Notes of RT-mDL
Supporting Real-Time Mixed Deep Learning Tasks on Edge Platforms

AIoT Technologies

Zijian Luo

October 1, 2022

Abstract

The authors[1] present RT-mDL, a novel framework to support mixed real-time DL tasks on edge platform with heterogeneous CPU and GPU resource.

[1] Neiwen Ling et al. "RT-MDL: Supporting Real-Time Mixed Deep Learning Tasks on Edge Platforms". In: *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*. SenSys '21. Coimbra, Portugal: Association for Computing Machinery, 2021, pp. 1–14. ISBN: 9781450390972. DOI: 10.1145/3485730.3485938. URL: https://doi.org/10.1145/3485730.3485938.

Abstract

The authors[1] present RT-mDL, a novel framework to support mixed real-time DL tasks on edge platform with heterogeneous CPU and GPU resource.

- It aim to optimize the mixed DL tasks execution to meet their diverse real-time/accuracy requirements.

[1] Neiwen Ling et al. "RT-MDL: Supporting Real-Time Mixed Deep Learning Tasks on Edge Platforms". In: *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*. SenSys '21. Coimbra, Portugal: Association for Computing Machinery, 2021, pp. 1–14. ISBN: 9781450390972. DOI: 10.1145/3485730.3485938. URL: https://doi.org/10.1145/3485730.3485938.

Abstract

The authors[1] present RT-mDL, a novel framework to support mixed real-time DL tasks on edge platform with heterogeneous CPU and GPU resource.

- It aim to optimize the mixed DL tasks execution to meet their diverse real-time/accuracy requirements.
- It employs a novel storage-bounded model scaling method to generate a series of model variants by joint model variants selection and task priority assignment.

[1] Neiwen Ling et al. "RT-MDL: Supporting Real-Time Mixed Deep Learning Tasks on Edge Platforms". In: *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*. SenSys '21. Coimbra, Portugal: Association for Computing Machinery, 2021, pp. 1–14. ISBN: 9781450390972. DOI: 10.1145/3485730.3485938. URL: https://doi.org/10.1145/3485730.3485938.

Abstract

The authors[1] present RT-mDL, a novel framework to support mixed real-time DL tasks on edge platform with heterogeneous CPU and GPU resource.

- It aim to optimize the mixed DL tasks execution to meet their diverse real-time/accuracy requirements.
- It employs a novel storage-bounded model scaling method to generate a series of model variants by joint model variants selection and task priority assignment.
- It address a new priority-based scheduler which employs a GPU packing mechanism and executes the CPU/GPU tasks independently.

[1] Neiwen Ling et al. "RT-MDL: Supporting Real-Time Mixed Deep Learning Tasks on Edge Platforms". In: *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*. SenSys '21. Coimbra, Portugal: Association for Computing Machinery, 2021, pp. 1–14. ISBN: 9781450390972. DOI: 10.1145/3485730.3485938. URL: https://doi.org/10.1145/3485730.3485938.

Abstract

The authors[1] present RT-mDL, a novel framework to support mixed real-time DL tasks on edge platform with heterogeneous CPU and GPU resource.

- It aim to optimize the mixed DL tasks execution to meet their diverse real-time/accuracy requirements.
- It employs a novel storage-bounded model scaling method to generate a series of model variants by joint model variants selection and task priority assignment.
- It address a new priority-based scheduler which employs a GPU packing mechanism and executes the CPU/GPU tasks independently.
- Its implementation can enable multi concurrent DL tasks to achieve real-time performance

[1] Neiwen Ling et al. "RT-MDL: Supporting Real-Time Mixed Deep Learning Tasks on Edge Platforms". In: *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*. SenSys '21. Coimbra, Portugal: Association for Computing Machinery, 2021, pp. 1–14. ISBN: 9781450390972. DOI: 10.1145/3485730.3485938. URL: https://doi.org/10.1145/3485730.3485938.

Motivation

DL model compressibility

Motivation

DL model compressibility

- It can be seen that almost all the DNN models exhibit a wide region of latency-accuracy trade-off.

Motivation

DL model compressibility

- It can be seen that almost all the DNN models exhibit a wide region of latency-accuracy trade-off.
- The model compressibility is highly diverse across different DNN models. Especially, VGG19 better than AlexNet.

