Combining Deep Deterministic Policy Gradient with Cross-Entropy Method

Tung-Yi Lai*, Chu-Hsuan Hsueh*†, You-Hsuan Lin*, Yeong-Jia Roger Chu*, Bo-Yang Hsueh* and I-Chen Wu*‡
*Department of Computer Science, National Chiao-Tung University, Hsinchu, Taiwan
†School of Information Science, Japan Advanced Institute of Science and Technology, Nomi, Ishikawa, Japan
‡Pervasive Artificial Intelligence Research (PAIR) Labs, Hsinchu, Taiwan
Email: icwu@cs.nctu.edu.tw

***Abstract*—This paper proposes a deep reinforcement learning algorithm for solving robotic tasks, such as grasping objects. We propose in this paper a combination of cross-entropy optimization (CE) with deep deterministic policy gradient (DDPG). More specifically, where in the CE method, we first sample from a Gaussian distribution with zero as its initial mean, we now set the initial mean to DDPG's output instead. The resulting algorithm is referred to as the DDPG-CE method. Next, to negate the effects of bad samples, we improve on DDPG-CE by substituting the CE component with a weighted CE method, resulting in the DDPG-WCE algorithm. Experiments show that DDPG-WCE achieves a higher success rate on grasping previously unseen objects, than other approaches, such as supervised learning, DDPG, CE, and DDPG-CE.**

***Index Terms*—reinforcement learning, robotics, object grasping, deep deterministic policy gradient, cross-entropy method**

I. Introduction

Deep reinforcement learning (DRL) has been successfully applied to many domains. For example, agents trained using deep Q-networks surpassed human experts in Atari games [1], AlphaGo Zero defeated top professional players in Go [2]. Recently, applying DRL to robotic applications, such as pushing or grasping objects using robotic arms, has been brought to attention in the field of reinforcement learning [3]–[5].

For robotic applications, action spaces are often continuous. For example, the degree of rotation for robotic arms can be between 0° and 360°. For tasks in continuous action spaces, one approach is using deep deterministic policy gradient (DDPG) algorithm, proposed by Lillicrap *et al.* [3], to learn a deterministic policy in continuous action space. DDPG is a kind of actor-critic algorithm where an actor generates an action and then a critic evaluates the action. Other approaches for tasks in continuous action space are proposed by Kalashnikov *et al.* [4] and Levine *et al.* [5]. Their approach takes a state and an action as inputs, and outputs a success rate for grasping. Different from DDPG, instead of having an actor generating actions, their algorithm generates actions by sampling. In order to generate better samples, they used cross-entropy (CE) method [6], a Monte Carlo method for optimization, to obtain a good action efficiently.

978-1-7281-4666-9/19/$31.00 ©2019 IEEE

This paper reviews related work in Section II, and then proposes in Section III, (i) an algorithm DDPG-CE that combines CE method with DDPG, and (ii) the weighted CE (WCE) method, an enhancement of the CE method. In DDPG-CE, the CE method uses the actions generated by the actor of DDPG as the initial mean of a Gaussian distribution, and then samples from the distribution. The critic of DDPG evaluates the quality of the samples. However, in a binary reward environment, samples generated by CE method may rarely be located around the optimum. The samples far away from the optimum may mislead the mean of CE method. To overcome this problem, WCE method assigns weights to the elites generated by CE method based on the values provided by the critic of DDPG, and computes the weighted mean and the weighted variance from these elite samples. Thus, the weights of samples around the optimum are higher and hence the weighted mean will be closer to the optimum. We compare the performance of DDPG-WCE with supervised learning, DDPG, CE method and DDPG-CE. The experiments in Section IV show that DDPG-WCE outperforms other approaches in the testing tasks with previously unseen objects. Finally, we make a conclusion in Section V.

