      Sample action from the Explorer Network's behavior policy $a \sim \pi_{\theta Explorer}$
10:     Play action a and observe (s', r)
```

```
11: Sample action from the Expert critic Network's
    behavior policy a ~ $\pi_{\theta Expert}$
12: Play action a and observe (s'', r')
13: Compute cumulative reward for both agents $R_{Expert}$ and
    $R_{Explorer}$
14: Store (s, a, r, s′) into $D^{PER}$, overwriting oldest self-
    generated transition if overcapacity
15: Sample a mini-batch of n transitions from $D^{PER}$
16: Calculate loss $J_{clip}(Q)=E_t(\min(J(Q), clip(J(Q, 1-$
    $\varepsilon^{Clip}(\beta_{Expert}), 1+\varepsilon^{Clip}(\beta_{Expert}))$ using Explorer
    network
17: Perform a gradient descent step to update $\theta_{Explorer}$
18: if t mod τ = 0 and | $R_{Explorer} - R_{Expert}$ | < $\beta_{Expert}$ ,
    then $\theta_{Expert} \leftarrow \theta_{Explorer}$ end if
19: s ← s′
20: end for
```

5. OCIRL IMPLEMENTATION BASED ON A2C

We choose to build our OCIRL model on the top of the Advantage Actor-Critic A2C DRL algorithm, that we call A2CfDoC which leverage the robustness of Actor-Critic methods that mix Policy Gradient and value-based approaches and also the trickiness of prioritized experience replay (PER) that proves its importance as the main factor that has significant impact on DRL models performance.

For the A2CfDoC network architecture, we apply Relu function to all the network's layers for non-linearity purposes, and batch normalization [14] after each convolutional layer and at the end of the fully connected network to standardize the inputs, allowing to use a higher learning rate and mini-batches which speed up the training.

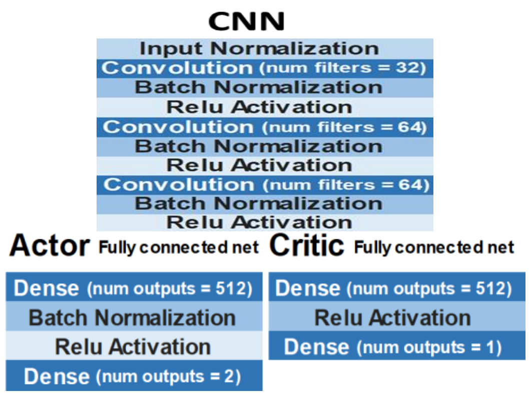

Figure 3. A2CfDoC: CNN, actor & critic FCN

5.1 Adapting the A2C model

To take advantage of expert demonstrations, A2CfDoC adopt the same approach as the DQfD algorithm, as we described in the previous chapter, the PER is filled in by expert's demonstrations data set which is used in the pre-training phase to initialize the DRL network with expert's skills in supervised learning way. In this pre-training off-policy phase, the data does not come from the agent's current behavioral policy but only from the PER memory, which served first and foremost to break the strong temporal /chronological relationship between transitions, and to learn to imitate the expert with a Q function that are compliant with the Bellman equation so we can apply TD updates once the agent begins to interact with his environment.

More concretely, we use a loss function inspired by the DQfD loss that combines between the A2C loss and the supervised large-margin classification loss, an imitation loss that comes on top of the standard TD-error. It is applied during the pre-training phase, where the DRL agent samples transitions from the PER expert's demonstrations and is encouraged to mimic expert's actions. For the training phase where the input data is self-generated, the supervised large-margin classification loss is not applied ($\lambda_E = 0$):

A2C Loss: $L_{A2C} = \frac{1}{2} L_{value} - L_{policy}$ with:

$$L_{value} = \sum (R - V(s))^2$$

$$and\ L_{policy} = -log(\pi(a|s)).A(s) - \beta.H(\pi(a|s)) \quad (4)$$

$$H_{entropy}(\pi) = -\sum P(x).log(P(x)) \quad (5)$$

Supervised large-margin classification loss:

$$L_E(Q) = max_{a\epsilon A}\left[Q(s,a) - l(a_{Expert}, a)\right] - Q(s, a_{Expert}) \quad (6)$$

$l(a_{Expert}, a)$ is the margin loss, with $l = 0$ when $a = a_{Expert}$ and positive otherwise.

