Real-time Policy Distillation in Deep Reinforcement Learning

Yuxiang Sun
Computer Science and Engineering Department
University of South Carolina
syuxiang@email.sc.edu

Pooyan Fazli
Department of Computer Science
San Francisco State University
pooyan@sfsu.edu

Abstract

Policy distillation in deep reinforcement learning provides an effective way to transfer control policies from a larger network to a smaller untrained network without a significant degradation in performance. However, policy distillation is underexplored in deep reinforcement learning, and existing approaches are computationally inefficient, resulting in a long distillation time. In addition, the effectiveness of the distillation process is still limited to the model capacity. We propose a new distillation mechanism, called *real-time policy distillation*, in which training the *teacher* model and distilling the policy to the *student* model occur simultaneously. Accordingly, the teacher's latest policy is transferred to the student model in real time. This reduces the distillation time to half the original time or even less and also makes it possible for extremely small student models to learn skills at the expert level. We evaluated the proposed algorithm in the Atari 2600 domain. The results show that our approach can achieve full distillation in most games, even with compression ratios up to 1.7%.

1 Introduction

Recent research has demonstrated the power of reinforcement learning to learn effective control policies in various domains. One breakthrough is the deep-Q network (DQN) [7]. In this approach, applied to the Atari domain, consecutive screenshots of games are fed as input to the network, and the network is expected to predict the best action that maximizes the expected future reward. Using this approach, a well-trained agent can outperform human-expert players in most Atari games. However, DQN models are usually cumbersome, which prevents their applications to compute- and power-constrained devices, such as drones and cell phones, that require efficient networks. To this end, policy distillation techniques have been widely investigated [10, 13, 8] for their ability to transfer a policy to a compressed model with acceptable performance loss, primarily using distribution regression. However, most existing policy distillation methods require a long time to complete the distillation process, fail in some task domains or in providing full distillation, and have an approximation capacity that limits the model compression ratio.

In this work, we mainly focus on the quality of policy distillation and address the challenges of time and approximation capacity. First, we propose a new policy distillation method, in which training the teacher and distilling the policy to the student are carried out simultaneously in real time. This method is different from typical policy distillation approaches [10], in which the teacher model is trained, and then the student is trained via distillation under the supervision of the teacher model. Second, instead of optimizing the student's policy only based on the output distribution regression between the teacher and student models [10], we import a *self-learning* term in the loss function to help regulate the student's behavior in an additional way. Third, beyond the supervision via distribution regression, we employ another channel of supervision by asking the student to imitate how the teacher sets

reasonable targets. Finally, we investigate different variations of the loss function and their effects on the quality of the distillation process.

2 Real-time Policy Distillation

In policy distillation, one or more deep reinforcement learning models are trained as expert models that can provide supervision to student training. Distillation techniques such as prediction regression [3] and feature regression [1, 9] have been comprehensively explored in different aspects of image classification. In deep reinforcement learning, KL divergence has proved to be the most effective method to date [10]. It is formulated as

$$\mathcal{L}_{KL}(\theta^{(S)}) = \sum_{i=1}^{N} softmax(\frac{q_i^{(T)}}{\tau})ln\left(\frac{softmax(\frac{q_i^{(T)}}{\tau})}{softmax(\frac{q_i^{(S)}}{\tau})}\right),\tag{1}$$

where q_i^T and q_i^S are q-values approximated by the teacher model and the student model, respectively. θ^S represents weights in the student model, and N is the number of actions. The softmax function is used to normalize the q-values that can be taken as a probability distribution or confidence of the actions. The temperature τ is used to adjust the degree of similarity of the distributions that is expected to be achieved. By the definition of KL divergence, the teacher's distribution is assumed to be the true distribution and used to supervise the student.