II. Related Work

Action spaces for robotic applications are usually continuous. For example, the rotation angle for each joint of a robot arm could be any angle between 0° and 360°. A well-known method for solving tasks in continuous action spaces is DDPG, proposed by Lillicrap *et al.* [3]. DDPG learns a deterministic policy in continuous action space and consists of 2 components: an actor and a critic. The actor requires a state s as input and outputs a deterministic action $a = \pi_\theta(s)$ of the state, where θ is the parameters of the actor. The critic requires a state s and an action a as inputs and predicts the Q-value $Q_\phi(s, a)$ to score the quality of the action, where ϕ is the parameters of the critic.

Other approaches for tasks in continuous action spaces are proposed by Kalashnikov *et al.* [4] and Levine *et al.* [5]. Their approach takes a state and an action as inputs, and outputs a success rate for grasping. Different from DDPG, instead of having an actor generate actions, their algorithm generates actions by sampling. In order to generate better

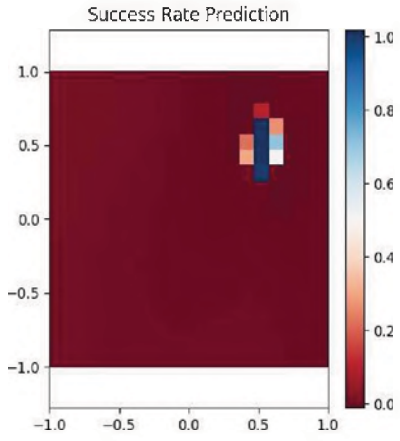

Fig. 1. An example Q-value on a 2-D action environment.

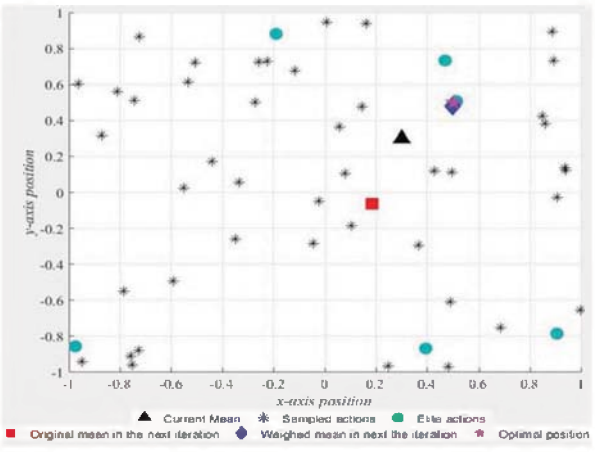

Fig. 2. The results of the CE method and the WCE method.

samples, they use cross-entropy (CE) method [6], a Monte Carlo method for optimization, to obtain a good action efficiently. In general, the CE method consists of 2 phases, which are executed repeatedly: (i) generate some random samples according to a specific mechanism, evaluate the quality of the samples, and choose the top M samples as *elite* samples, where M is a hyperparameter; (ii) update the parameters of the sampling mechanism based on the elites, and then produce better samples in the next iteration. The cross-entropy term is involved in the second phase of the CE method to update the parameters of the sampling mechanism. Zdravko *et al.* [7] has shown that when the sampling mechanism of the CE method is a Gaussian distribution, there is a closed-form solution for updating its mean and variance.

III. Approaches

This section presents the DDPG-CE algorithm and the fake-elite problem in Subsection III-A. Subsection III-B then describes the enhanced algorithm DDPG-WCE, which is designed to deal with the fake-elite problem.

A. DDPG-CE

We propose an algorithm DDPG-CE that combines CE method with DDPG. The actions provided by the actor of DDPG can be improved through the optimization of the CE method. The process of DDPG-CE is as follows: (i) We train a model using DDPG and fix the parameters of the model. (ii) We use the DDPG's actor to generate an action given a state, and then uses the action as initial mean for the Gaussian distribution of CE method. This is slightly different from Levine *et al.*'s work [5] in that their initial mean was set to zero. (iii) In the first phase of CE method, the samples of actions are generated by the Gaussian distribution, and are then evaluated by the DDPG's critic. (iv) In the second phase of the CE method, M samples with the highest Q-values are chosen as elites, and then are used to compute the mean and variance for the next iteration. (v) Repeat (iii) and (iv) multiple times to obtain a good action.

