Comprehensive and Efficient Data Labeling via Adaptive Model Scheduling

Mu Yuan¹, Lan Zhang¹, Xiang-Yang Li¹, Hui Xiong²

¹University of Science and Technology of China
²Rutgers University

Lab for Intelligent Networking and Knowledge Engineering
Outline

• **Resource-wasting multi-model inference workloads**
  • Rule-based scheduler
  • Learning-based scheduler
  • Evaluation
  • Conclusion
Multi-Model Inference Workloads

- Image Retrieval
  - walking dogs

- Image Labeling
Multi-Model Inference Workloads

- Walking dogs
- Image Retrieval
- Image Labeling

Degree of Congestion: HIGH

ILLEGAL

20 mph
Multi-Model Data Labeling Workloads

Data Search Engine

Raw Data

Labels

Object Detector

Scene Classifier

... person, 0.994
chair, 0.565
tv monitor, 0.996
pub, 0.727
beer hall, 0.198
...

...
Observation

CV Models

Raw Images

- **Pose Estimator**
- **Face Detector**
- **Object Detector**
- **Action Classifier**
- **Scene Classifier**
- **Dog Classifier**

**Useful Output**  **No Output**  **Low-Confidence Output**
Observation

<table>
<thead>
<tr>
<th>Pose Estimator</th>
<th>Body Keypoints</th>
<th>Face Location</th>
<th>Body Keypoints</th>
</tr>
</thead>
<tbody>
<tr>
<td>Face Detector</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Object Detector</td>
<td>Dog (0.96)</td>
<td>Person (0.43)</td>
<td>Person (0.96)</td>
</tr>
<tr>
<td>Action Classifier</td>
<td>Fall Down (0.87)</td>
<td>Make Up (0.9)</td>
<td>Bike (0.97)</td>
</tr>
<tr>
<td>Scene Classifier</td>
<td>Lawn (0.85)</td>
<td>Lobby (0.91)</td>
<td>Bathroom (0.14)</td>
</tr>
<tr>
<td>Dog Classifier</td>
<td>Akita (0.91)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Wasted computing resources!
Wasted computing resources!
Data-Driven Analysis

394,170 images
30 CV models
Data-Driven Analysis

394,170 images
30 CV models

Executing **ALL** models.
Data-Driven Analysis

394,170 images
30 CV models

Executing models that output **VALUABLE** labels.
Data-Driven Analysis

394,170 images
30 CV models

Valuable(model, img) = (Output(model, img).conf > 0.5).any() ? True : False;
394,170 images  
30 CV models

Valuable(model, img) = (Output(model, img).conf > 0.5).any() ? True : False;
Data-Driven Analysis

394,170 images
30 CV models

Valuable\((\text{model, img})\) = (\text{Output}(\text{model, img}).\text{conf} > 0.5).\text{any}() \ ? \ True : False;

Executing models RANDOMLY.
Without assumption of input data distribution, how to predict the value of models before execution?
Without assumption of input data distribution, how to predict the value of models before execution?

**IMPOSSIBLE** for single-model workloads …

But not for the multi-model tasks!
Without assumption of input data distribution, how to predict the value of models before execution?

IMPOSSIBLE for single-model workloads ... But not for the multi-model tasks! We can get hints from executed models.
Outline

• Resource-wasting multi-model inference workloads
• Rule-based scheduler
• Learning-based scheduler
• Evaluation
• Conclusion
Manual Rules

Unknown Content

Object Detector

Action Classifier

Vehicle Classifier

person, 0.82

car, 0.05
Limitations

- Hard to express the non-pairwise rules.
- Too expensive for large-scale semantic labels.
- Challenging to tune the effects of multiple rules.
• Hard to express the non-pairwise rules.
• Large-scale semantic labels (>1000 labels in our workloads).
• Effects of multiple rules are difficult to tune.

*Deep learning could help!*
• Resource-wasting multi-model inference workloads
• Rule-based scheduler
• **Learning-based scheduler**
• Evaluation
• Conclusion
Framework

One dimension per label.
Framework

Input Data → Model Execution → Q-Value Network (DRL Agent) → Resource Constraints

Model Value Prediction → Update → Labeling State

One **action** per model.
For versatility of DRL agent, resource constraints are left to scheduling algorithms.
Framework

Reward:
\[ r(w, d) = \log(\theta_m \sum_{l_i \in O'(\{m\}, d)} l_i \cdot \text{conf} + 1), \quad O'(\{m\}, d) \neq \emptyset \]
\[ r(w, d) = -1, \quad O'(\{m\}, d) = \emptyset \]
**Framework**

