Adaptive Model Scheduling for Resource-efficient Data Labeling*

MU YUAN, University of Science and Technology of China, China
LAN ZHANG†, University of Science and Technology of China, China
XIANG-YANG LI, University of Science and Technology of China, China
LIN-ZHUO YANG, University of Science and Technology of China, China
HUI XIONG, Rutgers University, USA

Labeling data (e.g., labeling the people, objects, actions and scene in images) comprehensively and efficiently is a widely needed but challenging task. Numerous models were proposed to label various data and many approaches were designed to enhance the ability of deep learning models or accelerate them. Unfortunately, a single machine-learning model is not powerful enough to extract various semantic information from data. Given certain applications, such as image retrieval platforms and photo album management apps, it is often required to execute a collection of models to obtain sufficient labels. With limited computing resources and stringent delay, given a data stream and a collection of applicable resource-hungry deep-learning models, we design a novel approach to adaptively schedule a subset of these models to execute on each data item, aiming to maximize the value of the model output (e.g., the number of high-confidence labels). Achieving this lofty goal is nontrivial since a model's output on any data item is content-dependent and unknown until we execute it. To tackle this, we propose 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 the model execution order under a deadline or deadline-memory constraints respectively. The proposed framework doesn't require any prior knowledge of the data, which works as a powerful complement to existing model optimization technologies. We conduct extensive evaluations on five diverse image datasets and 30 popular image labeling models to demonstrate the effectiveness of our design: our design could save around 53% execution time without loss of any valuable labels.

CCS Concepts: • **Computing methodologies** → **Reinforcement learning**; • **Theory of computation** → **Scheduling algorithms**.

Additional Key Words and Phrases: model scheduling, reinforcement learning, data labeling

*This article is a substantially extended and revised version of Yuan et al. [34], which appeared in the proceedings of the 2020 IEEE 36th International Conference on Data Engineering (ICDE).

†Lan Zhang is the corresponding author.

This paper has been accepted for publication at TKDD'22. This version is a preprint identical in content to the version to be published.

Authors' addresses: Mu Yuan, University of Science and Technology of China, Hefei, China, ym0813@mail.ustc.edu.cn; Lan Zhang, School of Computer Science and Technology, School of Data Science, University of Science and Technology of China, Hefei, China, zhanglan@ustc.edu.cn; Xiang-Yang Li, University of Science and Technology of China, Hefei, China, xiangyangli@ustc.edu.cn; Lin-Zhuo Yang, University of Science and Technology of China, Hefei, China, ustcylz@mail.ustc.edu.cn; Hui Xiong, Rutgers University, Newark, USA, hxiong@rutgers.edu.

ACM Reference Format:
Mu Yuan, Lan Zhang, Xiang-Yang Li, Lin-Zhuo Yang, and Hui Xiong. 2022. Adaptive Model Scheduling for Resource-efficient Data Labeling. *ACM Trans. Knowl. Discov. Data.* 1, 1 (March 2022), 22 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 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 comprehensive collection of semantic labels can power a wide variety of functionalities, such as multi-label image retrieval and image classification [17, 36, 37], and provide more possible avenues of making the most of images. Image retrieval platforms usually employ a series of machine learning models to provide as many labels as possible to describe each image to improve the quality of search results. By identifying faces, landmarks, scenes, objects, emotions and events in photos, many smartphones support thousands of searchable keywords to find photos in one's album. On data trading platforms [18], the richer the label of a data set, the higher the price of the data set. To facilitate the extraction of rich labels, there are two main streams of previous work. One stream of efforts have been devoted to enhancing a single model's capability. Multi-label learning [9, 32] and multi-task learning [13, 20] have been proposed to enable a single model to extract more complex semantics of input data. The other stream of work focus on accelerating the model execution by designing a variety of methods, including model compression via parameter pruning & sharing [4], network architecture optimization [29] and adaptive model configuration [12].

Despite the efforts made by existing work to provide us with increasingly powerful and light-weight deep learning models for various tasks, there still exist several major limitations hindering comprehensive and efficient data labeling for many applications. (1) The limited ability of a single model: one single model can usually only capture features of certain aspects of the data. Thus, in many cases, e.g., image retrieval, image analysis for video surveillance, and data trading, it is inevitable to execute a series of deep learning models to achieve a broad understanding of data. Usually, for these tasks, the more diverse models utilized, the better the services provided. (2) Limited computing resources: one simple policy is to execute all possible applicable deep learning models on each piece of input data. Unfortunately, this is infeasible in many cases due to the limited computing resources and the intolerable long delay. For example, an advanced human pose estimation model [2] consumes more than 10GB GPU memory. (3) Limited expertise of users: facing a large number of available deep learning models and massive raw data, it is quite difficult, if not impossible, for a user to select an optimal subset of models for achieving a comprehensive labeling as well as a low computing overhead. To tailor different deep learning models for different input data is also a very difficult and time-consuming task. (4) Diverse and unknown content of raw data: even if an oracle can select a series of models suitable for the upcoming data, executing all those models for each piece of input data can still result in serious computing waste. As the example shown in Fig. 1, though every model outputs some valuable labels for some input images, 16/30 model executions didn't generate anything useful. It is nontrivial to avoid such waste since the content of the data is unknown before model executions. As a result, given a set of models, how to extract the maximum value (i.e., as many high-confidence labels as possible) from large-scale data at a minimum cost remains a very challenging problem.

The blue boxes in Fig. 1 show the significance of executing multiple diverse models to achieve a broad understanding of data, while the white and grey boxes reveal that a large portion of computing resources could be wasted without careful model selection. Towards comprehensive and efficient data labeling, in this work we propose a novel *adaptive model scheduling* framework,

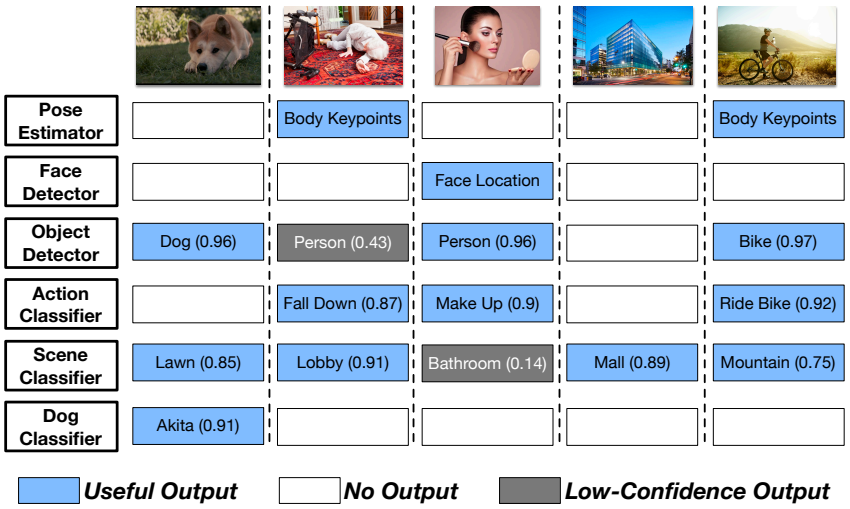


Fig. 1. Output of six models executed on the same set of images from a public dataset.

which works orthogonally to model enhancing and accelerating methods. Specifically, given a set of deep learning models and a stream of raw data to be processed, we propose to adaptively schedule model execution on each piece of input data to maximize the value of extracted labels under some constraints on computing resources and delay. This ambitious goal, however, is nontrivial to achieve due to the following reasons. **First**, an optimal scheduling policy should only select models that will output valuable labels, i.e., labels with high confidence, for each piece of data. A model's output, however, highly depends on the content of the data, which is generally unknown in advance and hard to predict. As the example in Fig. 1, the ideal policy should only conduct those blue-box model executions. Without knowing if there is a human body in an image, however, it is difficult to decide whether or not to execute the computation-intensive pose estimator. **Second**, for each piece of raw data, even if we can somehow obtain the subset of models that will generate valuable labels, it is still theoretically NP-hard to schedule them to maximize the value of outputs under multi-dimensional constraints. The most relevant well-defined problem is the submodular function maximization with multi-dimensional knapsack constraintsïijŇ however, different from which the values of model outputs are unknown in our model scheduling problem. Even when all output values are given in advance, which is the simplified version of our case, the problem remains NP-hard.