In our approach, we use a binary reward setting, where the agent gets a reward of 1 when success and 0 otherwise, because binary rewards are easy to obtain in most applications and thus are general. However, under a binary reward environment, the Q-value function is nearly a step function whose value is close to one in a certain optimal spots and zero otherwise. Figure 1 illustrates an example of Q-value function predicted by the critic of DDPG. Under such a scenario, the samples of CE method may rarely be located around the optimal spots, and elites far away from the optimum may mislead the mean of the next iteration. To simplify the discussion, we use a 2-D action environment as an example to explain. Figure 2 illustrates the problem, where the optimum is at (0.5, 0.5) and the initial mean of CE method is set to (0.3, 0.3), which is close to the optimal spot. The optimum and the initial mean are marked with a star and a triangle respectively in Figure 2. In the first phase, the CE method samples from the Gaussian distribution with mean (0.3, 0.3), the resulting samples are marked with $*$ in Figure 2. Then the CE method chooses M samples with the highest Q-values as elites, which are marked with circles in Figure 2 with $M = 6$. Among the six elites, only two are near the optimum, while the other four are far away. We call the elites far away from the optimum as *fake* elites. The reason to have fake elites is explained in more details as follows. Although the Q-values of the actions far away from the optimum are all close to zero, the critic is still not perfect. It may predict small values such as 10^{-3} or 10^{-5}. In addition, the CE method always selects M elites. Thus, we may need to select better ones among bad actions, which results in fake elites. In the second phase, the CE method uses the 6 elites to compute the mean of the next iteration, which is marked with a square. This demonstrates the effect of fake elites misleading the mean of the next iteration.

B. DDPG-WCE

To overcome the fake-elite problem, we propose weighted cross-entropy (WCE) method. For a state s, the WCE method assigns each elite action a_i a weight w_i that is the exponential of its corresponding Q-value $Q_\phi(s, a_i)$ divided by T as shown

in Equation (1), and then calculates the weighted mean μ and variance σ^2 of these elites for the next iteration. The hyperparameter T determines how close the weighted mean and original mean are. Larger T makes the weights of all elites closer, and therefore the mean tends to be closer to the original mean of the CE method. When $T \to \infty$, the WCE method is equivalent to CE method. In contrast, smaller T makes the weights higher for the elites with higher Q-values. Thus, the mean tends to be closer to elites with the higher Q-values and ignores elites with smaller Q-values. Figure 2 shows the effect. With $T = 0.2$, the weighted mean, marked with a diamond, moves to the optimal spots.

$$
\begin{aligned}
w_i &= e^{\frac{Q_\phi(s,a_i)}{T}}, T \in (0, \infty) \\
\mu &= \frac{\sum_i w_i a_i}{\sum_i w_i} \\
\sigma^2 &= \frac{\sum_i w_i (a_i - \mu)^2}{\sum_i w_i}
\end{aligned}
\tag{1}
$$

We replace the CE method of DDPG-CE with the WCE method, and propose DDPG-WCE. The difference between DDPG-CE and DDPG-WCE is the method to compute the mean and the variance of the next iteration. The whole process of DDPG-WCE is shown in Algorithm 1. Besides calculating the weighted mean and variance, we also make an enhancement to improve the optimization of DDPG-WCE: Record the best action during the search and return that as the search result. The experiments are shown in Section IV.

Algorithm 1 DDPG-WCE

Require: actor π_θ, critic Q_ϕ, state s, initial variance σ^2, population size N, elite size M, and iteration number R

Ensure: A deterministic action a^*

1: Initialize $\mu = \pi_\theta(s)$
2: **for** $i = 1 : R$ **do**
3: Sample N actions $(a_1, ..., a_N)$ from the Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$;
4: Add μ into the sampled action set $A = \{\mu\} \cup \{a_1, ..., a_N\}$;
5: Calculate $Q_\phi(s, a) \forall a \in A$;
6: Set $a^* = \text{argmax}_{a \in A \cup \{a^*\}} Q_\phi(s, a)$;
7: Update the μ and σ^2 by Equation (1) according to the top M elite actions in A;
8: **end for**
9: **return** a^*

IV. Experiments

This section presents the experiments, including environment settings in Subsection IV-A and results in Subsection IV-B.