Motivation

Real-Time Scheduling for Mixed DL tasks

Motivation

Real-Time Scheduling for Mixed DL tasks

- GPU driver will schedule the operation(Matrix multiplication) in Round-robin manner.

Motivation

Real-Time Scheduling for Mixed DL tasks

- GPU driver will schedule the operation(Matrix multiplication) in Round-robin manner.
- They choose to run 4 mixed DL tasks simultaneously under all possible settings of task priorities(Two AlexNet tasks and two VGG11 tasks).

Motivation

Real-Time Scheduling for Mixed DL tasks

- GPU driver will schedule the operation(Matrix multiplication) in Round-robin manner.
- They choose to run 4 mixed DL tasks simultaneously under all possible settings of task priorities(Two AlexNet tasks and two VGG11 tasks).
- Figure 3 shows the tasks with low priorities can be frequently blocked by the tasks with high priorities.

Motivation

DL Task CPU/GPU Utilization

Motivation

DL Task CPU/GPU Utilization

- It lead to a temporal underutilization of CPU/GPU resource

Motivation

DL Task CPU/GPU Utilization

- It lead to a temporal underutilization of CPU/GPU resource
- A DL task is occupying CPU, the GPU is left idle even when there are other lower-priority DL tasks in the task queue.

Motivation

DL Task CPU/GPU Utilization

- It lead to a temporal underutilization of CPU/GPU resource
- A DL task is occupying CPU, the GPU is left idle even when there are other lower-priority DL tasks in the task queue.
- Figure 4,5 and 6 shows that GPU is often efficiently utilized, and CPU execution time should not be neglected when running mixed DL tasks on the edge.

Motivation

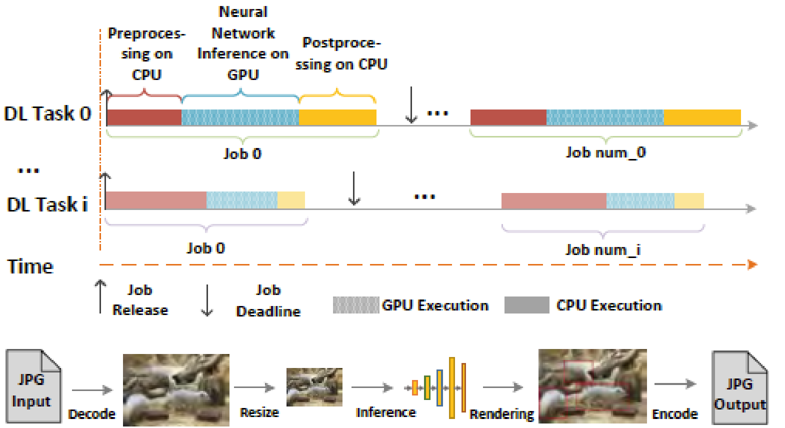

Figure 4: Timeline of DL task execution: the neural network computation is executed on GPU, and the pre-processing and post-processing are executed on CPU.

Motivation

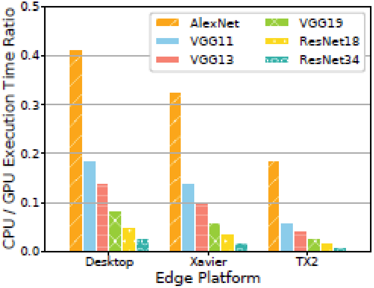

Figure 5: Execution time ratio between CPU and GPU on different platforms.

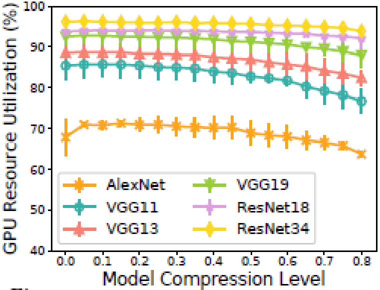

Figure 6: GPU Spatial Utilization of DNN model under different levels of model compression (on Xavier).

Key idea of RT-mDL

For example, A autonomous driving vehicle is usually equipped with several on-board cameras and Lidars to detect the traffic lights and signs.

- The DL tasks for traffic light detection typically have tight deadline as well as extremely low probability of deadline missing.

Key idea of RT-mDL

For example, A autonomous driving vehicle is usually equipped with several on-board cameras and Lidars to detect the traffic lights and signs.