Given a stream of data, a simple case is that the data could be partitioned into chunks (such as segments of video), and each chunk has a certain correlation in its content. For such kind of data, our extensive evaluations confirmed that a simple exploration-exploitation solution works extremely well: at the beginning of the chunk, we explore almost all possible models and find the best subset of models for this chunk, then for the remaining of this chunk, we run this subset of models, i.e., exploitation. The challenging part is the other case when the data items are not correlated, e.g., a set of randomly taken images. To tackle such a question, we explore the implicit semantic relationship among several deep learning models, which is inspired by the observation that some executed models' outputs provide hints of other models' outputs. For example, if a pose estimator fails to detect any body keypoint, then a face detector is very likely to output nothing and should not be executed. A straightforward solution is to manually design such execution rules to characterize the dependency among different output labels. Our experimental results show that

handcrafted rules are cumbersome and inefficient when there are tens of models, which support more than one thousand labels in our experiments.

Contributions: In this work, we propose an adaptive model scheduling framework, 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.

2 RELATED WORK

Towards comprehensive and efficient data labeling, previous work can be divided into two main streams: enhancing one single model's ability and accelerating the model execution.

2.1 Model Ability Enhancement

Multi-label learning. Many industrial applications, including document categorization and automatic image annotation, require to label the raw data with multiple tags. Multi-label learning is proposed to enable models to output a set of labels for one input data. The extreme output dimension and complex structures are the main challenges of multi-label learning tasks [31]. Han et al. [9] proposed a collaborative embedding approach, which exploits the association between embedding features and annotations. Yang et al. [32] designed a NNs which can learn to predict labels and exploit correlation among them simultaneously.

Multi-task learning. Another line of work studies multi-task learning (MTL) [23] that aims to leverage implicit relatedness among different tasks to improve the performance of all of them. Hard and soft parameter sharing are the most commonly used technologies in MTL. Kaiser et al. [13] proposed a deep learning model that can learn multiple tasks on different data types, including image captioning (image data), speech recognition (audio data) and English parsing (text data). Recent work [20] designed a method to automatically learn to model the task relationships directly from data, without human expertise.

Multi-label learning and multi-task learning exploit the correlation among output labels and related tasks and successfully enhance a single model's ability. However, the increasing complexity of industrial tasks, where the more extracted information the better, results in the insufficiency of a single model. As an effective complement, our proposed adaptive model scheduling framework leverages the power of multiple models with minimum computation waste.

2.2 Model Execution Acceleration

Model compression. Existing literature provides many technologies for deep learning model compression. Parameter pruning and sharing [4] is one popular technology to reduce redundant parameters and operations in deep neural networks. Another line of work [29] focuses on designing efficient convolutional filters, which is crucial to the execution speed of visual analysis models.

Adaptive model configuration. Except for improving the computing efficiency, one recent work [12] found optimization potential from the adaptive model configuration. They consider configurable variables like the video frame rate, resolution, and neural networks architecture and proposed a framework to adaptively select configurations to accelerate a video analyzing system.

Different from directly accelerating model execution, our proposed adaptive model scheduling avoids executing unnecessary models by learning implicit relationships among different models. We believe our work is a powerful complement to the existing optimization technologies.

3 DATA-DRIVEN ANALYSIS

This work is motivated by increasing demands for comprehensive and efficient data annotation. The main bottlenecks of this task are the limited ability of a single model and limited computing resources. Instead of enhancing the ability of every single model or accelerating its execution, we propose to select the optimal combination of off-the-shelf models to be executed for each piece of raw data, which is to schedule the execution of models adaptively. To demonstrate the necessity and potential of this work, we conduct a data-driven analysis of comprehensive image labeling for applications like image retrieval and photo management. We collected 394,170 images from three well-known public datasets (MSCOCO 2017 [19], Places365 [38] and MirFlickr25 [11]). MSCOCO dataset is collected for the object detection task; Place365 dataset is collected for the place classification task; MirFlickr25 consists of images from the social photography site Flickr. We believe the content of these three datasets is diverse enough to represent most real-life cases. We deployed 30 deep learning models for 10 different visual analyzing tasks, such as objection detection, face detection, action classification, and place classification. In all, these models extract 1104 labels. See Section 7.1 and Table 1 for more details for datasets and models.

We executed all 30 models on every image and collected the output results, consisting of labels and their corresponding confidences. We refer to such an "executing all models on each image" procedure as "no policy". We analyze the drawbacks of the naive "no policy" model execution. Using 4 Tesla P100 GPU, it took 5.16s on average to process one image on a GPU card and about **6 days** to analyze all 394,170 images. The cost is unacceptable for many delay-sensitive tasks and will increase significantly as the number of images/models grows. Fortunately, not all computing costs are necessary. Fig. 1 reveals that there is a significant waste of computing resources for the "no policy" model execution since a large portion of model executions only generate low confidence outputs or even nothing. To further quantify the waste, we simulate a process of data labeling with an ideal "optimal policy". Based on the complete execution results we obtained, the "optimal policy" only selects the model executions which generate high-confidence outputs. As shown in Fig. 2, the "optimal policy" requires 1.14s to process one image on average, which costs only 22.1% time of the "no policy". It speeds up the total labeling process from 6 days to about 1.3 days.

Our data-driven analysis shows that leveraging a set of diverse models can greatly enrich the data annotation without requiring any preknowledge of the raw data or expertise of deep learning models. An optimal model execution policy can save up to **77.9%** computing cost without the loss of any valuable labels. The significant benefits and huge improvement potential strongly motivate us to search for the optimal policy. To design an optimal model scheduling policy, however, is nontrivial due to the reason that it is very challenging to estimate a model's output for a piece of raw data before the actual execution. Here, we simulated a "random policy" using the complete outputs of "no policy", which executes models in a random order for each data until all valuable labels for this data have been recalled. Fig. 2 shows that the "random policy" does not help much, which only saves 0.52s per image on average. In the CDF of time cost per image, we can see a huge gap between the "optimal policy" and the "random policy". It reveals that an effective model scheduling policy requires a much more insightful design.

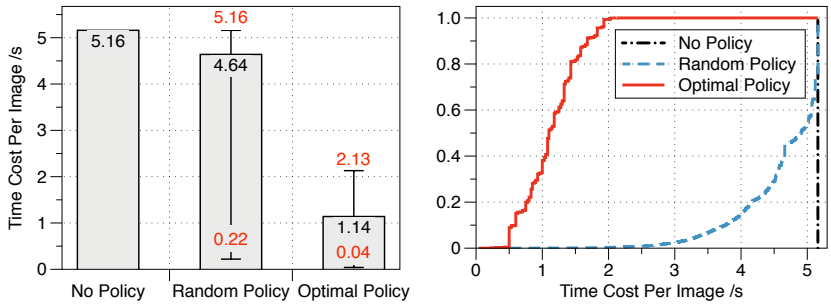


Fig. 2. Time cost of three policies to obtain all valuable labels for each image (Left: average time cost per image; Right: CDF of the time cost per image).

