COMP4436 Indivdual Project

Towards Optimal Dispatch in AIoT Inference

WANG Yuqi
Computing, The Hong Kong Polytechnic University, HK

Abstract

This project implements a full AIoT inference pipeline from arrival patterns to edge dynamic model loading and cloud offloading. In addition to static and queue-adaptive baselines, schedulers based on **Lyapunov drift-plus-penalty** and **dynamic programming** are further implemented to facilitate comprehensive analysis of accuracy-latency trade-offs. Extensive experiments are conducted across six variants of LeNet and ResNet-152, two datasets (MNIST and CIFAR-10), as well as three different arrival patterns (uniform, poisson, and gamma). Results reveal that queue-adaptive heurstics improve throughput but at the cost of accuracy; Lyapunov preserves accuracy while improving throughput; DP planner further pushes the accuracy-latency pareto frontier via optimal lookaheads given partial information.

1 Introduction

In AIoT inference systems, data rates often exceeds the pipeline's current processing capacity, necessitating joint decision over admission control (which to process), model selection (what to use), and cloud offloads (where to place). These decisions are made online as samples arrive and inherently requires reasoning over the joint decision space. However, naive approach such as fixed dispatchers ignore system state, achieving high accuracy but low throughput. Queue-adaptive methods balance this by heuristically swtiching to lighter models based on queue pressure; but this sacrifices accuracy as it never reasoned about future consequences. Without addressing these limitations, the accuracy-latency landscape cannot be sufficiently explored. In light of this, two additional optimization strategies are developed:

- 1) A **One-Step Optimal** dispatcher via Lyapunov drift-plus-penalty (DPP).
- 2) A **Multi-Step Optimal** dispatcher via time-quantized Dynamic Programming (DP).

The One-Step Optimal Lyapunov DPP dispatcher scores each device-model action against the current queue by jointly optimizing over the entire model-device action space and replacing uniform tiers with a single auto-calibrated weight. The Multi-Step Optimal dispatcher replaces this greedy policy to a finite-horizon Bellman recursion over the entire queue, allowing for more calibrated value estimates and deliberated action selection; this achieves near-optimal accuracy-latency trade-offs up to some time quantization error, given zero knowledge of future arrivals.

Empirically, the LeNet-5 + MNIST does not create meaningful accuracy and latency spread. Therefore, to ensure non-trivial evaluations, the system implements ResNet-152 on CIFAR-10[1], with six compressed variants from quantization and pruning, as well as three different data arrival patterns in increasing difficulties (i.e., uniform, poisson, gamma distributions) under sustained overloads. Results demonstrates that the multi-step optimal planner clearly achieves the highest throughput, loweset latency, and near full model accuracy simutaneously, followed by one-step optimal dispatcher.

[1]The pipeline still supports LeNet-5 on MNIST to meet this project's requirements.

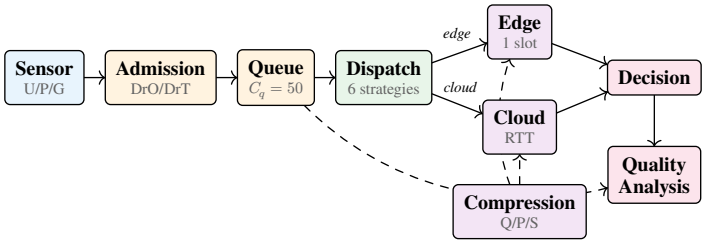

Table 1: Profile of actual trained model variants. **Lat.** means the average per-sample inference latency on edge and cloud (denoted CPU/CUDA). **Acc.** is computed as the model's test set accuracy. **MACs** are counted via `thop`. All model except Full and Quant INT8 undergo finetune post-compression. LeNet-5 + MNIST shows little accuracy and latency spread compared to ResNet-152 + CIFAR-10.

	LeNet-5 + MNIST			ResNet-152 + CIFAR-10		
Variant	**Acc.**	**Lat. (ms)**	**MACs**	**Acc.**	**Lat. (ms)**	**MACs**
Full	99.3%	0.5/0.2	44.3 M	**89.1%**	26.5/3.9	3750.8 M
Quant INT8	99.4%	**0.1**/0.2	44.3 M	87.7%	21.2/**3.6**	3750.8 M
Pruned 30%	**99.5%**	0.5/0.2	44.3 M	88.9%	26.8/3.9	3750.8 M
Pruned 60%	99.4%	0.5/0.2	44.3 M	88.3%	24.7/3.8	3750.8 M
Struct 30%	99.4%	0.4/0.2	21.8 M	86.2%	19.3/4.3	1835.1 M
Struct 50%	99.2%	0.2/**0.1**	**11.4 M**	66.6%	**11.2**/3.9	**945.3 M**

2 System Design

Admission Control. Following the project specs, admission control is implemented as a DropOld and DropTail admission policy. Given a bounded queue with capacity C_q and length Q, DropOld evicts earliest queued samples and admit new arrivals when $Q = C_q$, whereas DropTail simply rejects any new incoming samples until $Q < C_q$. Samples whose deadlines were already expired ($t \geq t_{\text{arr}} + t_{\text{ddl}}$), whether on arrival or within queue, are dropped immediately.