- The DL tasks for traffic light detection typically have tight deadline as well as extremely low probability of deadline missing.
- but for the speech recognition tasks for voice should tolerate more relaxed deadlines and missing probabilities.

Key idea of RT-mDL

Formulation of execution strategy

$$\begin{aligned} & \min_{\mathrm{s}} \mathsf{LOSS}(\mathrm{s}) = \sum_i \frac{\mathsf{LOSS}_i(\mathrm{s})}{\mathsf{ACC}_i^{\max}} \\ & \text{s.t.} \quad \mathsf{MIS}_i(\mathrm{s}) \le \zeta_i, \quad \sum_i \sum_k \mathsf{Storage}\left(\tilde{N}_{i,k}\right) \le \overline{\mathsf{Storage}} \end{aligned} \tag{1}$$

System Architecture

- To support flexible model scaling for mixed DL tasks, RT-mDL adopts a component called storage-bounded multi-level model scaling.

System Architecture

- To support flexible model scaling for mixed DL tasks, RT-mDL adopts a component called storage-bounded multi-level model scaling.
- To ensure that all model variants can be stored on the edge platform, RT-mDL adopts a new weight sharing mechanism in multi-level model scaling to limit the total used storage of generated model variants.

System Architecture

- To support flexible model scaling for mixed DL tasks, RT-mDL adopts a component called storage-bounded multi-level model scaling.
- To ensure that all model variants can be stored on the edge platform, RT-mDL adopts a new weight sharing mechanism in multi-level model scaling to limit the total used storage of generated model variants.
- To find the best execution strategy which consists of model variant selection and priority assignment, RT-mDL adopt a new Multi-Objective Evolutionary Algorithm (MOEA).

System Architecture

- To support flexible model scaling for mixed DL tasks, RT-mDL adopts a component called storage-bounded multi-level model scaling.
- To ensure that all model variants can be stored on the edge platform, RT-mDL adopts a new weight sharing mechanism in multi-level model scaling to limit the total used storage of generated model variants.
- To find the best execution strategy which consists of model variant selection and priority assignment, RT-mDL adopt a new Multi-Objective Evolutionary Algorithm (MOEA).
- To estimate the response time, RT-mDL adopts a proxy model to estimate deadline.

System Architecture

- To support flexible model scaling for mixed DL tasks, RT-mDL adopts a component called storage-bounded multi-level model scaling.
- To ensure that all model variants can be stored on the edge platform, RT-mDL adopts a new weight sharing mechanism in multi-level model scaling to limit the total used storage of generated model variants.
- To find the best execution strategy which consists of model variant selection and priority assignment, RT-mDL adopt a new Multi-Objective Evolutionary Algorithm (MOEA).
- To estimate the response time, RT-mDL adopts a proxy model to estimate deadline.
- To support flexible real-time scheduling of mixed DL tasks, RT-mDL adopts a priority-based DL task scheduler.

System Architecture of RT-mDL

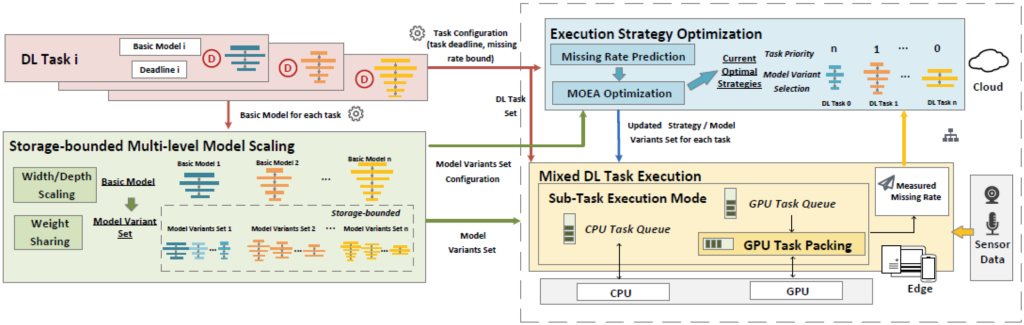

Figure 7: System Architecture of RT-mDL: Storage-bounded multi-level model scaling generates model variants for each DL task, priority-based DL task scheduler supports flexible real-time scheduling of mixed DL task, execution strategy optimizer searches for the optimal execution strategy of model variant selection and priority assignment.