4 PROBLEM & DESIGN OVERVIEW

In this work, we aim to design such an optimal policy that adaptively schedules the execution of a set of off-the-shelf deep learning models for each piece of input data to maximize the value of outputs under computing resources and delay constraints. For the policy to be widely applicable, it doesn't need any modification of the existing deep learning models, nor knowledge of the input data content in advance. In this section, we first present the formal definition of the problem and then introduce the overview of our design.

4.1 Problem Definition

Given a set of available models M, let $L(m)$ denote the supported labels for $\forall m \in M$. Then the whole set of labels is $L(M) = \bigcup_{m \in M} L(m)$. Each label $l_i \in L(M)$ has a profit $p_i > 0$, which indicates the value of the label l_i to the user. For an input data d, let $O(S, d)$ denote the output of a subset of models $S \subseteq M$ executed on the data d, which is a set of labels. We evaluate the output by the following function:

$$f(S, d) = \sum_{l_i \in O(S,d)} p_i \tag{1}$$

The objective is to select such a model subset S that maximizes the evaluation function $f(S, d)$, while the computing costs of S are under some constraints, which can be formalized as follows:

$$\begin{aligned} &\max_{S \subseteq M} f(S, d) \\ &\text{s.t. constraints on } S \end{aligned} \tag{2}$$

Some common constraints include time delay caused by executing S and GPU memory occupied by S.

Lemma 1. *The evaluation function f is a non-negative, non-decreasing submodular function.*

Proof. 1) Non-negative. $\forall S \subseteq M$, $f(S, d) = \sum_{l_i \in O(S,d)} p_i$. Since $p_i > 0$, we have $f(S, d) \geq 0$.

2) Non-decreasing. For $\forall m \in M \setminus S$,

$$\begin{aligned} &f(S \cup \{m\}, d) - f(S, d) \\ &= \sum_{l_i \in O(S \cup \{m\}, d) \setminus O(S,d)} p_i \\ &\geq 0. \end{aligned}$$

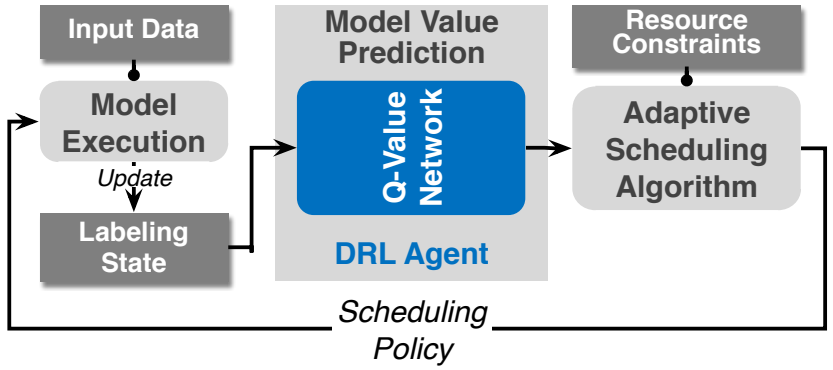


Fig. 3. Overview of our proposed *adaptive model scheduling* framework.

3) Submodular. For $\forall S \subseteq T \subseteq M$, obviously we have $O(S,d) \subseteq O(T,d)$. So, for $\forall m \in M \setminus T$, $O(T \cup \{m\},d) \setminus O(T,d) \subseteq O(S \cup \{m\},d) \setminus O(S,d)$). Then

$$\sum_{l_i \in O(T \cup \{m\},d) \setminus O(T,d)} p_i \leq \sum_{l_j \in O(S \cup \{m\},d) \setminus O(S,d)} p_j.$$

According to the definition of evaluation function in Eq.(1), we have $f(T \cup \{m\}) - f(T) \leq f(S \cup \{m\}) - f(S)$. So, the evaluation function is submodular. □

Note that, if the supported labels of different models do not overlap, the evaluation function becomes a modular function, i.e. the inequality characterizing submodularity holds with equality. Traditional submodular function optimization problems commonly assume that the cost of the evaluation function itself is negligible, e.g., the weighted coverage function. In our problem, however, to calculate the evaluation function f requires actual execution results of computation-intensive deep learning models. But once we have executed those models, there is no need for scheduling them anymore. This dilemma makes classic submodular function optimization approaches [7, 16, 26] infeasible to solve our problem. Therefore, the biggest challenge here is to estimate the evaluation function before the model execution. Moreover, even we somehow know models' outputs in advance, how to schedule the execution of them to maximize the evaluation function under multi-dimensional constraints remains a hard problem.

4.2 Core Idea and Design Overview

Based on the observation that executed models' outputs could provide hints for the outputs of unexecuted ones, we propose to utilize the implicit *semantic relationship* among labeling capacities of models to estimate the evaluation function before the actual execution.

4.2.1 Handcrafted rule-based solution. A straightforward solution to characterize the semantic relationship is to manually design some model execution rules based on common sense, which increase (or decrease) the probabilities to execute models that ought to (or not to) output valuable labels. For instance, if the label "undersea" is returned by a scene classifier with high confidence, the probability of running a fish classifier should be increased, while a household object detector should be assigned with a low probability for execution. Table 2 gives more examples of handcrafted rules. The number of models and the supported labels, however, could be too large to design a set of effective rules. In our implementation, 30 deep learning models support 1104 labels in total. Designing execution rules for a system at that scale is cumbersome and ineffective. The evaluation

results in Section 7.3 show that handcrafted rules slightly improve the performance compared with the "random policy", but still leaves a large room for optimization.

4.2.2 Reinforcement learning-based framework. We propose a novel deep reinforcement learning (DRL) based approach to better characterize the complex semantic relationship among models and predict the value of unexecuted models.With the estimated value of the evaluation function, we design scheduling algorithms to optimize the model execution under different constraints. As shown in Fig. 3, our proposed framework consists of two components:

(1) Model value prediction. A DRL model is trained to map the labeling state, i.e., outputs of currently executed models, to values of unexecuted models. When each piece of input data arrives, the labeling state is initialized to an empty set. In each iteration, the DRL agent predicts the values of remaining unexecuted models based on the current state.

(2) Adaptive model scheduling. According to the predicted values of unexecuted models, the adaptive scheduling algorithm determines the model (or a set of models) to execute in this iteration based on the current computing resources and/or delay constraints. Two common types of constraints, delay and GPU memory, are considered in this work. The labeling state is updated according to the actual model execution results and a new iteration begins.

This "prediction-scheduling-execution" loop continues until a stopping condition is reached, e.g., the running time reaches the deadline constraint. Our two-components framework disentangles models' semantic relationship from their computing resource costs. The model value prediction component only cares about whether the model will output valuable labels or not, leaving the costs issue to the scheduling algorithms. In this way, the learned semantic relationship can be adopted flexibly in various situations with different resource constraints. In the following sections, we present the design of each component in detail.