$$L_{\text{A2CfDoC}} = \frac{1}{2} L_{value} - L_{policy} + \lambda_E . L_E \quad (7)$$

$$L_{\text{A2CfDoC}} = \frac{1}{2} L_{value} - L_{policy} + \lambda_E (l(a_{Expert}, a) - \bar{l}) \quad (8)$$

To give the model more robustness and the ability to learn more general features, we use dropout [26] in our case, instead of L2 regularization [27] loss that is mainly adopted for supervised learning to avoid overfitting during the training.

Being applied to the Self-driving car context, the A2CfDoC model must have the environment image as input and outputs two commands, the steering angle for the orientation and the speed to manage the acceleration and braking that are normalized Float values [-1.0, +1.0].

5.2 Simulation environment and data preparation

Reinforcement learning algorithms are always tested on the simulation environment due to the Markov Decision Process that rely on trial and error principle. These environments allow training the DRL agent without any real risks; and once the policy is good within the defined criteria, then the tests in ground truth can start to validate the learned policy in the simulation.

For this paper, we adopt the context of Self-driving cars with two open-source simulators that deal with continuous control tasks. The first is CarRacing-v0 [28], a basic environment made by openAI, specialized in lane-keeping and offer measures like speed, car position, steering wheel position and gyroscope, The second simulator is CARLA, an autonomous driving simulator with a high-fidelity realistic driving, developed jointly by Computer Vision Center, Intel

Labs and Toyota Research Institute and built on top of the UnrealEngine4 game engine. it supports camera raw image and provides direct measurements such as forward speed, orientation. CARLA requires a GPU to run, for the simulations we use a dual Xeon processor server with a Nvidia GPU Geforce GTX 1080Ti (3584 Cuda cores) [29].

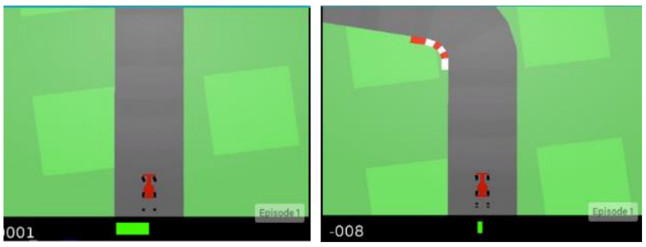

Figure 4. OpenAI carracing-v0 simulator environment

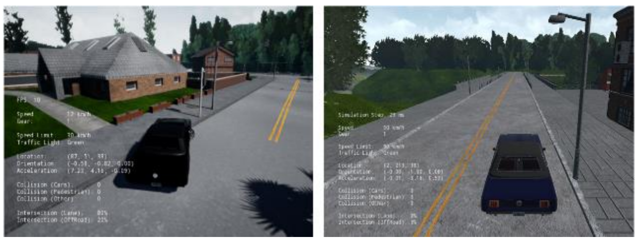

Figure 5. CARLA simulator environment

For both environment Carracing-v0 and CARLA, we design two intrinsic reward functions that will be used during imitation learning as a performance indicator for the expert demonstrations, and as a reward function in the RL phase [30].

The reward function that we choose for the Carracing-v0 is a basic one since only basic measurements are available in this environment:

$$Reward = \begin{cases} \cos(\theta) - \lambda \sin(\theta) - \frac{d}{w} & , if\ \theta < \frac{\pi}{2} \\ -2 & , \text{otherwise} \end{cases} \tag{9}$$

With $\in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$, d the distance to lane center, w is the half-width of the lane and λ a coefficient that adjusts the influence of θ. When the vehicle runs out of track or runs backward, we terminate the episode and penalize the action with a high penalty −2.