Storage-bounded Multi-level Model Scaling

In order to ensure that all model variants can be stored on the resource constrained edge platform for optimization at run-time, we also propose a fine-grained weight sharing mechanism to bound the total used storage of generated model variants.

$$
\begin{aligned}
&\forall i, k \quad \min \mathcal{L}\left(\mathcal{W}_{i,k}, \tilde{N}_{i,k}\right) \\
&\text{s.t. } \text{FLOPS }\left(\tilde{N}_{i,k-1}\right) - \text{FLOPS}\left(\tilde{N}_{i,k}\right) \geq \frac{\varepsilon}{K_i}, \\
&\sum_i \text{Storage }\left(\cup_{k=0}^{K} \mathcal{W}_{i,k}\right) \leq \overline{\text{Storage}}
\end{aligned}
\tag{2}
$$

The objective is to minimize the accuracy loss of the model variants under storage constraint while their compute workloads decrease in an equal gradient, which leads to different levels of execution latency.

Storage-bounded Multi-level Model Scaling

Width and Depth Scaling:

- Multi-level model scaling can be conducted either by width scaling or depth scaling.

Storage-bounded Multi-level Model Scaling

Width and Depth Scaling:

- Multi-level model scaling can be conducted either by width scaling or depth scaling.
- Width scaling adjusts the width of DNN models, and they use filter pruning technique here to reduce the channels of each DNN layer.

Storage-bounded Multi-level Model Scaling

Width and Depth Scaling:

- Multi-level model scaling can be conducted either by width scaling or depth scaling.
- Width scaling adjusts the width of DNN models, and they use filter pruning technique here to reduce the channels of each DNN layer.
- For depth scaling, they firstly identify the repeating units of the DNN model architecture.

Storage-bounded Multi-level Model Scaling

Width and Depth Scaling:

- Multi-level model scaling can be conducted either by width scaling or depth scaling.
- Width scaling adjusts the width of DNN models, and they use filter pruning technique here to reduce the channels of each DNN layer.
- For depth scaling, they firstly identify the repeating units of the DNN model architecture.
- Multi-level model scaling can also be achieved by scaling DNN architecture in both width and depth dimensions simultaneously. However, such an approach will lead to an explosion of the number of model variants.

Fine-grained weight sharing mechanism

- Under their weight sharing mechanism, model weights $\mathcal{W}_{i,k}$ of the k-th model variant consist of its private weights $W_{i,k}$ and shared weights W_i^*, which are shared among all the model variants of DL task τ_i.

Fine-grained weight sharing mechanism

- Under their weight sharing mechanism, model weights $\mathcal{W}_{i,k}$ of the k-th model variant consist of its private weights $W_{i,k}$ and shared weights W_i^*, which are shared among all the model variants of DL task τ_i.
- Shared weights and private weights are separated based on a layer granularity as shown in Eq. (3).

$$W_i^* = \sum_{j=0}^{J_{\text{share}}} W^j, W_{i,k} = \sum_{j=J_{\text{share}}}^{J_{\text{Jall}}} W_k^j \tag{3}$$

Fine-grained weight sharing mechanism

- J_{share} is the number of layers that share weights among all model variants, and K is the number of extracted model variants from all the $K_{\max}$ generated model variants (extracted in the equal interval). The total storage usage of task τ_i can be rewritten as in Eq. (4), which reduces the storage usage of shared layers among model variants.

$$\text{Storage}\left(\cup_{k=0}^{K} W_{i,k}\right) = \text{Storage}\left(W_i^*\right) + \sum_{k=0}^{K} \text{Storage}\left(W_{i,k}\right) \tag{4}$$

Joint Model Variant Selection and Task Scheduling

Challenges:

- The deadline missing ratio for a given strategy doest not have closed-form expressions.

Joint Model Variant Selection and Task Scheduling

Challenges:

- The deadline missing ratio for a given strategy doest not have closed-form expressions.
- It can be calculated based on the response time of the task execution, but it is too pessimistic. Because the worst case of execution time of each task is longer than its actual execution time.

Joint Model Variant Selection and Task Scheduling

Challenges:

- The deadline missing ratio for a given strategy doest not have closed-form expressions.
- It can be calculated based on the response time of the task execution, but it is too pessimistic. Because the worst case of execution time of each task is longer than its actual execution time.
- The solution for each execution strategy contains contains a large space on both model variants selection and task priorities. To seacr them, it is not easy.