A. Environment settings

The experiments are conducted based on a robotic grasping task. We setup a simulation environments under V-REP 3.5 [8]. We designed 18 blocks, where each consists of one to six $5 \times 5 \times 5$ cm^3 cubes, as grasping objects and the grasping agent is a robotic arm with a gripper 8.5 cm wide. Amongst the 18 blocks, 7 are chosen for training while the remaining 11 are for testing, as illustrated in Figure 3. In each episode, one block is randomly selected from the training/testing set and placed in a 30×30 cm^2 working space with a random rotation. The state of the grasping environment is represented by a 128×128 full-color image and a 128×128 gray-scale depth image captured by an overlooking camera. The action is a 3-dimensional vector with continuous values corresponding to the x-position, y-position and the z-axis rotation of the gripper. The agent receives a reward of 1 when the grasping is successful and 0 otherwise.

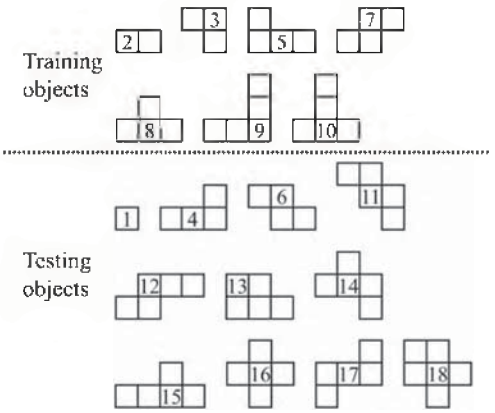

Fig. 3. Designed blocks.

Our actor network and critic network of DDPG consist of 4 convolutional layers with 32 channels, stride of 2, and kernel sizes of 7, 5, 5, and 3 respectively. The output of the last convolutional layer is flattened into a one-dimensional vector and passes through 2 fully connected layers with 300 and 200 hidden units respectively. Finally, a fully connected layer is used to shape the output to a desired dimension (3 for actor and 1 for critic). Additionally, the extra action input for the critic is concatenated with the flattened output of the last convolutional layer.

For the training process, the batch size is 256 and the replay buffer size is 1024, which is prefilled with 512 experiences under uniform random policy before the training process starts. The networks are trained for a total of 50,000 iterations. We also use prioritized experience replay [9], with hyperparameters α and β, where α is set to 0.7 and β is initialized to 0.4 and then linearly increased to 1.0 in the first 20,000 iterations. We use Adam optimizer [10] with L2-norm regularization and the learning rate is set to 0.001. Our exploration strategy is a variant of ϵ-greedy. With probability ϵ, we sample a random action from a Gaussian distribution centering at the actor's action; otherwise we greedily follow the actor's action. ϵ is initialized to 100% and linearly decreased to 5% in the first 20,000 iterations; and the standard deviation of the Gaussian distribution is initialized to 0.5 and linearly decreased to 0.015

TABLE I
THE SUCCESS RATES OF DIFFERENT APPROACHES FOR TRAINING AND TESTING. THE BEST ARE IN **BOLD**

Approach	Training	Testing
SL	**97.71**±1.11%	40.18±2.90%
DDPG	94.71±1.66%	57.00±2.93%
CE method	79.00±3.02%	53.09±2.95%
DDPG-CE	97.00±1.26%	60.73±2.89%
DDPG-WCE	97.57±1.14%	**62.91**±2.85%

in the first 20,000 iterations. If the action sampled from the Gaussian distribution is outside of the working space, we uniformly sample another one within the working space to replace it. As for CE method and WCE method, the iteration number R is 3, the population size N is 64, the elite size M is 6, and the initial variance of Gaussian distribution σ^2 is set to 0.09. The hyperparameter T of the WCE method is 0.2.