For CARLA, the environment affords more advanced measures that help to craft a more accurate reward function that encourages speed and penalizes intersecting lanes, off-road driving, collisions, and excessive turns.

$$\begin{aligned} Reward = {} & [capped\ speed] - \\ & \alpha_1 * [intersecting\ other\ lanes] - \\ & \alpha_2 * [off\ road] - \alpha_3 * [Collision] \\ & -\alpha_4 * |steer\ value| \end{aligned} \tag{10}$$

We choose $(\alpha_1, \alpha_2, \alpha_3, \alpha_4)$ = (100, 5, 10, 10).

Thereafter, we launch the simulation for the two environments to collect expert demonstrations, by recording videos using manual commands for Carraccing-v0 and the auto-pilot mode for CARLA and store the expert demonstrations of agent's experiences et=(st, at, rt, st+1) at each time-step t in a dataset D^{carla}_{t}={e1,…,et} and $D^{Carracing}_{t}$={e'1,…,e'$_t$}. As we indicate above, the reward is also calculated as a performance signal.

5.3 The pre-training

We populate the PER memory with the transitions from Expert Demonstrations and choose a medium buffer size to get better performance (with a total of 5.10^4 steps, instead of the 10^6 default value that is usually used). This part of the buffer filled with expert data will never be overwritten since it represents reliable and highly recommended transitions.

We start the off-policy training of the A2CfDoC model before starting to explore the environment for a set number of steps (60.000 for Carracing-v0 and 150.000 steps for CARLA), based only on the PER memory that contains the expert's demonstrations, like for supervised imitation learning phase [31]. This allows to initialize the model's network by expert's skills, and shorten the slow initial learning of DRL algorithms. At this phase, both explorer and expert critic networks are instantiated by the expert's driving skills.

5.4 Training phase

The next step starts by allowing the A2CfDoC two agents to interact with the Carracing-v0 and CARLA environments (with the same seed) and run for a set number of steps (respectively 200.000 and 450.000), with a lower exploration rate to prevent instability. During the training, the expert critic network brings stability to improving learning. It is a mandatory brick that allows constraining the update of the explorer network, whose role is to discover better driving ways and improve the initial expert policy. The synchronization between the explorer network and the expert critic network, only if the expert's trust margin reward is respected every N steps (10,000), to ensure that the A2CfDoC is in the right way to outperform the expert demonstrations, and to prevent from a performance drop.

Also, the use of Expert Gradient clipping limits the size of the update of the network that can provoke large changes from the previous policy, by keeping the update in a secured region and avoiding a dramatic decrease in performance.

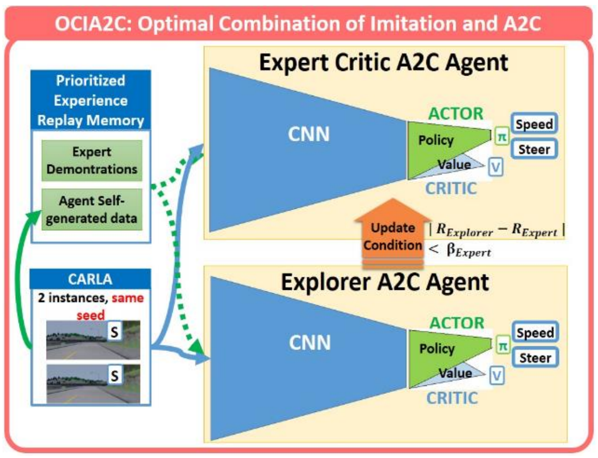

Figure 6. Optimal Combination of Imitation and A2C (A2CfDoC)

We also activate the prioritization for the experience replay memory, where we begin to inject the agent's self-generated sequences that have the higher importance transitions based on TD error criteria (the higher the TD value, the more interesting it is and the more often it is sampled), when the replay buffer is full, we replace the less important transitions by the new more important ones based on their priorities.

The resulting PER memory contains a mix of both a changing agent self-generated data and fixed and prioritized expert demonstrations data in the same buffer. During the training, the sampling from the PER is done from both populations with a higher probability for the expert demonstrations to avoid catastrophic forgetting.