In the real-time distillation framework, both student and teacher models use the exact same architecture of the DQN. They both maintain two identical deep networks, a target network and a prediction network. Only the teacher interacts with the environment and determines the action to take, and the corresponding experiences are stored into a replay buffer. For each training iteration, the teacher and student models share the same data sampled from the replay buffer as a training set, and their weights are updated simultaneously and independently. The teacher's policy is optimized by minimizing the loss of the Bellman error [7], which we call the DQN loss:

$$\mathcal{L}_{DQN} = \mathbb{E}_{s,a}[\underbrace{r_{i+1} + \gamma \times \max_a Q(s_{i+1}, a, \theta_i^-)}_{target} - \underbrace{Q(s_i, a; \theta_i)}_{prediction}]^2.\tag{2}$$

In contrast, the student's loss is a combination of the DQN loss and the KL divergence:

$$\mathcal{L}_{student} = L_{KL} + L_{DQN}.\tag{3}$$

We name the DQN loss in the student's loss function as self-learning, as it encourages the student to find the most suitable way of shaping itself instead of strictly imitating the experts' behavior, as even a very small model still has some approximating ability. When distillation is not informative enough or lacks high-quality supervision, applying knowledge only via the KL divergence channel would lead to a less effective distillation. In image classification [4, 2], a combination of the student's own classification and the distillation loss provides a reference from different aspects. In our case, according to our observation, the student's weight distribution changes more drastically than that of the teacher during the training process. As the student adjusts itself to handle various states, self-learning is assumed to identify the right direction for updating the weights. Even when informative supervision is available, self-learning can still provide additional information, similar to self-adaptive adjustment or correctness.

2.1 Imitation via Target-term

The target term in the teacher's DQN loss is decoupled as

$$Y_i^{(T)} = r_{i+1} + \gamma \times \max_a Q^{(T)}(s_{i+1}, a; \theta_i^{(T),-}).\tag{4}$$

Here, maximization is used to determine the action as well as estimate the q-value using the same approximator, that is, the target network. Inspired by the theoretical analysis in Double-DQN [12, 11],

overestimation of the action value is more likely to occur in an insufficiently flexible approximation and may blow up via accumulation during training. In this prior work, double approximators were used to eliminate this penalty. Intuitively, the student's self-learning accumulates more errors than a teacher model during training, so it is necessary to incorporate double estimators to reduce it. Therefore, inviting the teacher's target network to be the second estimator in the student model that decides which action to take is a valid option. With this, the student's target term in its DQN loss is changed to be

$$Y_i^{(S)} \equiv r_{i+1} + \gamma \times Q^{(S)}(s_{i+1}, \underset{a}{argmax} Q^{(T)}(s_{i+1}, a; \theta_i^{(T),-}); \theta_i^{(S),-}). \tag{5}$$

Note that the *argmax* function extracts the action chosen by the teacher. In other words, the student uses weights $\theta_i^{(T),-}$ in the teacher's target network to determine the action and uses its own weights $\theta_i^{(S),-}$ to estimate the q-values. The student is required to imitate the teacher's choices in the target estimation, a method that we call *imitation*. Conversely, if the student takes the form of Equation 4, it is called *no-imitation*. Imitation via the target is a second channel of knowledge transference, beyond KL divergence.

2.2 Forward KL vs. Reverse KL

The uncertainty in action prediction is significant in some tasks, even when the agent has mastered some real skills. Consider the following extreme case, in which all actions are selected with equal probability both in the teacher and student models. The KL divergence tries to match the two random distributions. To deal with the uncertainty problem in image classification, Malinin and Gales [6] adopted reverse KL divergence (RKL) instead of forward KL divergence (FKL) when given highly uncertain data in classification. Back to reinforcement learning, the prediction uncertainty is generally high and causes more random action choices, which increases data uncertainty and confuses the student further. The reason is that the student learns from the teacher's demonstration too. RKL originates from the posterior belief [5] and can be decomposed to reverse cross-entropy (RCE) and self-entropy:

$$\mathcal{L}_{RKL}(\theta^{(S)}) = \underbrace{\sum_{i=1}^{N} -p_i^{(S)} ln(p_i^{(T)})}_{RCE} - \underbrace{\sum_{i=1}^{N} -p_i^{(S)} ln(p_i^{(S)})}_{Entropy}, \tag{6}$$

where p is the probability over actions, and N is the number of actions. If there is no distinction between q-values, all actions are approximately equiprobable and selected with a uniform probability of $1/N$. The gradient of RKL for a single logit is

$$\frac{\partial RKL}{\partial q_i^{(S)}} = \frac{1}{\tau}(\frac{1}{N} - \underbrace{(-p_i^{(S)} ln(p_i^{(S)}))}_{Entropy})). \tag{7}$$