5 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 model value prediction task. Deep reinforcement learning has been applied in many fields, including robotics control [15], video games [21], system tuning[35], etc. 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, i.e., $|L(M)|$). 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. Each model in M is regarded as an action and the complete set of models compose a $|M|$-dimension discrete action space. In every iteration, after the DRL agent selects an action/model $m \in M$, the system executes the model on the input data d. According to the execution result $O(m, d)$, the labeling state is updated and the agent receives a reward. The reward function is the key to the performance of the DRL agent. We cannot directly adopt Eq. (1) as the reward function since it evaluates the final value of the complete output set when the execution stops. The DRL agent, however, needs a reward function for intermediate feedback to each selected action.

5.1 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 p_i. However, based on the confidence of each output label, directly using

Eq. (1) as the reward function has some shortcomings. The first problem is that the number of output labels influences the reward overly, which varies among different deep learning models. As an example, an action classification model usually outputs only one label, while a face landmark detector outputs at most 70 keypoint labels for each face in the image. In this case, the agent would easily tend to prefer the face landmark detector regardless of the data content, due to the overwhelming returned value. We use a logarithmic function to mitigate the bias caused by different numbers of models' output labels. Our experimental results show that other smoothing functions such as the average confidence of a model's output labels can achieve a similar effect as long as the values of different models are in the same order of magnitude. 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. For example, in a surveillance video monitoring system, when computing resources are insufficient, an abnormal event detection model should have a higher priority than a common object detector. Experimental results in Section 7.5 show that by adjusting the parameter $theta$, we can effectively control the model execution priority without sacrificing the optimization performance.

Taking all the aforementioned factors into consideration, the reward function is defined as:

$$r(m,d) = \begin{cases} \ln(\theta_m \sum_{l_i \in O'(\{m\},d)} l_i.conf + 1) & , O'(\{m\},d) \neq \emptyset \\ -1 & , O'(\{m\},d) = \emptyset \end{cases} \tag{3}$$

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.

5.2 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. The traditional table-based approach cannot handle this task since the mapping space increases exponentially with the number of labels and models. Facing the complexity of the problem, we adopt a deep Q-value network (DQN) [21] 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 [21], Double DQN [27], Dueling DQN [28] and Deep SARSA [5]. We compare their performance in Section 7.2. Theoretically, the proposed DRL-based agent can be trained by any Q-value network-based DRL approach.

END action. 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. 3, 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. Note that, the "END" action is only used in

the training process. In the scheduling process, the agent stops when certain computing resource runs out, e.g., running time exceeds the delay budget.

6 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 $f(S, d)$. It is worth mentioning that if there is no resource constraint, the DRL agent itself can schedule the model execution in a greedy manner which selects the remaining model with maximal Q-value at each iteration and terminates when all models have been executed. In this work, we consider the two most common constraints for data labeling tasks: deadline and limited memory. We first analyze the scheduling problem in a single-processor setting with a deadline constraint and then in a multi-processor setting with both deadline and memory constraints.

6.1 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. For example, a video surveillance system needs to respond to suspicious events or objects in seconds. With the deadline constraint, our optimization problem defined in Eq. (2) is specified as follows:

$$\begin{aligned} &\max_{S \subseteq M} f(S, d) \\ &\text{s.t.} \sum_{m \in S} m.time \leq B_{time} \end{aligned} \tag{4}$$

, 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. It is obvious that in the worst case, the performance of this algorithm can be arbitrarily bad. A greedy algorithm combined with the partial enumeration of all subsets of size 3 has been proved to achieve a $1 - 1/e$ approximation [26]. In the traditional knapsack problem, the value of each item is known (easily calculated) and fixed. In our problem, however, the real model value is unknown and we can only obtain the exact value after model execution. Therefore, the enumeration-based algorithm is infeasible for our problem, not to mention its high computational complexity ($O(|M|^5)$) [26]. Moreover, the prediction of a model's value is not fixed, which changes with the labeling state. The uncertainty and dynamics of the model value make the scheduling task particularly challenging. 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. Algorithm 1 demonstrates the scheduling policy in detail. This efficient algorithm achieves near-optimal performance in our extensive evaluations (see Section 7.6), which also verifies the effectiveness of our DRL-based model value prediction method.