6. EXPERIMENTS: APPLICATION TO SELF-DRIVING CAR

In this section, we first train our model with different values of the hyperparameter β_{Expert} the Expert's trust margin, to find out the right value that gives the best results, and then we assess the three models the A2C, the A2C from Demonstrations that we call A2CfD and our model, the A2CfDoC, by training them and assessing them based on the average cumulated reward. These two experiments were launched on both OpenAI Carracing-v0 and CARLA simulation environments.

6.1 Defining the Expert's trust margin

The β_{Expert} Expert's trust margin is a model hyper-parameter involved in two complementary mechanisms that balance the exploration vs stability of the algorithm, through the limitation of the risk of divergence by conditioning the network update and also, by clipping the loss magnitude to avoid important network changes.

For the OCIRL algorithm applied to A2C model, we have to define the optimal value of the β_{Expert} Expert's trust margin value that leads to an improvement of the learning level compared to the expert skills level, and we evaluate this performance by running the model with several β_{Expert} values (100, 300 and 1000).

The experimentations show that when β_{Expert} is high, the model underperforms and finishes by lowering the agent initial expert skills acquired during the pre-training phase using PER memory, it then fails to meet the expectations of improving the expert level. Also, when $\beta_{Expert}$is too low, the improvement margin is very small and the model stays at the same expertise level since the trust margin condition is mostly unfulfilled and the expert critic network is rarely updated. For the context of self-driving car in CARLA simulator, the best value of β_{Expert} is 300, it allows to capitalize on the expert demonstrations and push the learning further to outperform the cumulative reward of the teacher.

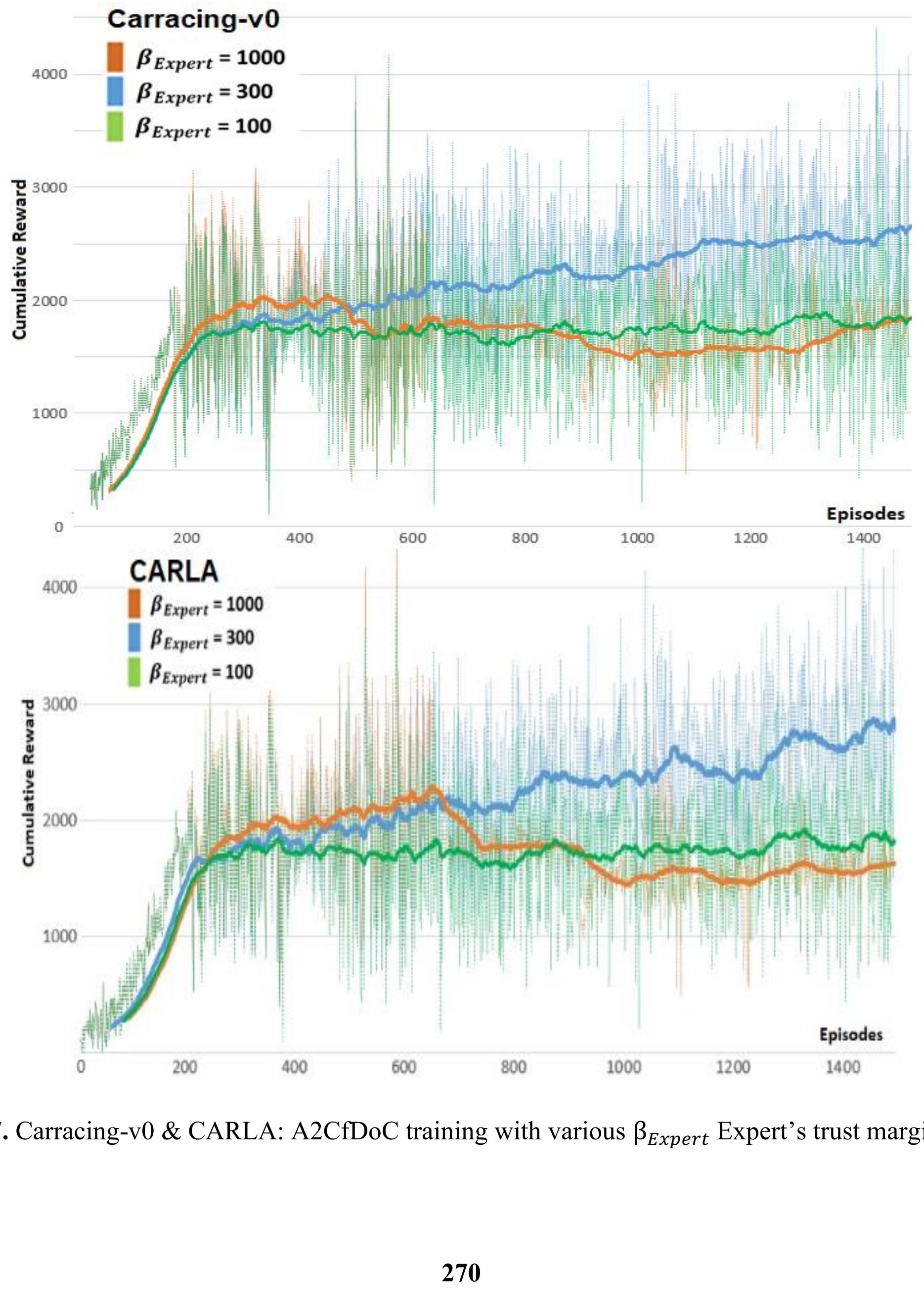

Figure 7. Carracing-v0 & CARLA: A2CfDoC training with various β_{Expert} Expert's trust margin values