According to Equation 7, the gradient update only depends on the changes in the student's entropy, ignoring the teacher's supervision. This mechanism can effectively prevent *harmful* knowledge transfer via distillation, because if the teacher's prediction ability degrades catastrophically and no constructive supervision is available, then the student is less affected by the teacher's predictions, thus stabilizing the student's learning. Theoretically [6], in contrast to forward KL divergence, which is *zero-avoiding* and drives the approximated distribution to be spread over each action, RKL is *zero forcing*, which allows students to allocate zero probability to other actions as long as the student is confident with the predicted best action, consequently reducing the effect of the teacher.

However, uncertainty does not always damage learning effectiveness. According to our empirical study, the games that are more aware of state values, such as Space Invaders, are less affected by the application of FKL or RKL. This means that any action may yield a similar reward, so there is little impact due to prediction uncertainty. For games like Ms. Pacman, where each action choice determines the outcome, policies are more effectively distilled via RKL if high uncertainty exists.

Table 1: Training results by applying the *imitation* method for different model sizes. Scores calculated by the percentage of teacher's scores (%). RTPD: Real-time Policy Distillation, PD: Policy Distillation [10]. Net1, Net2, and Net3 are the student models.

| | $Net1$ | | | $Net2$ | | | $Net3$ | | |
| | PD | $RTPD$ | | PD | $RTPD$ | | PD | $RTPD$ | |
	Max	Max	$Mean$	Max	Max	$Mean$	Max	Max	$Mean$
FKL									
Ms.Pacman	102.5	95.2	**112**	86.4	96.9	**92.3**	**96.2**	97.7	58.2
Qbert	129.6	80.8	**234**	130.6	76.5	**320.7**	**107.9**	55.5	101.9
Beamrider	**87.1**	47.4	35.8	**85.3**	62.6	37.	**75.2**	47.6	15.91
Pong	**100.6**	104.8	97.5	**103.7**	98	100.7	96.9	100.8	**99.1**
SpaceInvaders	78.6	101.9	**95.1**	49.4	100.6	**100.5**	20.9	94.2	**95.4**
Breakout	**105.6**	73.8	100.1	**98.1**	91.5	97.3	78.6	69.5	**91.1**
Enduro	**142.4**	103.9	116.4	**141.3**	106.5	105.1	117.1	109.2	**120.3**
RKL									
Ms.Pacman	102.5	130.5	**112.9**	86.4	120.3	**111.5**	96.2	124	**99.2**
Qbert	129.6	91.8	**306.7**	130.6	86.6	**229.3**	107.9	75.6	**233.7**
Beamrider	87.1	93	**108.6**	85.3	79	83.6	**75.2**	60.1	59.1
Pong	100.6	105.9	**113.3**	103.7	104.6	**114.4**	96.9	101.8	**100.5**
SpaceInvaders	78.6	108.9	**93.7**	49.4	105.3	**97.4**	20.9	102.6	**98.4**
Breakout	105.6	102.9	**106.9**	98.1	99.5	**101.3**	78.6	101.2	**107.7**
Enduro	**142.4**	99.9	104.0	**141.3**	101.4	104.3	**117.1**	100.6	103.6

3 Experiments and Results

To assess the performance of the proposed distillation approach, we selected seven games, one teacher model, Teacher=$\{32, 64, 64, 512\}$, and three student models [10, 13] with different capacities, Net1=$\{16, 32, 32, 256\}$, Net2=$\{16, 16, 16, 128\}$, and Net3=$\{16, 16, 16, 64\}$. The first three numbers represent the number of filters in the convolutional layers, and the last is the number of neurons in a fully connected layer. Students' corresponding compression ratios, in terms of the total number of parameters, are 25.2%, 6.7%, and 3.7%. To save training time, in each experiment, one teacher model and multiple student models with different sizes are constructed in the same graph, and all students share the same teacher. All training and evaluation settings are exactly the same as in DQN [7] except for the option of *null-op*. Ideally, after the teacher's convergence, additional iterations may help the student to consolidate accomplishments, but when training time is limited, we force all models to stop at the same epoch. Note that we have much fewer training epochs than does Policy Distillation (PD) [10], only as a rough reference, and the average training time is 200 hours for each experiment.