6.2 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

Algorithm 1 Model 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
```

considered for each input data. The tangible formulation of this optimization problem is:

$$\begin{aligned} &\max_{S \subseteq M} f(S, d), \text{ where } S = \bigcup_{i=1}^{N} S_i \\ \text{s.t. } &\sum_{i=1}^{N} S_i.time \leq B_{time} \\ &\sum_{m \in S_i} m.mem \leq B_{mem} \ (\forall i, 1 \leq i \leq N) \end{aligned} \tag{5}$$

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}.

The optimization problem in the multi-processor setting is more challenging than that in the single-processor case. The most related problem is the two-dimension orthogonal knapsack problem: given a set of squares, each of which has a profit, and the objective is to pack a subset of squares into a fixed-size square to maximize the total profit. It is a NP-hard problem and several polynomial time approximation schemes (PTAS) have been proposed for this problem [10]. Due to the uncertainty and dynamics of model value prediction, however, those algorithms with performance guarantee are not feasible to solve our problem. In this work, we design an efficient heuristic algorithm. 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. See Algorithm 2 for details. The intermediate model value is predicted by the pre-trained DRL agent. Once a model completes its execution, its occupied memory will be released and a new iteration will begin. We conduct extensive evaluations in Section 7.7 to show the effectiveness of our algorithm in practice.

Algorithm 2 Model 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 $m.mem > B_{mem}$
  $m^* \leftarrow \arg\max_{m \in M \setminus S} \frac{Q(m,d)}{m.time \times m.mem}$
  $S_t \leftarrow S_t \cup \{m^*\}$
  $B^t_{time} \leftarrow TimeCost + m^*.time$
  Filter out models by temporary deadline B^t_{time}
  while $B_{mem} > 0$ **do**
    $m^{*'} \leftarrow \arg\max_{m \in M \setminus S} \frac{Q(m,d)}{m.mem}$
    $S_t \leftarrow S_t \cup \{m^{*'}\}$
    $B_{mem} \leftarrow B_{mem} - m^{*'}.mem$
  end while
  $S.append(S_t)$
  Wait until one model $m^{*''} \in S_t$ finishes execution
  Update $TimeCost$
  $B_{mem} \leftarrow B_{mem} + m^{*''}.mem$
  $S_t \leftarrow S_t \setminus \{m^{*''}\}$
end while
return S

6.3 Performance Analysis

Due to the unknown real model value and the ever-changing predicted model value, existing algorithms for both the submodular function optimization with knapsack constraint and the two-dimension orthogonal knapsack constraints cannot be applied to solve our problem. To measure the performance of our heuristic algorithms, we need to compare them with the optimal solution. Under certain deadline and deadline-memory constraints, the NP-hard problem requires to enumerate $O(|M|!)$ 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 problem to that even though the remaining resources are not enough for one model to complete its execution, the model can still be selected into the set S and contribute the corresponding proportion of its value to $f(S, d)$. 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 of our original problem. Experiment results in Section 7 show that the performance ratio between our heuristic algorithms and the "optimal* policy" exceeds 0.7 in most cases, which means that the true performance ratio of our algorithm is better than 0.7 in most cases. More evaluations of our algorithms will be presented in Section 7.

7 EVALUATION

We implemented the proposed adaptive model scheduling framework and conducted extensive evaluations for large-scale comprehensive image labeling tasks. Our framework can also be adapted for other types of data labeling tasks. We will open source our code for use or modification.

7.1 Experimental Setup

Deep learning models. We consider 10 diverse visual analysis tasks and deployed a total of 30 popular deep learning models for these 10 tasks by directly utilizing the off-the-shelf pretrained weights or training models on public datasets. These models can label images with a wide range of semantic information (1104 different labels in all). Table 1 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.

Task Name	Label#
Object Detection [22]	80
Place Classification [38]	365
Face Detection [1]	1
Face Landmark Localization [2]	70
Pose Estimation [30]	17
Emotion Classification [8]	7
Gender Classification [25]	2
Action Classification [3]	400
Hand Landmark Localization [24]	42
Dog Classification [14]	120
10 Tasks	**1104 Labels**

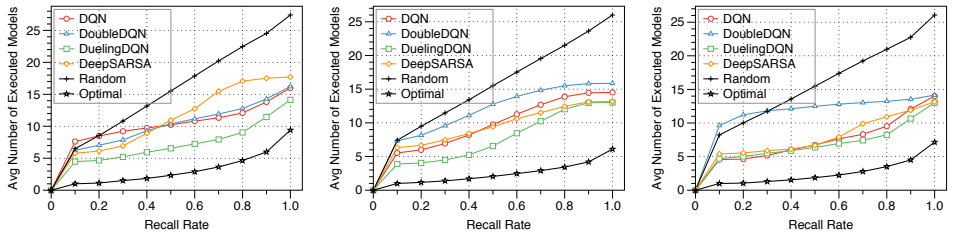
Table 1. Summary of 10 visual analysis tasks.

Datasets and ground truth. We conducted experiments on five public image datasets: 1) Stanford40 [33], 2) PASCAL VOC 2012 [6], 3) MSCOCO 2017 [19], 4) MirFlickr25 [11] and 5) Places365 [38]. 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 5 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.

Hardware setting. For all evaluations, we employed a server with 48 Intel Xeon CPU E5-2650 v4 cores and one Tesla P100 GPU.

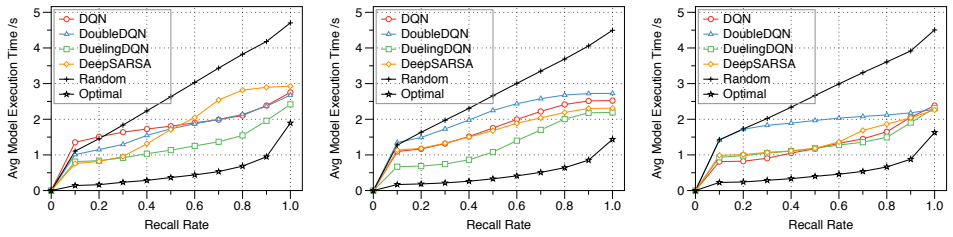
7.2 RL-Based Model Value Prediction

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 5.2, including DQN [21], DoubleDQN [27], DuelingDQN [28] and DeepSARSA [5]. We trained and tested these four DRL agents on three datasets, MSCOCO 2017, MirFlickr25 and Places365, separately. We cannot directly compare the model value predicted by DRL agents with the ground truth. The reason is that the predicted Q value of a model changes with the labeling state in each iteration, which is dependent on the set of currently executed models. For example, when the labels of a model have been output by other executed models, the predicted Q value of the model is -1 rather than the value of its output. To evaluate the performance of these agents, we conducted



(a) MSCOCO 2017 (b) MirFlickr25 (c) Places365

Fig. 4. The average number of executed models per image vs. required recall rate of output value.



(a) MSCOCO 2017 (b) MirFlickr25 (c) Places365

Fig. 5. The average time cost of executed models per image vs. required recall rate of output value.

the following experiments. 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 value reaches a given threshold [1]. Without considering any delay or resource constraint, the model selection is only dependent on the predicted model value. The more accurate the predicted model value, the fewer the average number of executed models per image and the shorter the total execution time. 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.

- **Random policy**: randomly selects the next model, until the recall rate of true output value exceeds the threshold;
- **Optimal policy**: selects models in the descending order of their true output value, until the recall rate of true output value exceeds the threshold.