Inference Control. Inference control consists of a dynamic model loader and a cloud offloader; fixed and queue-adaptive heuristic schedulers treat them as separate modules, whereas One-Step and Multi-Step Optimal dispatchers treat them as joint decisions. The edge device is restricted to holds a single model variant at a time (simulate limited memory) and a load delay penalty $\delta_{\text{load}} = 10\text{ms}$ is is incurred when switching models. The cloud node is assumed to hold all model variants at the same time (greater memory capacity) but incurs a network round-trip latency of $\delta_{\text{rtt}} = 100 \pm 20\text{ms}$ (Gaussian jitter). Six dispatchers are evaluated: **FixedEdge** always dispatches to the edge using the full model; **FixedOffload** adds cloud fallback for parallelism; **AdaptQueue** adaptively trades accuracy for speed based on queue pressure Q/C_q; **AdaptDeadline** extends AdaptQueue with hard deadline feasibility checks: if no model variant can finish before the deadline, drop the sample. **One-Step Optimal** and **Multi-Step Optimal** will be discussed in-depth in Section 2.1 and Section 2.2.

Model Compression. Six model variants of ResNet-152 are created via dynamic INT8 quantization, unstructured L1 pruning (30% and 60% sparisty), and structured channel-wise pruning (30% and 50% channel pruned). The INT8 variants are obtained through static post-training quantization: it starts by exporting FP32 full model to ONNX, calibrate on 256 samples, and finally quantized to INT8 weights and activations. Inference is done through ONNX Runtime (edge and cloud are simulated via CPU and TensorRT respectively) rather than PyTorch due to compatibility issues. The same is applied to the LeNet-5. As shown in Table 1, LeNet-5 + MNIST shows minimal latency and accuracy spread, making the task trivially easy; whereas, ResNet-152 + CIFAR-10 migitates this.

Quality Analysis. To systematically assess model performance, a wide range of summary statistics are computed. This includes end-to-end **latency** (avg and p95/p99 tails), accuracy (**Acc**), deadline miss ratio (**DDL%**), queue length (***Q***), and Age of Information (**AoI**). AoI is computed as the time-averaged oldness of the most recently completed decision:

$$\text{AoI} = \frac{1}{T - T_0} \int_{T_0}^{T} (t - g^*(t))\, \partial t \tag{1}$$

where $g^*(t)$ is the generation time of the freshest completed sample at time t (in case more than one sample completed at t). T_0 is the time in which the first sample is completed processing, and T is the full simulation duration. Lower AoI indicates better timeliness of the model's decision.

2.1 One-Step Optimal

The One-Step Optimal is a Lyapunov strategy that frames each dipatch decision as a DPP optimization over the joint decision space of model m and device $d \in \{\text{edge}, \text{cloud}\}$. The standard DPP principle selects the action that minimizes:

$$\Delta L(t) + V \cdot p(t) \tag{2}$$

where $\Delta L(t)$ is the **one-step Lyapunov drift** (change in queue instability), $p(t)$ is the penalty (in our case, prediction accuracy), and $V > 0$ controls this accuracy-stability tradeoff.

As discussed previously, my pipeline is implemented as a discrete-event based simulation (i.e., clock advances event to event, rather than by ticks). Each decision thus occupies a variable-length *service interval* that is equal to the service time $\tau_{m,d}$ of the chosen model-device action pair. Let $L(Q) = \frac{1}{2}Q^2$ be the Lyapunov function and let A be the mean sample arrival rate. Then, over one service interval, $A \cdot \tau_{m,d}$ new samples are expected to arrive while one sample is being processed. Therefore, the queue length after a sample is processed with (m, d) is $Q^+ \approx Q + A\tau_{m,d} - 1$. Setting the penalty term to negative accuracy $p(t) = -a_m$ then expand the equation gives:

$$\begin{aligned}
\min_{m,\ell}[\Delta L + V \cdot p] &= \min_{m,\ell}[L(Q^+) - L(Q) - Va_m] \\
&= \min_{m,\ell}\Big[\tfrac{1}{2}\big(Q + A\tau_{m,d} - 1\big)^2 - \tfrac{1}{2}Q^2 - Va_m\Big] \\
&\le \min_{m,\ell}\big[AQ\tau_{m,\ell} - Va_m\big] + B \\
&\equiv \max_{m,\ell} \underbrace{Va_m - Q\tau_{m,\ell}}_{S(m,\ell)}
\end{aligned} \tag{3}$$

where $B = \frac{1}{2}(A\tau_{\max} - 1)^2 - Q$ upperbounds $\frac{1}{2}\big(A\tau_{m,d} - 1\big)^2 - Q$, and thus reduced to a constant. The final equivalence absorbs the arrival rate A into V by rescaling $V \leftarrow V/A$.