B. Experiment results

1) Compare with baseline: We compare the performance of the following five approaches. (i) the supervised learning (SL) method, which is trained by predefined grasping poses, (ii) DDPG, which uses the actor to provide actions, (iii) CE method, which uses the critic of DDPG to evaluate the sampled actions, (iv) DDPG-CE and (v) DDPG-WCE. The DDPG networks for (ii) to (v) are the same. All the approaches are evaluated by the average success rates of all blocks, where each block is tested for 100 episodes.

Table I shows that DDPG-WCE outperforms other approaches in the previously unseen (testing) objects, and only slightly worse than the SL method in the training case. Compared with DDPG and the CE method, DDPG-WCE increases the success rate by 5.91% and 9.82% respectively in the testing cases, where the success rate is 62.91%. If we only consider some blocks that are well learned, like block_4, block_6, block_11, block_12, block_14 and block_15, the success rate goes up to 77.33%. Interestingly, the simplest block, block_1, has a much lower success rate, which is open to be investigated.

The performance of the CE method is worse than DDPG, which means that using an actor to generate action is better than using the CE method to generate actions with an initial mean of zero. On the contrary, with better initialization, DDPG-CE improves the success rate of DDPG by 3.73%. Note that the SL method outperforms other approaches in the training case, but the success rate in testing case is only 40.18%, which is lower than DDPG-WCE by 22.73%.

2) Analyze Enhancements of Weighted Cross-Entropy Method: There are 3 features of our WCE, namely: (i) Initializing the search with actor's action, (ii) calculating weighted mean and variance for Gaussian distribution, and (iii) recording the best action during the search and using it as the search result. To verify the effectiveness of each enhancement, we separately remove each of them from WCE and compare them with WCE and the basic CE. All approaches run the optimization for 3 iterations.

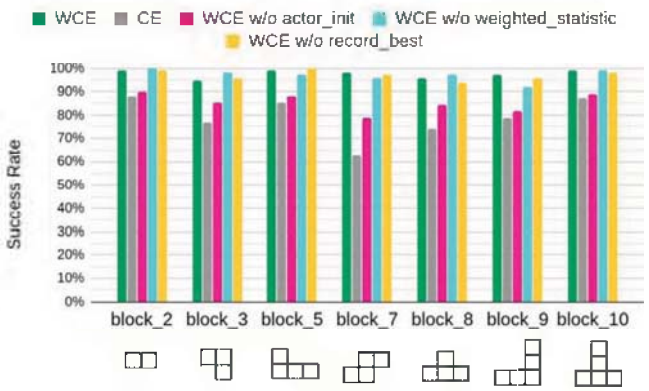

Fig. 4. Success rates on training objects of different CE settings.

Figure 4 shows the success rates on training objects. "w/o actor_init" means the WCE starts the optimization process with the initial mean of the Gaussian distribution set to (0, 0, 0). "w/o weighted_statistic" refers to the mean and variance of the elite data is calculated without weights. "w/o record_best" means to return the mean of the elites in the last iteration instead of the action with the max Q-value during the whole optimization process.

WCE outperforms basic CE in all training objects. Among all enhancements, "w/o actor_init" reduces the success rate the most, and the effects of removing "weighted_statistic" and "record_best" seem to be similar. Since the actions predicted by the actor is usually very close to the Q-value's optimums, if the mean of the Gaussian distribution is set to the actor's prediction, there would usually be enough "elite actions" located in the optimal region in the samples. As a result, "actor_init" can improve the grasping success rates. With better initialization already, the contributions from "weighted_statistic" and "record_best" are relatively small. Removing either of them does not greatly decrease the success rates because the remaining mechanism can still handle the fake-elite problem.