6.2 Models training comparison

To assess our A2CfDoC model, we make a benchmark of three algorithms, the A2C model that is a pure Deep reinforcement learning algorithm, the A2C from demonstrations (A2CfD), which is an adaptation of A2C to the Deep Q-learning from Demonstrations (DQfD) that represents a successful combination of Reinforcement and Imitation Learning algorithm, but without any constraint on the learning process to preserve the expert knowledge during the RL training phase and finally our model the A2CfDoC which leverage DQfD principles and implement mechanisms that allow improving expert skills while avoiding any performance drop.

We can clearly notice in figure 8 that the moments when learning begins to improve the model, is different, and the A2C which is a pure DRL algorithm lags to trigger the learning, since it relies only on trial and error principle, while the A2CfD and the A2CfDoC that both combine imitation and RL, benefit from expert demonstrations to boost their learning at an early stage (the pre-training phase based on expert demonstrations). We also note for the A2CfD and A2CfDoC that the pre-training phases (200 episodes) are quite similar and that their behaviors start to change during the RL training phase where the progression of the cumulative reward became different.

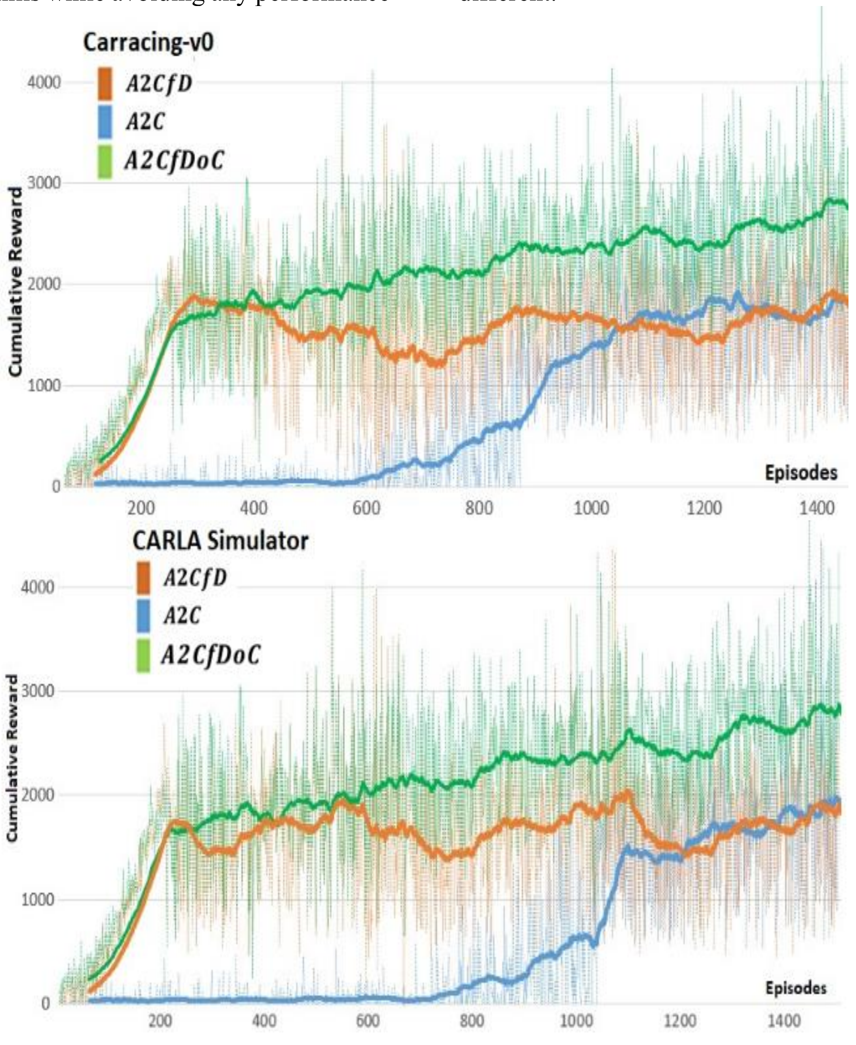

Figure 8. Carracing-v0 & CARLA: training A2C, A2C with demonstrations and our algorithm A2CfDoC

For the A2CfDoC model, the cumulative reward continues to grow even after the pre-training based on expert demonstrations. The A2CfDoC model shows more steadiness and does not suffer from the fluctuation of the A2CfD algorithm, which proves that our model succeeds at capitalizing on the learned skills, avoid any drop of performance and manage to stabilize the improvement of learning.

6.3 Testing results

