Comprehensive and Efficient Data Labeling via Adaptive Model Scheduling

Mu Yuan, Lan Zhang , Xiang-Yang Li
University of Science and Technology of China
Hefei, China
ym0813@mail.ustc.edu.cn, zhanglan@ustc.edu.cn, xiangyangli@ustc.edu.cn

Hui Xiong
Rutgers University
Newward, USA
xionghui@gmail.com

***Abstract*—Labeling data comprehensively and efficiently is a widely needed but challenging task. With limited computing resources, given a data stream and a collection of deep-learning models, we propose to adaptively select and schedule a subset of these models to execute, aiming to maximize the value of the model output. Achieving this goal is nontrivial since a model's output on any data item is content-dependent and hard to predict. In this paper, we present an *Adaptive Model Scheduling* framework, consisting of 1) a deep reinforcement learning-based approach to predict the value of unexecuted models by mining semantic relationship among diverse models, and 2) two heuristic algorithms to adaptively schedule models under deadline or deadline-memory constraints. The proposed framework does not require any prior knowledge of the data, which works as a powerful complement to existing model optimization technologies. We conduct extensive evaluations on 30 popular image labeling models to demonstrate the effectiveness of our design.**

***Index Terms*—multi-model inference, data labeling, adaptive model scheduling, deep reinforcement learning**

I. Introduction

With the explosive growth of data volume and the rapid development of the AI industry, it is an appealing task to comprehensively label large amounts of data as fast as possible. For example, annotating each image with a collection of semantic labels can power a wide variety of functionalities, such as multi-label image retrieval and image classification. On data trading platforms [4], the richer the label of a data set, the higher the price of the data set. There are two main streams of previous work towards this objective. One stream of efforts is enhancing the capability of a single model. Multi-label learning and multi-task learning [3], [5] have been proposed to enable a single model to extract more complex semantic. The other stream focuses on accelerating the model execution by designing a variety of methods, including model compression via parameter pruning & sharing [1] and network architecture optimization [9].

In this paper we propose an adaptive model scheduling framework (see Fig. 1), consisting of 1) a novel deep reinforcement learning (DRL) based method to model the semantic relationship among models and predict the output of unexecuted models based on outputs of executed ones, and 2) two heuristic scheduling algorithms to maximize the value of output labels under a deadline or deadline-memory constraints. Our adaptive model scheduling framework gathers the power of existing models to achieve comprehensive and efficient labeling of large-scale data. It does not require users to have deep learning related expertise or know the content of the raw data in advance. To the best of our knowledge, this work is the first step to explore the relationship among multiple models. We conduct extensive evaluations on large-scale and highly diverse images from five public datasets using 30 popular deep learning models for 10 visual analysis tasks. Our experimental results show that the proposed adaptive model scheduling framework could save 53.1% execution time when we need a 100% recall of all valuable labels. We could save about 70.0% execution time when we only need 80% recall of all valuable labels. Given the 0.5s delay budget for each image, our proposed algorithms could improve the obtained output value by 132-310% compared with the randomly scheduling of models.

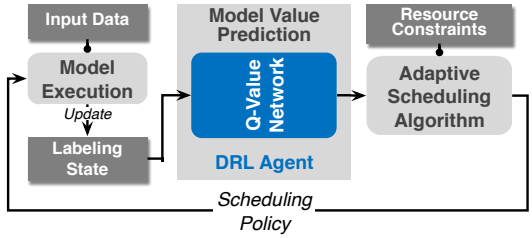


Fig. 1. Overview of our proposed *adaptive model scheduling* framework.

II. Model Value Prediction

The first key component of our proposed adaptive model scheduling framework is to predict the output values of unexecuted models based on the outputs of executed ones. The prediction accuracy is critical to the subsequent scheduling algorithms. To confirm the interactive characteristic of the proposed framework, we propose a DRL based method for the mode value prediction task. To our best knowledge, this work is the first step to explore the DRL for mining semantic relationships among labeling capacities of multiple models. There are three main parts of a DRL method, including the environment observation, the action space, and the reward feedback. In our problem, we model the *labeling state* as the environment observation, which is a n-dimension binary vector (n is the number of supported labels). The i-th binary value 1 or 0 indicates the state that the i-th label has been or has not been output by executed models. In every iteration, after

Lan Zhang is the corresponding author.