**Reward:**

\[ r(w, d) = \log(\theta_m \sum_{l_i \in O'(\{m\}, d)} l_i \cdot \text{conf} + 1) \]

\[ r(w, d) = -1, O'(\{m\}, d) = \emptyset \]
Framework

Model Value Prediction

Q-Value Network

DRL Agent

Resource Constraints

Adaptive Scheduling Algorithm

Input Data

Model Execution

Labeling State

Scheduling Policy

Reward:

\[ r(w, d) = \log(\theta_m \sum_{l_i \in O'(\{m\}, d)} l_i \cdot conf_i + 1), O'(\{m\}, d) \neq \emptyset \]
\[ r(w, d) = -1, O'(\{m\}, d) = \emptyset \]

Priority parameter

Sum of label confidence
Framework

**Reward:**

\[
r(w, d) = \begin{cases} 
\log(\theta m \sum_{l_i \in O'(\{m\}, d)} l_i \cdot \text{conf} + 1), & O'(\{m\}, d) \neq \emptyset \\
-1, & O'(\{m\}, d) = \emptyset 
\end{cases}
\]

Logarithmic smoothing
Reward:
\[ r(w,d) = \log(\theta_m \sum_{l_i \in O'(\{m\},d)} l_i \cdot \text{conf} + 1), O'(\{m\},d) \neq \emptyset \]
\[ r(w,d) = -1, O'(\{m\},d) = \emptyset \]

Punishment
• Deep Q-Network
  • input -> dense-layer -> ReLU -> (N+1)-output
  • The +1 action is an **END** action
DRL Agent

• Deep Q-Network
  • 1104-input -> 256-dense -> ReLU -> (30+1)-output
  • The +1 action is an END action
  • The reward of selecting END is 0
DRL Agent

• Deep Q-Network
  • 1104-input -> 256-dense -> ReLU -> (30+1)-output
  • The +1 action is an $END$ action
  • The reward of selecting $END$ is 0

• Training mechanisms:
  • Original DQN
  • Double DQN
  • Dueling DQN
  • Deep SARSA
Adaptive Scheduling

• An adaptive submodular function maximization problem.
• Existing approximate algorithms with performance guarantee is infeasible for our problem, since they require partial permutation on items.
• Two common constraints of computing resources are studied:

**1-D Deadline Constraint**

Algorithm 1 Scheduling under deadline constraint.

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

1: $S \leftarrow \emptyset$
2: while $B_{time} > 0$ do
3:  Filter out models that $m.time > B_{time}$
4:  $m^* \leftarrow \arg \max_{m \in M \setminus S} Q(m,d)$
5:  $S \leftarrow S \cup \{m^*\}$, $B_{time} \leftarrow B_{time} - m^*.time$
6: end while
7: return $S$

**2-D Deadline-Memory Constraints**

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 $S$

1: $S \leftarrow \emptyset$, $TimeCost \leftarrow 0$, $S_t \leftarrow \emptyset$
2: while $TimeCost < B_{time}$ do
3:  Filter out models that $m.mem > B_{mem}$
4:  $m_1^* \leftarrow \arg \max_{m \in M \setminus S} \frac{Q(m,d)}{m.time \times m.mem}$
5:  $S_t \leftarrow S_t \cup \{m_1^*\}$, $B_{time} \leftarrow TimeCost + m_1^*.time$
6:  Filter out models by temporary deadline $B_{time}^t$
7:  while $B_{mem} > 0$ do
8:  $m_2^* \leftarrow \arg \max_{m \in M \setminus S} \frac{Q(m,d)}{m.mem}$
9:  $S_t \leftarrow S_t \cup \{m_2^*\}$, $B_{mem} \leftarrow B_{mem} - m_2^*.mem$
10: end while
11: $S.append(S_t)$, Wait until model $m_3^* \in S_t$ finishes
12: $B_{mem} \leftarrow B_{mem} + m_3^*.mem$, $S_t \leftarrow S_t \setminus \{m_3^*\}$
13: end while
14: return $S$
Adaptive Scheduling

• An adaptive submodular function maximization problem.

• Two common constraints are studied:

1-D Deadline Constraint

Algorithm 1 Scheduling under deadline constraint.