Auto-tuned V. Unlike traditional Lyapunov framework however, the V in this pipeline is auto-tuned by sorting Pareto frontier model variants and compute per-unit latency savings $p_i = (a_i - a_{i+1})/(\tau_i - \tau_{i+1})$. Since the model switches at $Va_i - Q\tau_i = Va_{i+1} - Q\tau_{i+1}$, rearranging:

$$Va_i - Q\tau_i = Va_{i+1} - Q\tau_{i+1} = V \cdot \frac{a_i - a_{i+1}}{\tau_i - \tau_{i+1}} = V \cdot p_i \tag{4}$$

Setting $V = \big(C_q/2\big)/\ \text{median}(\{p_i\})$ thus provides an optimal balance.

2.2 Multi-Step Optimal

State Space. The state is $\boldsymbol{s} = (i, t, t_e, m_e, t_c)$, where i is next sample index, t current time tick, t_e edge availability tick, loaded edge model m_e, and finally cloud availablity tick t_c. Cloud model m_c is not needed as cloud is assumed to hold all model variants simutaneously with no model-switching latency penalty. The recursion maximizes the following cumulative reward:

$$V(\boldsymbol{s}) = \max_{a \in \mathcal{A}(\boldsymbol{s})} [r(a) + V(\boldsymbol{s}')] \tag{5}$$

where $\mathcal{A}(s)$ contains four action types:
1. **Drop** head sample: reward $r = 0$, advance to sample $i + 1$.
2. **Wait** for next device to free: reard $r = 0$, time advances.

3. **Dispatch Edge**: reward $r = u(m) - \lambda\tau_{m,e}$, edge busy till $t + \tau_{m,e}$, loaded model updated to m.
4. **Dispatch Cloud**: reward $r = u(m) - \lambda\tau_{m,c}$, cloud busy till $t + \tau_{m,c}$.

The penalty λ slightly bias towards faster model to prevent greedily maximizing cumulative accuracy. Intuitively, λ moves the model along the Pareto frontier, achieving different optimality trade-offs.

3 Experiments

Table 2: Dispatcher results across three arrival patterns ($n = 5$). FxE: FixedEdge, FxO: FixedOffload (reference baselines, always use full model); AdQ: AdaptQueue, AdD: AdaptDeadline, **1SO**: One-Step Optimal, **MSO**: Multi-Step Optimal. **Bold** marks best and underline second-best. DR: throughput (s^{-1}); Lat., AoI, P95, P99: latency in milliseconds. Miss, Acc.: ratio in percentage (%).

	Uniform						**Poisson**						**Gamma ($\alpha = 0.05$)**					
Metrics	FxE	FxO	AdQ	AdD	**1SO**	**MSO**	FxE	FxO	AdQ	AdD	**1SO**	**MSO**	FxE	FxO	AdQ	AdD	**1SO**	**MSO**
DR ↑	38.4	48.0	35.5	12.3	38.5	**55.7**	38.2	47.9	23.1	30.9	39.2	**64.0**	33.4	40.9	15.8	19.9	38.6	**42.3**
Acc. ↑	89.0	89.1	87.3	**89.3**	88.5	88.3	89.2	89.1	87.7	88.3	**89.3**	87.7	88.5	88.9	82.9	87.6	**89.0**	88.2
Lat. ↓	367	380	395	300	341	**252**	361	372	420	263	333	**220**	248	251	344	245	224	**176**
AoI ↓	381	377	423	337	353	**252**	374	369	452	275	345	**218**	277	265	388	295	252	**211**
P95 ↓	378	462	556	385	351	**321**	377	457	612	365	351	**305**	370	398	584	348	345	**298**
P99 ↓	381	490	617	403	354	**354**	381	486	621	390	355	**343**	377	460	615	375	359	**325**
Miss ↓	99.7	97.8	96.8	29.8	6.9	**1.4**	91.4	86.9	95.7	9.7	6.4	**0.7**	59.0	53.0	87.1	4.7	2.2	**0.2**