This paper has been accepted for publication at ICDE'20. This version is a preprint identical in content to the version to be published.

the DRL agent selects an action/model, the system executes the model on the input data. According to the execution result $O(m, d)$, the labeling state is updated and the agent receives a reward.

A. Reward Function

The confidence $l_i.conf$ represents the model's judgment of the accuracy of label l_i, which can be adopted as the profit. However, the number of output labels influences the reward, which varies among different deep learning models. We use a logarithmic function to mitigate the bias caused by different numbers of models' output labels. On the other hand, considering the different requirements of diverse applications for model priorities, we introduce a parameter θ_m as the user-defined priority for a model m. The greater θ_m is, the higher priority the model m has. Taking all the aforementioned factors into consideration, the reward function is defined as:

$$r(m,d) = \begin{cases} \ln(\theta_m \sum_{l_i \in O'(\{m\},d)} l_i.conf + 1), & O'(m,d) \neq \emptyset \\ -1, & O'(\{m\},d) = \emptyset \end{cases} \tag{1}$$

We define $O'(m, d)$ as the set of new labels output by a running model m, which have not been output by other models yet. Since the supported labels of different models may overlap, $O'(\{m\}, d) \subseteq O(\{m\}, d)$. To avoid selecting duplicated or valueless models, when $O'(\{m\}, d) = \emptyset$, the agent receives a punishment -1 as the feedback value.

B. Agent Learning

The DRL agent needs to learn the mapping from the environment observation to the action space, which is the mapping from the current labeling state to the selection of the model to be executed. Facing the complexity of the problem, we adopt a deep Q-value network (DQN) [6] as the observation-action mapping function. The DQN architecture can be adjusted to adapt to different sizes of observation and action spaces. In our implementation, a hidden dense layer with 256 neurons activated by ReLu is used to cope with a 1104-dimension observation space and a 30-dimension action space. For training the agent, we implement four popular DRL approaches: Original DQN [6], Double DQN [7], Dueling DQN [8] and Deep SARSA [2]. Theoretically, the proposed DRL-based agent can be trained by any Q-value network-based DRL approach. The classic epsilon-greedy policy is applied in the training process, which either selects the action with the maximal Q value or randomly selects an action with the probability epsilon. In this way, however, the training process is hard to reach convergence. The reason is that, for each input, after some iterations, the agent will reach the state that all valuable labels have already been recalled. So according to the reward function Eq. 1, any further action will bring back a punishment, which obstructs the agent against improving subsequent actions. To fix this problem, we add an **"END"** action, whose reward is zero, to indicate the end of the selection process for the current input. Then the agent has the option to select the "END" action to avoid further punishment when no valuable model is left. Results of experiments on agent learning confirm that the "END" action effectively quickens the velocity of convergence.

Algorithm 1 Scheduling under deadline constraint.

Input: model set M, time budget B_{time}, DRL agent Q
Output: model subset S

```
S ← ∅
while B_time > 0 do
    Filter out models that m.time > B_time
    m* ← arg max_{m∈M\S} Q(m,d)/m.time
    S ← S ∪ {m*}, B_time ← B_time − m*.time
end while
return S
```

III. Adaptive Model Scheduling

The second key component of our framework is adaptive scheduling algorithms under various computing resources constraints. Given a set of available deep learning models and input data, the trained DRL agent provides the value prediction for unexecuted models based on the current labeling state. Based on the predicted values and remaining resources, a scheduling algorithm determines the execution policy that aims to maximize the evaluation function $f(S, d)$. In this work, we consider the two most common constraints for data labeling tasks: deadline and limited memory.