Joint Model Variant Selection and Task Scheduling

They tackle these challenges by adopting multi-objective optimization and performance approximation techniques. We employ a proxy model $\mathrm{f}(\cdot) = [f_1(\cdot), \ldots, f_n(\cdot)]^T$ to predict the deadline missing rate of each task for a given execution strategy.

$$f_i(\mathrm{s}) \approx \mathrm{MIS}_i(\mathrm{s}) \tag{5}$$

The Proxy model for Missing Rate Prediction

- The proxy model is based on Random Forests (RF) approach[2], although other methods such as Gaussian Process Regression (GPR) are also applicable.

[2] Md Shahriar Iqbal et al. *FlexiBO: A Decoupled Cost-Aware Multi-Objective Optimization Approach for Deep Neural Networks*. 2020. DOI: 10.48550/ARXIV.2001.06588. URL: https://arxiv.org/abs/2001.06588.

The Proxy model for Missing Rate Prediction

- The proxy model is based on Random Forests (RF) approach[2], although other methods such as Gaussian Process Regression (GPR) are also applicable.
- Actually, I am still confused on this point of proxy model and its topic of reference paper.
- Given a set S of strategies whose actual performance are measured on the edge platform, the training process can be regarded as minimizing the square error (MSE) with respect to set S

$$\min \sum_{s \in S} \left\| [f_1(s), \ldots, f_n(s)]^T - [\mathrm{MIS}_1(s), \ldots, \mathrm{MIS}_n(\mathrm{s})]^T \right\|_2 \tag{6}$$

[2] Md Shahriar Iqbal et al. *FlexiBO: A Decoupled Cost-Aware Multi-Objective Optimization Approach for Deep Neural Networks*. 2020. DOI: 10.48550/ARXIV.2001.06588. URL: https://arxiv.org/abs/2001.06588.

Priority-based DL Task Scheduler

In order to fullfil the requirement of DL task scheduler, its design contains these advantages.

- Support flexible priority assignment.

Priority-based DL Task Scheduler

In order to fullfil the requirement of DL task scheduler, its design contains these advantages.

- Support flexible priority assignment.
- Separate the CPU/GPU sub-tasks, which enables more efficient temporal utilization of CPU/GPU resource.

Priority-based DL Task Scheduler

In order to fullfil the requirement of DL task scheduler, its design contains these advantages.

- Support flexible priority assignment.
- Separate the CPU/GPU sub-tasks, which enables more efficient temporal utilization of CPU/GPU resource.
- Enable GPU packing of multiple DNN inferences, which increases the GPU spatial utilization.

Priority-based DL Task Scheduler

A DL task consists of CPU execution part for pre-/post-processing like decoding and GPU execution part for DNN model inference. Therefore, each DL task τ_i ($i \in \{1, 2, \ldots, n\}$) comprises three sequential sub-tasks $C_{i,1}$, G_i, $C_{i,2}$ where $C_{i,q}$ denotes the execution time of the q-th CPU sub-task (pre-processing for $C_{i,1}$ and post-processing for $C_{i,2}$), while G_i is the execution time of GPU sub-task. Real-world DL applications usually require periodically processing input data in real-time. Therefore, in their DL task model, each DL task τ_i is a periodic task with period T_i and a user-defined relative deadline D_i, where D_i can equal to T_i. Each task τ_i is thus defined by an array $(C_{i,1}, G_i, C_{i,2}, D_i, T_i)$.

Priority-based DL Task Scheduler

Priority-based CPU-GPU Sub-Task Scheduling

- The scheduler divides the DL task into the CPU sub-task and the GPU sub-task by Mutex and puts these sub-tasks into independent CPU and GPU task queues.

Priority-based DL Task Scheduler

Priority-based CPU-GPU Sub-Task Scheduling

- The scheduler divides the DL task into the CPU sub-task and the GPU sub-task by Mutex and puts these sub-tasks into independent CPU and GPU task queues.
- The CPU sub-task from low-priority task can be executed when the high-priority task finishes its CPU sub-task and starts its GPU sub-task.