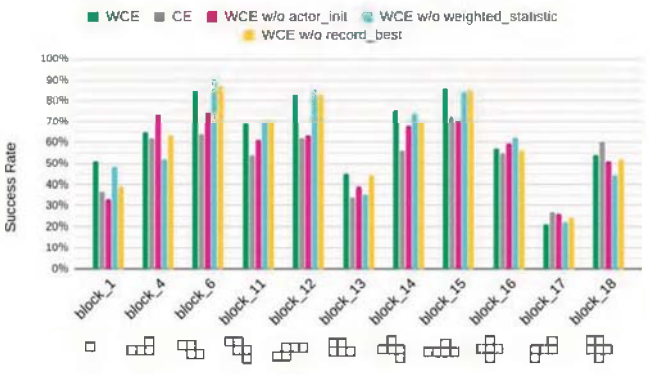

Fig. 5. Success rates on testing objects of different CE settings.

Figure 5 shows the success rates on testing objects which are unseen by the agent during training. Generally, each enhancement keeps the same tendency on the testing objects. WCE performs better than CE, "actor_init" is still a significant enhancement, and the performances of "weighted_statistic" and "record_best" are similar. However, the success rate of each enhancement does not follow the tendency in some blocks, such as block_4, block_16, and block_17. We speculate that this is because the Q-values predicted by the critic may be inaccurate for testing objects, and the inaccurate Q-values further disturb the optimization. In addition, we can see that "record_best" is more useful than "weighted_static" on block_1. Block_1 is a cube and there are multiple optimal actions to grasp a cube. The result demonstrates the "record_best" enhancement can handle optimization with multiple optimums by recording at least one optimum instead of returning a mean of multiple optimums.

V. Conclusions

We propose an algorithm DDPG-CE that combines the CE method with DDPG algorithm. We further overcome the fake-elite problem of the CE method in binary reward environments with WCE method, and modify DDPG-CE into DDPG-WCE. Our WCE features (i) using actions generated by the actor as initial mean, (ii) calculating weighted mean and variance for Gaussian distribution in the next iteration, and (iii) recording best action during the whole process of the optimization. We show the effects of the three features of WCE by removing one at a time and comparing their success rates. The experiments show that using actor's actions as initial means influences the most. Overall, DDPG-WCE outperforms supervised learning method, DDPG, the CE method and DDPG-CE for previously unseen testing objects.

The followings discuss some future research directions. One is to combine CE or WCE with DDPG's training process. In this work, we fix the trained DDPG during the optimization of CE/WCE. With the success of AlphaGo Zero [2], which improves the policy network by conducting searches, we think DDPG may be further improved with combining search such as CE/WCE. It is also worthy investigating to speed up CE/WCE since incorporating CE/WCE into the training of DDPG is expected to increase the training time (mainly from evaluation of sampled actions).

Acknowledgment

This research is partially supported by the Ministry of Science and Technology (MOST) of Taiwan under Grant Number 108-2634-F-009-011 through Pervasive Artificial Intelligence Research (PAIR) Labs, and also partially supported by the Industrial Technology Research Institute (ITRI) of Taiwan under Grant Number 108-EC-17-A-21-1516.

References

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[2] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, "Mastering the game of Go without human knowledge," *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[3] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *4th International Conference on Learning Representations (ICLR 2016)*, 2016.

[4] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, and S. Levine, "Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation," in *2nd Annual Conference on Robot Learning (CoRL 2018)*, 2018.

[5] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *The International Journal of Robotics Research*, vol. 37, no. 4-5, p. 421–436, Jun 2017.

[6] R. Rubinstein, "The cross-entropy method for combinatorial and continuous optimization," *Methodology And Computing In Applied Probability*, vol. 1, no. 2, pp. 127–190, Sep 1999.

[7] Z. I. Botev, D. P. Kroese, R. Y. Rubinstein, and P. L'Ecuyer, "Chapter 3 - the cross-entropy method for optimization," in *Handbook of Statistics*. Elsevier, 2013, vol. 31, pp. 35–59.

[8] *V-REP Simulator*. [Online]. Available: http://www.coppeliarobotics.com

[9] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," in *4th International Conference on Learning Representations (ICLR 2016)*, 2016.

[10] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations (ICLR 2015)*, 2015.

[11] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, "Tensorflow: A system for large-scale machine learning," in *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, 2016, pp. 265–283.