**Input:** model set $M$, time budget $B_{time}$, DRL agent $Q$

**Output:** model subset $S$

1: $S \leftarrow \emptyset$

2: while $B_{time} > 0$ do

3: Filter out models that $m.time > B_{time}$

4: $m^* \leftarrow \arg \max_{m \in M \setminus S} Q(m,d)$

5: $S \leftarrow S \cup \{m^*\}$, $B_{time} \leftarrow B_{time} - m^*.time$

6: end while

7: return $S$

Based on model profiling on sampled data.
Adaptive Scheduling

• An adaptive submodular function maximization problem.
• Two common constraints are studied:

1-D Deadline Constraint

**Algorithm 1**: Scheduling under deadline constraint.

**Input**: model set $M$, time budget $B_{time}$, DRL agent $Q$

**Output**: model subset $S$

1: $S \leftarrow \emptyset$
2: while $B_{time} > 0$ do
3:   Filter out models that $m.time > B_{time}$
4:   $m^* \leftarrow \arg \max_{m \in M \setminus S} Q(m,d)$
5:   $S \leftarrow S \cup \{m^*\}$, $B_{time} \leftarrow B_{time} - m^*.time$
6: end while
7: return $S$
Adaptive Scheduling

• An adaptive submodular function maximization problem.

• Two common constraints are studied:

2-D Deadline-Memory Constraints

**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 $S$

1. $S \leftarrow []$, $TimeCost \leftarrow 0$, $S_t \leftarrow \emptyset$

2. **while** $TimeCost < B_{time}$ **do**

3. **Filter out models that** $m_{mem} > B_{mem}$

4. $m_1^* \leftarrow \arg \max_{m \in M \setminus S} \max_{Q(m,d)} m_{time} \times m_{mem}$

5. $S_t \leftarrow S_t \cup \{m_1^*\}$, $B_{time}^t \leftarrow TimeCost + m_1^*.time$

6. **Filter out models by temporary deadline** $B_{time}^t$

**Step#1:** Determine the temporary *deadline*. 

Adaptive Scheduling

• An adaptive submodular function maximization problem.

• Two common constraints are studied:

2-D Deadline-Memory Constraints

5: \[ S_t \leftarrow S_t \cup \{m_1^*\}, \quad B_{time}^t \leftarrow TimeCost + m_1^*.time \]
6: Filter out models by temporary deadline \( B_{time}^t \)
7: \[ \textbf{while} \ B_{mem} > 0 \ \textbf{do} \]
8: \[ m_2^* \leftarrow \arg \max_{m \in M \setminus S \text{ m.mem}} Q(m,d) \]
9: \[ S_t \leftarrow S_t \cup \{m_2^*\}, \quad B_{mem} \leftarrow B_{mem} - m_2^*.mem \]
10: \[ \textbf{end while} \]
11: \( S.append(S_t) \), Wait until model \( m_3^* \in S_t \) finishes
12: \[ B_{mem} \leftarrow B_{mem} + m_3^*.mem, \quad S_t \leftarrow S_t \setminus \{m_3^*\} \]
13: \[ \textbf{end while} \]
14: \[ \textbf{return} \ S \]

Step#2: Greedily fill in the memory pool.
Adaptive Scheduling

- An adaptive submodular function maximization problem.
- Two common constraints are studied:

**2-D Deadline-Memory Constraints**

```
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 $S$
1: $S \leftarrow \emptyset$, $TimeCost \leftarrow 0, S_t \leftarrow \emptyset$
2: while $TimeCost < B_{time}$ do
3: Filter out models that $m_{mem} > B_{mem}$
4: $m_1^* \leftarrow \arg\max_{m \in M \cap S} Q(m, d)$
5: $S_t \leftarrow S_t \cup \{m_1^*\}, B_{time} \leftarrow TimeCost + m_1^*.time$
6: Filter out models by temporary deadline $B_{time}'$
7: while $B_{mem} > 0$ do
8: $m_2^* \leftarrow \arg\max_{m \in M \setminus S} Q(m, d)$
9: $S_t \leftarrow S_t \cup \{m_2^*\}, B_{mem} \leftarrow B_{mem} - m_2^*.mem$
10: end while
11: $S.append(S_t)$, Wait until model $m_3^* \in S_t$ finishes
12: $B_{mem} \leftarrow B_{mem} + m_3^*.mem, S_t \leftarrow S_t\setminus\{m_3^*\}$
13: end while
14: return $S$
```
Outline

• Resource-wasting multi-model inference workloads
• Rule-based scheduler
• Learning-based scheduler
• Evaluation
• Conclusion
## Experimental Setup

### Multi-Model Image Labeling Workloads

<table>
<thead>
<tr>
<th>Task</th>
<th>Label#</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object Detection</td>
<td>80</td>
</tr>
<tr>
<td>Scene Classification</td>
<td>365</td>
</tr>
<tr>
<td>Face Detection</td>
<td>1</td>
</tr>
<tr>
<td>Face Landmark Localization</td>
<td>70</td>
</tr>
<tr>
<td>Pose Estimation</td>
<td>17</td>
</tr>
<tr>
<td>Emotion Classification</td>
<td>7</td>
</tr>
<tr>
<td>Gender Classification</td>
<td>2</td>
</tr>
<tr>
<td>Action Classification</td>
<td>40</td>
</tr>
<tr>
<td>Hand Landmark Localization</td>
<td>42</td>
</tr>
<tr>
<td>Dog Classification</td>
<td>120</td>
</tr>
<tr>
<td><strong>10 Tasks</strong></td>
<td><strong>1104 Labels</strong></td>
</tr>
</tbody>
</table>