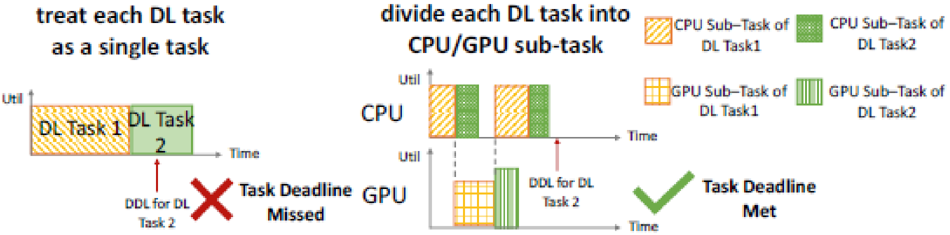

Figure 10: Improve the efficiency of priority-based scheduling for mixed DL

GPU Task Packing

In order to improve the GPU spatial utilization for model inference, the scheduler also includes a priority-guided GPU packing mechanism, which enables parallel execution of DL tasks under the guarantee of priority.

- GPU has more parallel execution units (Arithmetic Logic Unit) than CPU.
- Model inference cannot fully utilize all the execution units on GPU.

GPU Task Packing

- To achieve GPU task packing, the scheduler will employ two GPU streams (High-priority Stream (HS) and Low-priority Stream (LS)) for DNN inference (GPU sub-task) in each DL task.

GPU Task Packing

- To achieve GPU task packing, the scheduler will employ two GPU streams (High-priority Stream (HS) and Low-priority Stream (LS)) for DNN inference (GPU sub-task) in each DL task.
- After the GPU sub-task at the head of priority queue receives the starting signal, it will be pushed into the stream of high priority. The scheduler then search for another sub-task that can be executed in parallel with the current GPU sub-task and put it into the low priority stream for execution.

GPU Task Packing

The specific packing rule in our scheduler is

- When HS is empty, the task with the highest priority in GPU task queue is submitted to HS only if it has higher priority than the current task processed in LS.

GPU Task Packing

The specific packing rule in our scheduler is

- When HS is empty, the task with the highest priority in GPU task queue is submitted to HS only if it has higher priority than the current task processed in LS.
- When LS is empty, the packing algorithm finds task $\tau_{i'}(G)$ via looking ahead in the GPU task queue until the packing condition holds for task $\tau_{i'}(G)$ and the other task $\tau_i(G)$ in HS.

GPU Task Packing

The specific packing rule in our scheduler is

- When HS is empty, the task with the highest priority in GPU task queue is submitted to HS only if it has higher priority than the current task processed in LS.
- When LS is empty, the packing algorithm finds task $\tau_{i'}(G)$ via looking ahead in the GPU task queue until the packing condition holds for task $\tau_{i'}(G)$ and the other task $\tau_i(G)$ in HS.
- The packing condition is formulated as follows, Priority $_{i'} <$ Priority $_i$, Utilization $_{i'}+$ Utilization $_i \leq \theta$, and θ is a controllable threshold for the maximum allowed GPU spatial utilization rate.

My idea in this paper

Advantages of RT-mDL:

- It supports running mixed real-time DL tasks on edge platforms through joint optimization of DNN model scaling and real-time scheduling

My idea in this paper

Advantages of RT-mDL:

- It supports running mixed real-time DL tasks on edge platforms through joint optimization of DNN model scaling and real-time scheduling
- It divides the DL tasks into CPU sub-task and GPU sub-task, which can increase the utilization of all components and throughput of DL tasks in edge platform.

My idea in this paper

Advantages of RT-mDL:

- It supports running mixed real-time DL tasks on edge platforms through joint optimization of DNN model scaling and real-time scheduling
- It divides the DL tasks into CPU sub-task and GPU sub-task, which can increase the utilization of all components and throughput of DL tasks in edge platform.
- It really provide a solution to model scaling to reduce the computation cost in storage.

My idea in this paper

Disadvantages of RT-mDL:

- The methods of model scaling should be considered more.

My idea in this paper

Disadvantages of RT-mDL:

- The methods of model scaling should be considered more.
- Proxy model is still confused.

My idea in this paper

Disadvantages of RT-mDL:

- The methods of model scaling should be considered more.
- Proxy model is still confused.
- More and more advanced DNN models have been deployed in future edge platform, but the author still use the traditional method, such as AlexNet.