A. Deadline Constraint

In a single processor case, models can only be executed serially. The deadline constraint is set for each input data, which is a common requirement of delay-sensitive systems. With the deadline constraint, our optimization problem is specified as follows:

$$\max_{S \subseteq M} f(S,d), \text{ s.t. } \sum_{m \in S} m.time \leq B_{time} \tag{2}$$

, where $m.time$ is the execution time of model m and B_{time} is the constraint of the total execution time for all selected models in S. The most relevant problem is the submodular function maximization with a knapsack constraint, which is NP-hard. A commonly used heuristic approach to solve the knapsack problem is the cost-profit greedy algorithm, which selects the model m that maximizes $\frac{f(S \cup \{m\},d) - f(S,d)}{m.time}$ at each iteration. In this work, we propose to adopt the cost-profit greedy algorithm by using the Q value of each unexecuted model as its estimated profit at each iteration.

B. Deadline-Memory Constraint

In a multi-processor case, multiple deep learning models can be executed in parallel on a shared-memory computing platform. A two-dimension constraint, a deadline and memory constraint, is considered for each input data. The tangible formulation of this optimization problem is:

$$\begin{aligned} &\max_{S \subseteq M} f(S,d), \text{ where } S = \bigcup_{i=1}^{N} S_i \\ &\text{s.t. } \sum_{i=1}^{N} S_i.time \leq B_{time}, \sum_{m \in S_i} m.mem \leq B_{mem} \end{aligned} \tag{3}$$

Algorithm 2 Scheduling under deadline-memory constraints.

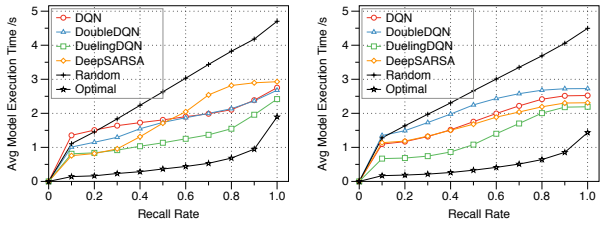
Input: model set M, time budget B_{time}, memory budget B_{mem}, DRL agent Q
Output: model scheduling policy S

$S \leftarrow [\,]$, $TimeCost \leftarrow 0$, $S_t \leftarrow \emptyset$
while $TimeCost < B_{time}$ **do**
  Filter out models that $m.mem > B_{mem}$
  $m_1^* \leftarrow \arg\max_{m \in M \setminus S} \frac{Q(m,d)}{m.time \times m.mem}$
  $S_t \leftarrow S_t \cup \{m_1^*\}$, $B_{time}^t \leftarrow TimeCost + m_1^*.time$
  Filter out models by temporary deadline B_{time}^t
  while $B_{mem} > 0$ **do**
    $m_2^* \leftarrow \arg\max_{m \in M \setminus S} \frac{Q(m,d)}{m.mem}$
    $S_t \leftarrow S_t \cup \{m_2^*\}$, $B_{mem} \leftarrow B_{mem} - m_2^*.mem$
  end while
  $S.append(S_t)$, Wait until model $m_3^* \in S_t$ finishes
  $B_{mem} \leftarrow B_{mem} + m_3^*.mem$, $S_t \leftarrow S_t \setminus \{m_3^*\}$
end while
return S

Given an input, let the model scheduling process have N iterations in total. S_i is the set of models being executed at the i-th iteration. Let $S_i.time$ denote the running time of the i-th iteration and B_{time} denote the acceptable total execution time for all selected models S. The memory cost of a model m is denoted as $m.mem$, which is measured by the peak memory usage in our experiments. At any time, the total memory usage of running models should not exceed the memory budget B_{mem}. In this work, we design an efficient heuristic algorithm 2. In each iteration, the algorithm first greedily selects one model that provides the highest value per unit resource area (the area here is the product of normalized time cost and memory cost) and sets the end time of this model as a temporary deadline. Subject to the temporary deadline, the algorithm repeatedly selects the model with the highest value-memory ratio until the memory budget is reached. The intermediate model value is predicted by the pre-trained DRL agent. Once a model completes its execution, releasing its occupied memory and a new iteration begins.