Fig. 4 and Fig. 5 show the experimental results of different policies on three diverse datasets, using the average executed number of models and average execution time as the metric respectively. The optimal policy consumes the smallest number of model executions to reach a given recall rate of output value since it knows the true value of each model. There is a significant gap between the random policy and the optimal policy. Compared with the random policy, the optimal policy saves 79.3-84.0% (or 65.6-76.5%) the number of model executions when the recall rate is 0.8 (or 1.0). 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 44.1-60.6% (or 48.4-50.0%) the number of model executions when the required recall rate is 0.8 (or 1.0), compared with the random policy. Moreover, Fig. 5 shows that, compared

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

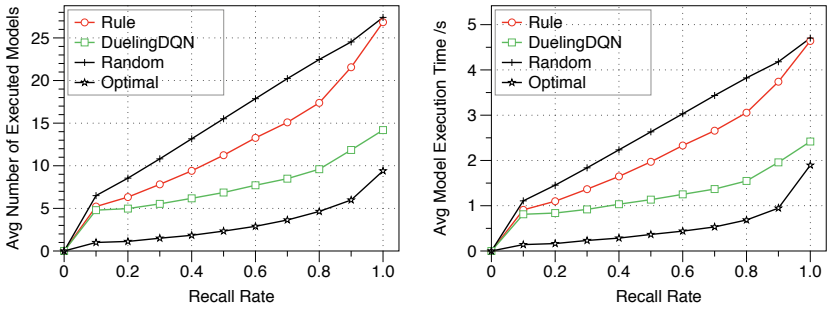


Fig. 6. (Left) Average number of executed models and (Right) average model execution time, vs. required recall rate of output value.

with the random policy, the DRL agent saves 45.6-59.5% (or 48.6-51.2%) execution time with 0.8 (or 1.0) recall rate. Besides, we notice that in both Fig. 4 and Fig. 5 the growth trends of model execution cost for DRL-based policies are similar to that of the random policy when the recall rate is under 10%. As the recall rate gets larger, the growth trends for the DRL-based policies are more and more similar to that of the optimal solution. It meets the expectation of our design: the DRL agent initially selects models in an almost random way due to the lack of information about the data; after obtaining some execution results, the agent can predict the model value accurately and select the next model with the largest value as the optimal policy does.

7.3 Agent Knowledge vs. Handcrafted Rules

A direct way to optimize model scheduling is to manually design execution rules based on common sense (see Section 4.2). To explore the effectiveness of handcrafted rules, we asked three volunteers with certain deep learning related knowledge to provide execution rules they can think of. We selected 10 rules (listed in Table 2) accepted by all of them and implemented a rule-based model scheduling policy.

Current Model Task	Output Label	Rule
Object Detection	person	2×$\mathcal{P}$(Pose Estimation)
Object Detection	person	2×$\mathcal{P}$(Gender Classification)
Object Detection	dog	2×$\mathcal{P}$(Dog Classification)
Face Detection	face	2×$\mathcal{P}$(Face Landmark Localization)
Face Detection	face	2×$\mathcal{P}$(Emotion Classification)
Pose Estimation	body keypoints	2×$\mathcal{P}$(Action Classification)
Pose Estimation	wrist keypoints	2×$\mathcal{P}$(Hand Landmark Localization)
Place Classification	indoor places	0.5×$\mathcal{P}$(Animal-Object Detection)
Place Classification	indoor places	0.5×$\mathcal{P}$(Sport-Action Classification)

Table 2. Ten handcrafted model execution rules.

Let $\mathcal{P}$(Task) denote the probability of executing models for certain "Task". Initially, all models have equal execution probabilities. After obtaining certain labels, the policy updates the execution probabilities of models according to the rules. Fig. 6 shows the experimental results on MSCOCO 2017. DuelingDQN agent significantly outperforms the rule-based policy, which only saves 22.6%

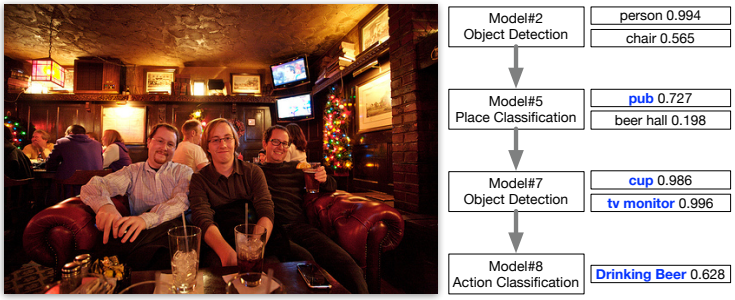
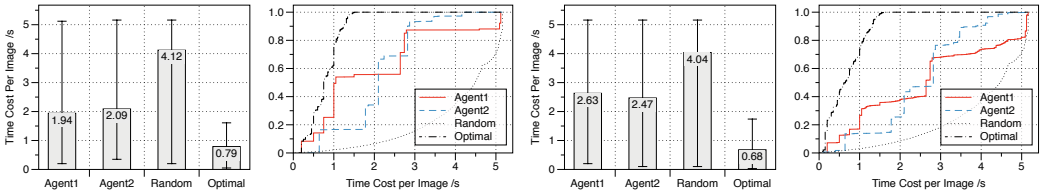


Fig. 7. Model execution sequence of an image in MirFlickr25, scheduled by DuelingDQN agent with Q-value greedy policy.



(a) Average Time Cost $Dataset_1$ (b) Time Cost CDF $Dataset_1$ (c) Average Time Cost $Dataset_2$ (d) Time Cost CDF $Dataset_2$

Fig. 8. Model execution time cost by different polices, tested on $Dataset_1$ and $Dataset_2$.

(or 2.1%) executed models and 20.1% (or 1.4%) running time with 0.8 (or 1.0) recall rate. Results on other datasets are similar.

To understand the difference between the handcraft rule and the semantic relationship learned by the DRL agent, we visualize model execution sequences that are scheduled by the Q-value greedy policy of DuelingDQN agent on different sample images. Fig. 7 shows an example of model execution sequence for an image (im6696.jpg) in MirFlickr25 dataset. The knowledge of DRL agent meets the common sense that "there should be some *cups* in the *pub* and people may *drink beer* here". We can infer that the implicit semantic relationship among labeling capacities of deep learning models are highly complicated. Even though the designers of the rules in Table 2 have some knowledge of deep learning models, the handcrafted rules only consider the pair-wise relationship and the influence on execution probabilities are fixed (2 or 0.5 times probability). The more rules we designed, the harder to adjust the influence of each rule. So we hold that designing a set of effective execution rules is infeasible for data labeling tasks at scale.

7.4 Knowledge Transferability

The experiments above show the effectiveness of DRL agents that were trained and tested on subsets of the same dataset. An interesting question is: how *transferable* the learned knowledge is? Stanford40 and PASCAL VOC 2012 are two datasets that have obvious variance in content distribution. Stanford40[33] is collected for the action recognition task which mainly consists of scenes related to human activities. PASCAL VOC 2012 dataset[6] covers a larger range of visual objects, including animals, vehicles, household furniture, etc. Since DuelingDQN outperforms other DRL schema (§7.2), we use the experimental results of DuelingDQN agents as the representative.

Two DuelingDQN-based agents are trained on the training samples from Stanford40 and PASCAL VOC 2012 respectively and tested them on both testing sets of the two datasets. In this experiment, we use the average model execution time cost when all output value is recalled to measure the effectiveness of the agent. The scheduling policy of DRL agents is still the Q-value greedy policy. Random policy and optimal policy are also utilized as comparisons. For simplicity, we use the following notations:

- $Dataset_1$: Stanford40 testing set;
- $Datasset_2$: PASCAL VOC 2012 testing set;
- $Agent_1$: DRL agent trained on Stanford40 training set;
- $Agent_2$: DRL agent trained on PASCAL VOC 2012 training set.

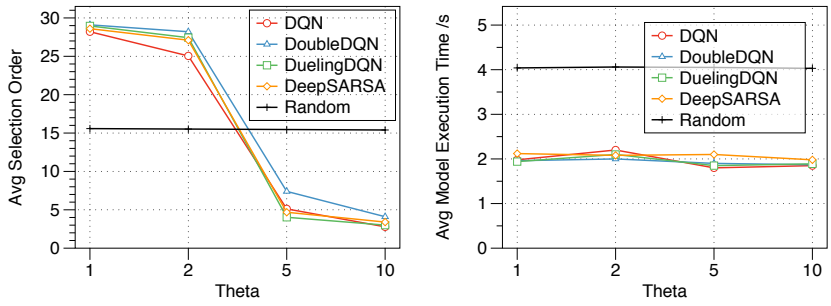
As shown in Fig. 8, executing all 30 models costs 5.16s per image and the random policy averagely takes 4.12s on $Dataset_1$ and 4.04s on $Dataset_2$. Compared with the random policy, $Agent_1$ costs 1.94s (or 2.63s) on $Dataset_1$ (or $Dataset_2$). $Agent_2$ costs 2.09s (or 2.47s) on $Dataset_1$ (or $Dataset_2$). So on average, DRL agents save 51.1% running time on $Dataset_1$ and 36.9% running time on $Dataset_2$. It is inspiring that for one dataset the knowledge learned from the other dataset with different content distribution can also optimize scheduling effectively. Cumulative distribution figures (CDF) in Fig. 8 show more details about the time cost distributions. We can infer that although the knowledge learned from the two datasets vary a lot, both of them could be utilized to optimize the model scheduling on a wide range of image datasets efficiently.