We trained the three agents RL (A2C, A2CfD, A2CfDoC) and a behavioral cloning (BC/IL) agent on CARLA's Town1 circuit and then we test their saved models on both Town1 and Town2 (another different CARLA circuit), that have respectively 1.4 km and 2.9 km of drivable roads, and this for 100 episodes each one, in order to assess their performance, their adaptation capacities and their abilities to reach a given destination. For both circuits, we apply the same driving conditions (sunny day), and for simplification, we choose to drive an empty road with no pedestrian and no other vehicles.

The episode is considered successful for a fixed duration when the agent made his tour without colliding any obstacle in a blocking manner. The episodes' data have been recorded and stored to analyze the agent's behavior during the episodes.

Finally, we get the following summary of results:

During the tests, we notice that the BC model outperforms the A2C RL model for the Town1 for which it was already trained on, but underperform the RL model for the circuit Town 2 since it's an unseen environment. Also, the two algorithms A2CfD and A2CfDoC that mixed IL and RL succeed in outclassing both RL (A2C) and IL (BC) models on the two circuits Town1 and Town2, which proves their strengths and their high adaptability, however, we note a better performance and adaptation of our model that excels on the two circuits and outstrip all the tested models including A2C, BC, and the combined model A2CfD, and we noticed especially, its good result for the Town2 that represent a new circuit that it was not trained for.

Table 1. Success rate of the three assessed models A2C, BC(IL), A2CfD and A2CfDoC

	Success Rate			
	A2C	BC(IL)	A2CfD	A2CfDoC
Town 1	58%	64%	63%	71%
Town 2	48%	43%	51%	57%

7. CONCLUSION

When comparing supervised imitation learning (IL) and Deep Reinforcement Learning (DRL) methods, we can rapidly note the dissimilarity of their training signals, the rewards vs the demonstrations, and using a naive combination of the two approaches could lead to catastrophic results. So many endeavors have been made by researchers to associate both algorithms and try to benefit from their strengths.