C. Performance Analysis

To measure the performance of our heuristic algorithms, we need the optimal solution. Under certain deadline and deadline-memory constraints, the NP-hard property requires to search $O(|M|!)$ policies to find the optimal one. In our implementation, it is infeasible to enumerate all practicable execution policies of 30 models. Therefore, we relax the constraint that when the remaining resources are not enough for one model to complete its execution, the model can still be selected and contribute label value partially. We refer to the optimal policy of this relaxed problem as the *optimal* policy*, which greedily selects the model with maximal $\frac{f(S\cup\{m\},d)-f(S,d)}{m.time}$ within the deadline constraint or the model with maximal $\frac{f(S\cup\{m\},d)-f(S,d)}{m.time*m.mem}$ within the deadline-memory constraint. The *optimal** policy provides a performance upper bound for the exact optimal policy.



(a) MSCOCO 2017 **(b) MirFlickr25**

Fig. 2. The average time cost of executed models per image vs. required recall rate of output value.

IV. Evaluation

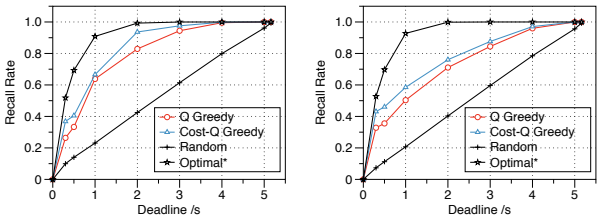
We implemented the proposed adaptive model scheduling framework and conducted extensive evaluations for large-scale comprehensive image labeling tasks. We consider 10 visual analysis tasks and deployed and trained a total of 30 popular deep learning models. These models can label images with a wide range of semantic information (1104 different labels in all). Table IV summaries the deployed models and their supported labels. For each model, the time cost ($m.time$) is set as the average value while the GPU memory cost ($m.mem$) is set as the peak value. We conducted experiments on two public image datasets: MSCOCO 2017 and MirFlickr25. To train the DRL agents and measure the effectiveness of our model scheduling system, it is necessary to obtain the ground truth of each model's performance. We executed all 30 models on two datasets and stored the output labels and confidences. For each dataset, we split it into a training set and a testing set with the ratio of 1:4. For all evaluations, we employed a server with 48 Intel Xeon CPU E5-2650 v4 cores and one Tesla P100 GPU. Due to the page limit we only demonstrate part of experiments. For the full version, please refer to [10].

Task Name	**Label#**
Object Detection	80
Place Classification	365
Face Detection	1
Face Landmark Localization	70
Pose Estimation	17
Emotion Classification	7
Gender Classification	2
Action Classification	400
Hand Landmark Localization	42
Dog Classification	120
10 Tasks	**1104 Labels**

TABLE I
SUMMARY OF 10 VISUAL ANALYSIS TASKS.

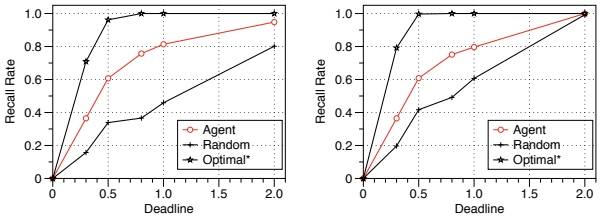
A. RL-Based Model Value Prediction

We implemented our designed DRL agent to predict model value before the execution. We trained the DRL agent using four methods with the identical Q-value network as introduced in Section II-B, including DQN [6], DoubleDQN [7], DuelingDQN [8] and DeepSARSA [2]. We trained and tested these four DRL agents on two datasets, MSCOCO 2017, MirFlickr25 separately. We use the Q-value greedy policy that greedily selects the model with maximal Q value as the next one to be executed until the recall rate of true output



(a) MSCOCO 2017 **(b) MirFlickr25**

Fig. 3. Value recall rate under deadline constraints.



(a) 8GB Memory **(b) 12GB Memory**

Fig. 4. Value recall rate under memory-deadline constraints.

value reaches a given threshold [1]. Through these experiments, we compare the performance of DRL agents trained by four schemas. To quantify the effectiveness of our DRL-based model value prediction method, we also implemented the following random policy and optimal policy as a comparison. Fig. 2 shows the experimental results of different policies on two datasets, using the average execution time as the metric respectively. There is a significant gap between the random policy and the optimal policy. By predicting the model value, all four DRL agents outperform the random policy and effectively improve the model selection. Among the four DRL agents, the one trained by DuelingDQN performs best, which saves 48.6-51.2% model execution time when the required recall rate is 1.0, compared with the random policy.