Limitations: Although the experimental results demonstrate the transferability of the learned semantic relationship among models by DRL agents, we want to highlight some limitations. 1) Assumption of the intersected content distribution. Stanford40 and PASCAL VOC have quite different content distributions, but effective optimization is still dependent on the assumption. We studied some extreme cases, for example, training agents only by dog-related images and testing them by human action-related images, which show worse model scheduling than the random policy and vice versa. 2) The requirement of relatively adequate training samples. Similar to other learning-based models, an effective DRL agent has some requirements for the volume of training samples. An empirical size of the training set is around 10-20% of the whole dataset with a random sampling method, though there is no theoretical proof of the convergence.

7.5 Model Priority

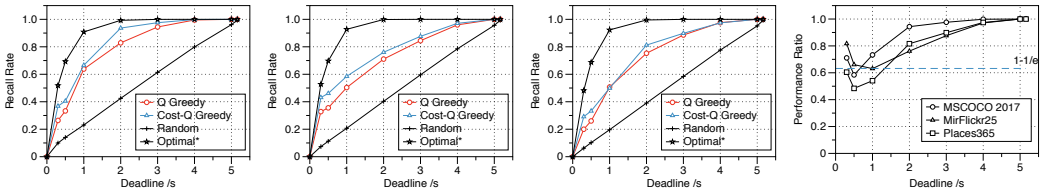
As introduced in Section 5, we use the parameter θ to control model priorities. It is worth mentioning that the experiments above were conducted with identical θ value (1.0) for every model. To study the effect of θ, we train DRL agents by setting the parameter of one "face detection model" to 1.0, 2.0, 5.0 and 10.0 to increase its priority, while keeping θ of other models as 1.0. Intuitively, with a higher θ parameter, we expect the DRL agent to predict a higher Q value for the "face detection" model in the early stages. So we analyze the order of the "face detection model" in the scheduling sequences.

Fig. 9(a) shows that the increased θ effectively brings forward the execution of the "face detection" model. It means that the user can obtain his/her preferred label ("face" in this experiment) with a shorter delay. As a representative, DuelingDQN agents schedule the "face detection" model at the 28.9 / 27.4 / 4.0 / 3.0 iterations on average, with 1.0 / 2.0 / 5.0 / 10.0 θ value. Meanwhile, as shown in Fig. 9(b), although θ shifts the order of model execution sequences, the DRL agents still keep a good optimization performance on the total execution time (with 1.0 recall rate of label value). On average, the DuelingDQN agents save 51.9 / 48.2 / 54.3 / 53.1% running time without any loss of valuable labels, compared with the random policies.



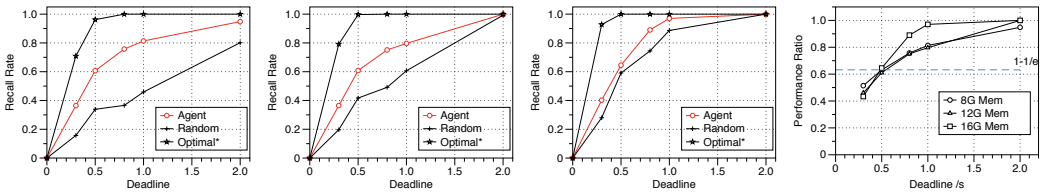
(a) Average execution order (b) Average time cost

Fig. 9. The effect of adjusting priority parameter θ of "face detection" model.



(a) MSCOCO 2017 (b) MirFlickr25 (c) Places365 (d) Performance ratio

Fig. 10. Value recall rate under deadline constraints.



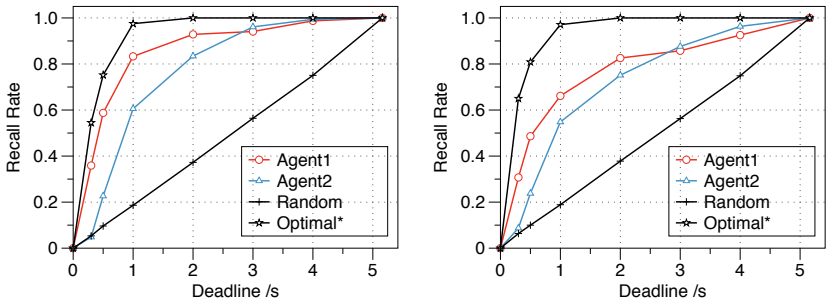
(a) 8GB Memory (b) 12GB Memory (c) 16GB Memory (d) Performance ratio.

Fig. 11. Value recall rate under different memory and deadline constraints.

Practical utility. Being able to control the model execution priorities, while keeping the excellent scheduling performance, is quite valuable in data labeling applications. For example, in a surveillance video monitoring system, we wish to obtain the output of the "abnormal action detection" model with higher priority (thus shorter delay) than the "common object localization" model, when the computing resources are limited. However, simply using a fixed scheduling policy will cause a serious waste of resources, since abnormal actions occur much less frequently than common objects. Then the ability of the proposed DRL agent is crucial to solving the challenge: flexibly controlling the model priorities and keeping effective optimization of scheduling.

7.6 Scheduling under Deadline Constraint

Experiments above use the required output value recall rate as the condition to terminate the scheduling process. More practical usage of the proposed framework is scheduling models under



(a) *Dataset*1 (b) *Dataset*2

Fig. 12. Value recall rate under deadline constraints.

computing resources or delay constraints. 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**: randomly selects model until the deadline;
- **Optimal* policy**: greedily selects the model with maximal $\frac{f(S\cup\{m\},d)-f(S,d)}{m.time}$ with relaxing constraint (§6.3);
- **Q-Greedy policy**: greedily selects the model with maximal Q value until the deadline (§7.2).

As a representative, all DRL agents in the following experiments are trained with DuelingDQN schema. Fig. 10(a-c) show that Algorithm 1 outperforms the Q-greedy policy on three datasets. Algorithm 1 boosts the value recall rate by 188.7-309.5% with a 0.5s delay budget, compared with the random policies. We plot the performance ratio of Algorithm 1 to optimal* policy in Fig. 10(d), which shows that in most cases our cost-Q greedy algorithm performs better than the provable guarantee $1 - 1/e$ of classic approaches.

Moreover, we evaluate $Agent_1$ and $Agent_2$ on $Dataset_1$ and $Dataset_2$ (§7.4), using Algorithm 1 instead of Q-greedy policy. As shown in Fig. 12, the DRL agent knowledge is transferable between dataset with different content distributions. With 1.0s deadline constraint, $Agent_1$ and $Agent_2$ improve the recalled value by 346.8 (or 250.5%) and 224.9 (or 190.5%) on $Dataset_1$ (or $Dataset_2$).

7.7 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. Random policy and optimal* policy are used as baselines:

- **Random policy**: randomly selects model that could be packed into GPU to execute until the deadline;
- **Optimal* policy**: greedily selects the model with maximal $\frac{f(S\cup\{m\},d)-f(S,d)}{m.time*m.mem}$ with relaxing constraints (§6.3);

As a representative, we use the DuelingDQN $Agent_1$ and $Dataset_2$ as the test set, which are the worst cases in our experiments. As shown in Fig. 11(a-c), 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% / 19.5% under 8GB / 12GB / 16GB GPU memory and 0.8s deadline constraints. With the increasing memory allocated for learning models, the room for improvement between random policy and optimal* policy shrinks. So it is reasonable that the improvement brought by Algorithm 2 is relatively small with 12GB and 16GB memory. As shown in Fig. 11(d), the performance ratio of Algorithm 2 to optimal* policy exceeds $1 - 1/e$ in most cases.

7.8 Scheduling Overhead

Experimental results above illustrate the effectiveness of DRL agents and adaptive model scheduling algorithms. Here, we measure the additional overhead brought by our framework. Fortunately, as shown in Table 3, the time cost for making one selection is only around 3-6ms, which is negligible compared with the execution time of deep learning models (50-400ms). For memory usage, the DRL agents require about 100MB CPU memory, which is quite lightweight compared with the deployed visual analysis models that consume 500MB to 8GB GPU memory.

	DRL Agent	**Deep Learning Model**
Time	3-6ms	50-400ms
Memory	100MB (CPU)	500-8000MB (GPU)

Table 3. Computing cost of DRL agent and models.

8 CONCLUSION & FUTURE WORK

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 compared with other approaches. Several challenging issues are left as future work. A critical innovative component of our framework is the propose and construction of the model-relationship graph. Firstly, we would like to design a fast method to construct this efficiently and effectively. Secondly, designing scheduling algorithms with theoretical performance guarantees under different constraints remains a very challenging and attractive problem. Recall that many notoriously difficult to tackle NP-hard problems are special cases of our scheduling problem. At last, we also need to further evaluate our proposed method on other data types like text and audio.