### MSCOCO-2017

![MSCOCO-2017 Image](image1)

### MIRFLICKR-25k

![MIRFLICKR-25k Images](image2)
Evaluation: DRL Agent

Different Training Mechanisms

(a) MSCOCO 2017

(b) MirFlickr25
Evaluation: DRL Agent

(a) MSCOCO 2017
(b) MirFlickr25

Saving 48.6-51.2% execution time without loss of valuable labels.
Evaluation: DRL vs. Rules

10 Manual Rules

<table>
<thead>
<tr>
<th>Current Model Task</th>
<th>Output Label</th>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object Detection</td>
<td>person</td>
<td>$2 \times \mathcal{P}$(Pose Estimation)</td>
</tr>
<tr>
<td>Object Detection</td>
<td>person</td>
<td>$2 \times \mathcal{P}$(Gender Classification)</td>
</tr>
<tr>
<td>Object Detection</td>
<td>dog</td>
<td>$2 \times \mathcal{P}$(Dog Classification)</td>
</tr>
<tr>
<td>Face Detection</td>
<td>face</td>
<td>$2 \times \mathcal{P}$(Face Landmark Localization)</td>
</tr>
<tr>
<td>Face Detection</td>
<td>face</td>
<td>$2 \times \mathcal{P}$(Emotion Classification)</td>
</tr>
<tr>
<td>Pose Estimation</td>
<td>body keypoints</td>
<td>$2 \times \mathcal{P}$(Action Classification)</td>
</tr>
<tr>
<td>Pose Estimation</td>
<td>wrist keypoints</td>
<td>$2 \times \mathcal{P}$(Hand Landmark Localization)</td>
</tr>
<tr>
<td>Place Classification</td>
<td>indoor places</td>
<td>$0.5 \times \mathcal{P}$(Animal-Object Detection)</td>
</tr>
<tr>
<td>Place Classification</td>
<td>indoor places</td>
<td>$0.5 \times \mathcal{P}$(Sport-Action Classification)</td>
</tr>
</tbody>
</table>

Saving only 1.4% execution time when required recall is 1.0.
Evaluation: Model Priority

Adjusting priority of the face-detection model.

\[ r(w, d) = \log(\theta_m) \sum_{l_i \in O'(\{m\}, d)} l_i \text{conf} + 1), O'(\{m\}, d) \neq \emptyset \]

Priority parameter

(a) Average execution order

(b) Average time cost
Evaluation: Scheduling Algorithms

1-D Deadline Constraint

- MSCOCO 2017
- MirFlickr25

2-D Deadline-Memory Constraints

- 8GB Memory
- 12GB Memory

Boosts 188.7-309.5% recall of valuable labels under 0.5s deadline constraint.

Boosts 106.9%/52.8% recall of valuable labels under 0.8s deadline and 8/12GB memory constraints.
Outline

• Resource-wasting multi-model inference workloads
• Rule-based scheduler
• Learning-based scheduler
• Evaluation

• Conclusion
Adaptive model scheduling can improve the efficiency of multi-model inference workloads by avoiding valueless execution.
Lab for Intelligent Networking & Knowledge Engineering

12 Faculty members, 2 Post-Doc, 3 Secretary; 7 with PhD from abroad

XiangYang Li
IEEE Fellow, ACM Fellow, ACM China Co-Chair

Panlong Yang
CCF Dist Speaker
Wireless network, Mobile computing

Bei Hua
High performance computing, Edge computing

Haisheng Tan
HK, Tsinghua Post-Doc
Cloud computing, Algorithms Analysis

Yanyong Zhang
IEEE Fellow, Prof. in Rutgers, NSF Career

Xin Guo
Edge computing, Security of IoT

Nikolaos M.Freris
USA NYU A.P.
CPS, Algorithms, Distributed optimization
Machine learning

Yu Zhang
system software, Software optimization/security, Quantum software

YuBo Yan
Wireless/Passive network, IntelliSense, IoT, SDR

Lan Zhang
CCF, ACM China Doctor Thesis Award, Youqing, Qingcheng Award
Data understanding/trading, privacy protection

Hao Zhou
Japan NTII Wireless Network Resource Management

Xin He
Doc. University of Oulu
Passive network, Theories of Information and Coding

Xuerong Huang
Master in HKBU
Research assistant

Ludi Xue
Research assistant