If perfect distillation performance of our algorithm is achieved, we expect the student's learning pace to keep up with the teacher's in real time and the student to outperform the teacher with respect to maximum and mean scores. All models are evaluated sequentially every 25,000 updates (called epochs) by playing up to 30 episodes. Limited by computation resources, we only consider 100 epochs. Besides comparing the student's maximum score over the whole training as a percentage of the teacher's maximum score, we also calculate the mean of all score percentages in the last 10 epochs.

In Table 1, we compare the mean percentages of our proposed method with Policy Distillation (PD) [10], as the mean value is more eligible to reflect the distillation performance. It is observed that the real-time architecture, combined with the imitation method, guarantees almost 100% distillation for most games, especially in terms of the mean value. Also, the distillation is less affected by the model capacity, and even the smallest model, Net3, is still able to achieve a performance comparable to

Table 2: Comparison of *imitation* and *no-imitation* with FKL (% of teacher).

| | Net4 | | | | Net5 | | | |
| | Imitation | | No-imitation | | Imitation | | No-imitation | |
	Max	Mean	Max	Mean	Max	Mean	Max	Mean
Ms.Pacman	**91.7**	**74.8**	81	63.9	**96**	**79.1**	84.4	73.7
Space	**99.8**	**116.4**	94.9	106.6	**94.2**	86	87.9	**89.3**
Breakout	**96.3**	102.2	81	**103**	**92.1**	**100.4**	87.5	100

Net1. We also conducted experiments applying RKL in contrast to FKL, which verifies the benefit of RKL.

In Table 2, two extra models, Net4=$\{8, 8, 16, 64\}$ and Net5=$\{8, 16, 8, 64\}$ were added to assess the advantage of imitation in very small models. The compression ratios are 3.2% and 1.7%, respectively. The result shows the imitation method truly helps improve the overall distillation quality for the three games.

4   Conclusion

Our real-time distillation architecture reduces the distillation time significantly and transfers knowledge to extremely small models without much degradation in performance. The investigation of the imitation method and two forms of KL divergence may contribute some insights into the improvement of distillation quality and other reinforcement learning tasks, such as robot control and planning.

References

[1] Sungsoo Ahn, Shell Xu Hu, Andreas Damianou, Neil D Lawrence, and Zhenwen Dai. Variational information distillation for knowledge transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9163–9171, 2019.

[2] Micah Goldblum, Liam Fowl, Soheil Feizi, and Tom Goldstein. Adversarially robust distillation. *arXiv:1905.09747*, 2019.

[3] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv:1503.02531*, 2015.

[4] Xiao Jin, Baoyun Peng, Yichao Wu, Yu Liu, Jiaheng Liu, Ding Liang, Junjie Yan, and Xiaolin Hu. Knowledge distillation via route constrained optimization. *arXiv:1904.09149*, 2019.

[5] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv:1312.6114*, 2013.

[6] Andrey Malinin and Mark Gales. Reverse kl-divergence training of prior networks: Improved uncertainty and adversarial robustness. *arXiv:1905.13472*, 2019.

[7] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[8] Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. Actor-mimic: Deep multitask and transfer reinforcement learning. *arXiv:1511.06342*, 2015.

[9] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *arXiv:1412.6550*, 2014.

[10] Andrei A Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. *arXiv:1511.06295*, 2015.

[11] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[12] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI)*, pages 2094–2100, 2016.

[13] Haiyan Yin and Sinno Jialin Pan. Knowledge transfer for deep reinforcement learning with hierarchical experience replay. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI)*, pages 1640–1646, 2017.