In this paper, we presented a novel hybrid algorithm that combines DRL and IL to achieve human-level performance on the Self-driving car simulation environments OpenAI Carracing-v0 and CARLA. It succeeds in learning the expert skills during the pre-training phase and allows surpassing this expertise level using a constrained and more intelligent RL exploration process, which has secured the taught expertise and led to the acquisition of more advanced abilities.

The use of such an algorithm in the context of autonomous vehicles has a very important benefit since it accelerates the learning and at the same time, it avoids any drop of performance that the RL agent could suffer from. By combining two correlated principles the Expert's trust margin and the Expert Gradient clipping, the OCRIL model succeed in balancing the risk of divergence and promote the learning progression toward higher expertise and better performance.

A next step could be a combination of other techniques that can solve the many challenges that Artificial Intelligence face in building an 100% reliable ADAS systems, by using a hierarchical deep learning network architecture to form a true whole single network that can deal with complex tasks and include other sub-function like sensor fusion, occupancy grid mapping and path planning, or handle several macro features going from pedestrians detection, road-sign recognition and Collision Avoidance to some more complex one like self-parking, lane-keeping and cruise control. Also we can combine Partial Observability Markov Decision Process (POMDP) principle to provide the deep learning network with the ability to deal with limited spatial and temporal perception of the environment by using RNN/LSTM to predict what it has not been sensed, by a historical dataset of past visual features that compensate the lack of information. Finally yet importantly, we can use inverse reinforcement learning to solve the reward sparsity problem by deriving a reward function and the goals to achieve from observed expert behavior using supervised deep learning.

REFERENCES

[1] Sutton & Barto Book: Reinforcement Learning: An Introduction. http://incompleteideas.net/book/the-book.html, accessed on 27-May-2019.

[2] Artificial intelligence: A modern approach. http://aima.cs.berkeley.edu/, accessed on 28-May-2019.

[3] Tijsma, A.D., Drugan, M.M., Wiering, M.A. (2016). Comparing exploration strategies for Q-learning in random stochastic mazes. 2016 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 1-8. https://doi.org/10.1109/SSCI.2016.7849366

[4] Attia, A., Dayan, S. (2018). Global overview of imitation learning. arXiv:1801.06503 [cs, stat], Jan. 2018.

[5] [1411.1792] How transferable are features in deep neural networks? Available: https://arxiv.org/abs/1411.1792, accessed on 27-May-2019.

[6] [1511.05952] Prioritized experience replay. https://arxiv.org/abs/1511.05952, accessed on 27-May-2019.

[7] Wang, H., Zariphopoulou, T., Zhou, X. (2019). Exploration versus exploitation in reinforcement learning: A stochastic control approach. arXiv:1812.01552 [cs, math], Feb. 2019.

[8] Hester, T., Vecerik, M., Pietquin, O., Lanctot, M., Schaul, T., Piot, B., Horgan, D., Quan, J., Sendonaris, A., Osband, I., Dulac-Arnold, G., Agapiou, J., Leibo, J., Gruslys, A. (2017). Deep Q-learning from demonstrations. arXiv:1704.03732 [cs], Apr. 2017.

[9] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A.,x Antonoglou, A., Wierstra, D., Riedmiller, M. (2013). Playing Atari with deep reinforcement learning. arXiv:1312.5602 [cs], Dec. 2013.

[10] Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L., Monfort, M., Muller, U., Zhang, J., Zhang, X., Zhao, J., Zieba, K. (2016). End to end learning for self-driving cars. arXiv:1604.07316 [cs], Apr. 2016.

[11] Mnih, V., Badia, A., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. arXiv:1602.01783 [cs], Feb. 2016.

[12] Fenjiro, Y., Benbrahim, H. (2018). Deep reinforcement learning overview of the state of the art. JAMRIS, 12(3): 20-39. https://doi.org/10.14313/JAMRIS_3-2018/15

[13] Konda, V. (2002). Actor-critic algorithms. PhD Thesis, Massachusetts Institute of Technology, Cambridge, MA, USA.