B. Scheduling under Deadline Constraint

We consider the most common constraint, the deadline of each input data, of data labeling tasks, and evaluate the proposed scheduling Algorithm 1 (referred to as *Cost-Q Greedy policy*). Using the output value recall rate under deadline constraint as the metric, three policies are implemented for comparison: random policy, optimal* policy (§III-C) and Q-greedy policy. As a representative, DRL agents in the following experiments are trained with DuelingDQN schema. Fig. 3 shows that Algorithm 1 outperforms the Q-greedy policy. Algorithm 1 boosts the value recall rate by 188.7-309.5% with a 0.5s delay budget, compared with the random policies.

C. Scheduling under Memory-Deadline Constraints

To tackle the more challenging problem, scheduling models under two-dimension knapsack constraints, we propose Algorithm 2. Deadline and GPU memory constraints are commonly limited computing resources which are *orthogonal*: GPU memory restricts the *spatial* size of the parallel deep learning models, while deadline limits the overall running time in *temporal* dimension.Identically, we use the output value recall rate as the metric and random policy and optimal* policy are used as baselines. As a representative, we use the results of DuelingDQN on MSCOCO 2017 dataset. As shown in Fig. 4, Algorithm 2 significantly improves the output value recall rate compared with random policies. More specifically, the recall rate of output value is improved by 106.9% / 52.8% under 8GB / 12GB GPU memory and 0.8s deadline constraints.

[1]The stop condition is determined by the ground truth

V. Conclusion

In this work, we tackled a challenging task, adaptive model scheduling, which works as an effective approach towards comprehensive and efficient data labeling. We designed a framework, including a novel method to predict unexecuted models' value and adaptive scheduling algorithms to improve the aggregated values of executed models for each data item. Our extensive evaluations demonstrate that our design achieves significant performance improvement.

VI. Acknowledgements

The research is partially supported by National Key R&D Program of China 2018YFB0803400, NSF China under Grants No. 61822209, China National Funds for Distinguished Young Scientists with No.61625205, NSF China under Grants No.61751211, 61932016, 61520106007, the Fundamental Research Funds for the Central Universities and Key Research Program of Frontier Sciences, CAS. No.QYZDY-SSW-JSC002.

References

[1] W. Chen, J. Wilson, S. Tyree, K. Q. Weinberger, and Y. Chen. Compressing convolutional neural networks in the frequency domain. In Proceedings of the 22nd ACM SIGKDD, pages 1475–1484. ACM, 2016.

[2] M. Corazza and A. Sangalli. Q-learning and sarsa: a comparison between two intelligent stochastic control approaches for financial trading. University Ca'Foscari of Venice, Dept. of Economics Research Paper Series No, 15, 2015.

[3] L. Kaiser, A. N. Gomez, N. Shazeer, A. Vaswani, N. Parmar, L. Jones, and J. Uszkoreit. One model to learn them all. arXiv preprint arXiv:1706.05137, 2017.

[4] X.-Y. Li, J. Qian, and X. Wang. Can china lead the development of data trading and sharing markets? Communications of the ACM, 61(11):50–51, 2018.

[5] J. Ma, Z. Zhao, X. Yi, J. Chen, L. Hong, and E. H. Chi. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In Proceedings of the 24th ACM SIGKDD, pages 1930–1939. ACM, 2018.

[6] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602, 2013.

[7] H. Van Hasselt, A. Guez, and D. Silver. Deep reinforcement learning with double q-learning. In Thirtieth AAAI, 2016.

[8] Z. Wang, T. Schaul, M. Hessel, H. Van Hasselt, M. Lanctot, and N. De Freitas. Dueling network architectures for deep reinforcement learning. arXiv preprint arXiv:1511.06581, 2015.

[9] B. Wu, F. N. Iandola, P. H. Jin, and K. Keutzer. Squeezedet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving. In CVPR Workshops, pages 446–454, 2017.

[10] M. Yuan, L. Zhang, X.-Y. Li, and H. Xiong. Comprehensive and efficient data labeling via adaptive model scheduling. arXiv preprint arXiv:2002.05520, 2020.