ACKNOWLEDGMENTS

The research is supported by National Key R&D Program of China 2018YFB0803400, China National Natural Science Foundation with No. 61822209, No. 61932016, 61625205, 62132018, Key Research Program of Frontier Sciences, CAS, No. QYZDY-SSW-JSC002.

REFERENCES

[1] Brandon Amos, Bartosz Ludwiczuk, and Mahadev Satyanarayanan. 2016. OpenFace: A general-purpose face recognition library with mobile applications. Technical Report. CMU-CS-16-118, CMU School of Computer Science.
[2] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. 2018. OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields. In arXiv preprint arXiv:1812.08008.
[3] Joao Carreira and Andrew Zisserman. 2017. Quo vadis, action recognition? a new model and the kinetics dataset. In proceedings of the IEEE CVPR. 6299–6308.

[4] Wenlin Chen, James Wilson, Stephen Tyree, Kilian Q Weinberger, and Yixin Chen. 2016. Compressing convolutional neural networks in the frequency domain. In Proceedings of the 22nd ACM SIGKDD. ACM, 1475–1484.

[5] Marco Corazza and Andrea Sangalli. 2015. 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).

[6] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. 2010. The pascal visual object classes (voc) challenge. International journal of computer vision 88, 2 (2010), 303–338.

[7] Daniel Golovin and Andreas Krause. 2011. Adaptive submodularity: Theory and applications in active learning and stochastic optimization. Journal of Artificial Intelligence Research 42 (2011), 427–486.

[8] Ian J. Goodfellow, David Warde-Farley, Pascal Lamblin, Vincent Dumoulin, Mehdi Mirza, Razvan Pascanu, James Bergstra, Frédéric Bastien, and Yoshua Bengio. 2013. Pylearn2: a machine learning research library. arXiv preprint arXiv:1308.4214 (2013). http://arxiv.org/abs/1308.4214

[9] Song Han, Jeff Pool, John Tran, and William Dally. 2015. Learning both weights and connections for efficient neural network. In Advances in neural information processing systems. 1135–1143.

[10] Xin Han, Kazuo Iwama, and Guochuan Zhang. 2008. On Two Dimensional Orthogonal Knapsack Problem. arXiv preprint arXiv:0803.4260 (2008).

[11] Mark J Huiskes and Michael S Lew. 2008. The MIR flickr retrieval evaluation. In Proceedings of the 1st ACM international conference on Multimedia information retrieval. ACM, 39–43.

[12] Junchen Jiang, Ganesh Ananthanarayanan, Peter Bodik, Siddhartha Sen, and Ion Stoica. 2018. Chameleon: scalable adaptation of video analytics. In Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication. ACM, 253–266.

[13] Lukasz Kaiser, Aidan N Gomez, Noam Shazeer, Ashish Vaswani, Niki Parmar, Llion Jones, and Jakob Uszkoreit. 2017. One model to learn them all. arXiv preprint arXiv:1706.05137 (2017).

[14] Aditya Khosla, Nityananda Jayadevaprakash, Bangpeng Yao, and Li Fei-Fei. 2011. Novel Dataset for Fine-Grained Image Categorization. In First Workshop on Fine-Grained Visual Categorization, IEEE CVPR. Colorado Springs, CO.

[15] Jens Kober, J Andrew Bagnell, and Jan Peters. 2013. Reinforcement learning in robotics: A survey. The International Journal of Robotics Research 32, 11 (2013), 1238–1274.

[16] Andreas Krause and Daniel Golovin. [n.d.]. Submodular function maximization.

[17] Tong Li, Sheng Gao, and Yajing Xu. 2017. Deep multi-similarity hashing for multi-label image retrieval. In Proceedings of the 2017 ACM on Conference on Information and Knowledge Management. ACM, 2159–2162.

[18] Xiang-Yang Li, Jianwei Qian, and Xiaoyang Wang. 2018. Can China lead the development of data trading and sharing markets? Commun. ACM 61, 11 (2018), 50–51.

[19] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In European conference on computer vision. Springer, 740–755.

[20] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H Chi. 2018. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In Proceedings of the 24th ACM SIGKDD. ACM, 1930–1939.

[21] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602 (2013).

[22] Joseph Redmon and Ali Farhadi. 2018. YOLOv3: An Incremental Improvement. arXiv (2018).

[23] Sebastian Ruder. 2017. An overview of multi-task learning in deep neural networks. arXiv preprint arXiv:1706.05098 (2017).

[24] Tomas Simon, Hanbyul Joo, Iain Matthews, and Yaser Sheikh. 2017. Hand Keypoint Detection in Single Images using Multiview Bootstrapping. In CVPR.

[25] Karen Simonyan and Andrew Zisserman. 2014. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014).

[26] Maxim Sviridenko. 2004. A note on maximizing a submodular set function subject to a knapsack constraint. Operations Research Letters 32, 1 (2004), 41–43.

[27] Hado Van Hasselt, Arthur Guez, and David Silver. 2016. Deep reinforcement learning with double q-learning. In Thirtieth AAAI.

[28] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Van Hasselt, Marc Lanctot, and Nando De Freitas. 2015. Dueling network architectures for deep reinforcement learning. arXiv preprint arXiv:1511.06581 (2015).

[29] Bichen Wu, Forrest N Iandola, Peter H Jin, and Kurt Keutzer. 2017. SqueezeDet: Unified, Small, Low Power Fully Convolutional Neural Networks for Real-Time Object Detection for Autonomous Driving.. In CVPR Workshops. 446–454.

[30] Yuliang Xiu, Jiefeng Li, Haoyu Wang, Yinghong Fang, and Cewu Lu. 2018. Pose Flow: Efficient Online Pose Tracking. In BMVC.

[31] Donna Xu, Yaxin Shi, Ivor W Tsang, Yew-Soon Ong, Chen Gong, and Xiaobo Shen. 2019. A Survey on Multi-output Learning. arXiv preprint arXiv:1901.00248 (2019).

[32] Yang Yang, Yi-Feng Wu, De-Chuan Zhan, Zhi-Bin Liu, and Yuan Jiang. 2018. Complex Object Classification: A Multi-Modal Multi-Instance Multi-Label Deep Network with Optimal Transport. In Proceedings of the 24th ACM SIGKDD. ACM, 2594–2603.

[33] Bangpeng Yao, Xiaoye Jiang, Aditya Khosla, Andy Lai Lin, Leonidas Guibas, and Li Fei-Fei. 2011. Human action recognition by learning bases of action attributes and parts. In ICCV. IEEE, 1331–1338.

[34] Mu Yuan, Lan Zhang, Xiang-Yang Li, and Hui Xiong. 2020. Comprehensive and Efficient Data Labeling via Adaptive Model Scheduling. In 2020 IEEE 36th International Conference on Data Engineering (ICDE). 1858–1861. https://doi.org/10.1109/ICDE48307.2020.00188

[35] Ji Zhang, Yu Liu, Ke Zhou, Guoliang Li, Zhili Xiao, Bin Cheng, Jiashu Xing, Yangtao Wang, Tianheng Cheng, Li Liu, et al. 2019. An end-to-end automatic cloud database tuning system using deep reinforcement learning. In Proceedings of the 2019 International Conference on Management of Data. ACM, 415–432.

[36] Zheng Zhang, Qin Zou, Qian Wang, Yuewei Lin, and Qingquan Li. 2018. Instance similarity deep hashing for multi-label image retrieval. arXiv preprint arXiv:1803.02987 (2018).

[37] Fang Zhao, Yongzhen Huang, Liang Wang, and Tieniu Tan. 2015. Deep semantic ranking based hashing for multi-label image retrieval. In Proceedings of the IEEE CVPR. 1556–1564.

[38] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. 2017. Places: A 10 million Image Database for Scene Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence (2017).