[14] [1805.01954] Behavioral cloning from observation. https://arxiv.org/abs/1805.01954, accessed on 28-Nov-2019.

[15] Pomerleau, D.A. (1989). ALVINN: An autonomous land vehicle in a neural network. presented at the Advances in Neural Information Processing Systems, pp. 305-313.

[16] Muller, U., Ben, J., Cosatto, E., Flepp, B., Cun, Y.L. (2006). Off-road obstacle avoidance through end-to-end learning. in Advances in Neural Information Processing Systems 18, Y. Weiss, B. Schölkopf, and J. C. Platt, Eds. MIT Press, pp. 739-746.

[17] Ross, S., Gordon, G.J., Bagnell, J.A. (2010). A reduction of imitation learning and structured Prediction to No-Regret Online Learning. arXiv:1011.0686 [cs, stat], Nov. 2010.

[18] Ross, S., Bagnell, J.A. (2014). Reinforcement and Imitation Learning via Interactive no-regret learning. arXiv:1406.5979 [cs, stat], Jun. 2014.

[19] Daumé III, H., Langford, J., Marcu, D. (2009). Search-based structured prediction. arXiv:0907.0786 [cs], Jul. 2009.

[20] Ross, S., Bagnell, J.A. (2010). Efficient reductions for imitation learning. AISTATS.

[21] Mnih, V., Badia, A., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., Kavukcuoglu, K. (2016). Asynchronous Methods for Deep Reinforcement Learning. arXiv:1602.01783 [cs], Feb. 2016.

[22] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O. (2017). Proximal policy optimization algorithms. arXiv:1707.06347 [cs], Jul. 2017.

[23] Wu, Y., Mansimov, E., Liao, S., Grosse, R., Ba, J. (2017). Scalable trust-region method for deep reinforcement learning using Kronecker-factored approximation. arXiv:1708.05144 [cs], Aug. 2017.

[24] Sun, W., Bagnell, J.A., Boots, B. (2018). Truncated horizon policy search: Combining reinforcement learning & imitation learning. arXiv:1805.11240 [cs, stat], May 2018.

[25] Cheng, C.A., Yan, X., Wagener, N., Boots, B. (2018). Fast policy learning through imitation and reinforcement. arXiv:1805.10413 [cs, stat], May 2018.

[26] Dropout: A simple way to prevent neural networks from overfitting. http://jmlr.org/papers/v15/srivastava14a.html, accessed on 04-May-2019.

[27] Deep Learning. https://srdas.github.io/DLBook/ImprovingModelGeneralization.html, accessed on 04-May-2019.

[28] OpenAI. (2019). Gym: A toolkit for developing and comparing reinforcement learning algorithms. https://gym.openai.com, accessed on 04-May-2019.

[29] Chauvin, S. (2018). Hierarchical decision-making for autonomous driving. https://doi.org/10.13140/RG.2.2.24352.43526

[30] Schorner, P., Tottel, L., Doll, J., Zöllner, J. (2019). Predictive trajectory planning in situations with hidden road users using partially observable Markov decision processes. 2019 IEEE Intelligent Vehicles Symposium (IV), pp. 2299-2306. https://doi.org/10.1109/IVS.2019.8814022

[31] Sharifzadeh, S., Chiotellis, J., Triebel, R., Cremers, D. (2016). Learning to drive using inverse reinforcement learning and deep Q-networks.

NOMENCLATURE

RL	Reinforcement learning
DRL	Deep Reinforcement Learning
IL	Imitation learning
BC	Behavioral Cloning
PER	Prioritized Experience Replay
A2C	Advantage Actor-Critic algorithm
A2CfD	Advantage Actor-Critic algorithm from Demonstrations
A2CfDoC	Advantage Actor-Critic algorithm from Demonstrations optimally Constrained

Greek symbols

β_{Expert}	Expert's trust margin threshold
ε^{Clip}	Expert Gradient clipping hyperparameter
L_{A2CfDoC}	A2CfDoC loss function
$R_{Explorer}$	The Explorer Network Reward
R_{Expert}	The Expert Network Reward