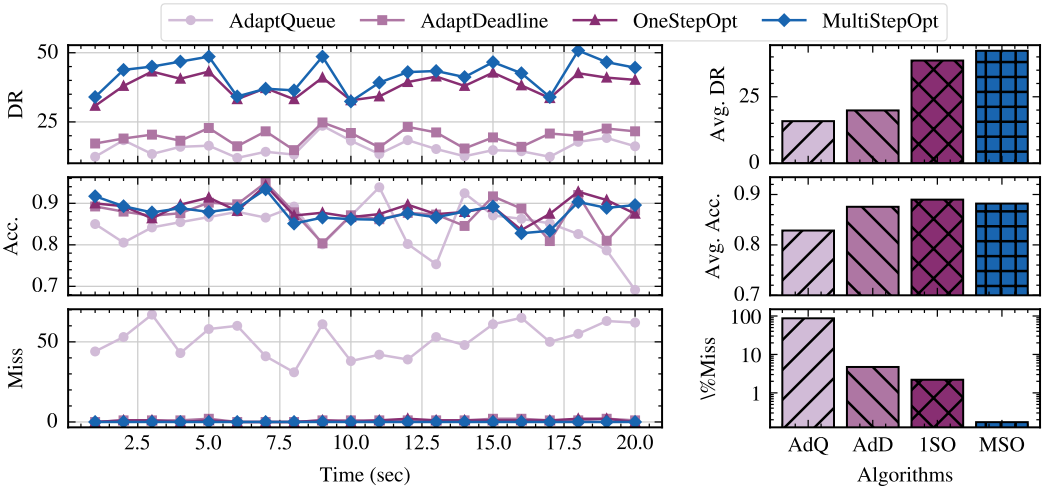

Figure 2: Visualization of Data Rate, Inference Accuracy and Deadline Miss rates across the four dispatchers, throughout the simulation. One-Step Optimal and Multi-Step Optimal clearly excells.

Setup. Results report six dispatchers evaluated on ResNet-152 + CIFAR-10 with six model variants across two devices to dynamically load/offload from, as described in Table 1. LeNet-5 + MNIST results are also included for completenesss, although they exhibit limited spread, and thus serve as reference only. Each configuration runs for 20 secs of simulated time with DropOld admission, queue capacity $C_q = 50$, and arrival rate $A = 80/s$. Results in Table 2 are averaged over $n = 5$ independent runs. The baselines used are discussed in Section 2, Inference Control.

Inference Control Results. Results clearly indicates the superiority of One-Step and Multi-Step Optimal dispatchers. Notably, Multi-Step Optimal tops in every single metrics across the three arrival patterns except for Accuracy. However, the Accuracy metric needs to be assessed with scrutiny. Specifically, fixed and adaptive methods outperforms MSO in Uniform and Poisson but at a significant cost of either extremely low throughput (DR = 12.3 for AdaptDeadline in Uniform, compared to **55.7** for MSO) or extremely high deadline miss rate (99.7% for FxE compared to **1.4%** for MSO in Uniform). Further, in the more challenging Gamma arrival pattern, MSO and ISO outperforms again.

Table 3: Admission control ablation under Gamma arrival ($A = 80$). DrO: DropOld; DrT: DropTail. **Bold** cells marks the better admission policy. DR: throughput (s^{-1}); Acc.: %; Lat.: ms.

	DR↑						Acc.↑						Lat.↓					
Policy	FxE	FxO	AdQ	AdD	1SO	MSO	FxE	FxO	AdQ	AdD	1SO	MSO	FxE	FxO	AdQ	AdD	1SO	MSO
DrO	**33.4**	**40.9**	**15.8**	19.9	**38.6**	**42.3**	**88.5**	**88.9**	82.9	**87.6**	**89.0**	**88.2**	**248**	**251**	**344**	245	**224**	**176**
DrT	33.3	40.9	14.6	19.9	38.4	42.1	88.4	88.9	**85.5**	87.2	88.8	87.8	255	258	357	**244**	229	178

Admission Control Results Table 3 ablates admission control choices. As shown, the DropOld admission policy exhibits a marginally better performance than DropTail, across three different metrices (data rate, accuracy, and average latency) and all six dispatchers.

Table 4: Dispatch results across three arrival patterns. Same as Table 2 but with LeNet-5 + MNIST.

	Uniform						Poisson						Gamma ($\alpha = 0.05$)					
Metrics	FxE	FxO	AdQ	AdD	1SO	MSO	FxE	FxO	AdQ	AdD	1SO	MSO	FxE	FxO	AdQ	AdD	1SO	MSO
DR↑	80.4	80.4	23.0	75.7	**80.4**	80.4	80.5	80.5	21.2	79.7	**80.5**	80.5	81.7	81.7	23.2	36.8	**82.1**	81.5
Acc.↑	99.0	99.0	**99.2**	99.1	99.2	99.2	99.0	99.0	99.1	99.2	**99.2**	99.2	99.0	99.0	99.1	**99.2**	99.2	99.2
Lat.↓	4.00	5.00	408	7.00	**4.00**	4.00	4.00	6.00	413	**5.00**	5.00	5.00	6.00	12.0	235	112	11.0	**9.00**
AoI↓	9.00	11.0	450	33.0	**10.0**	10.0	13.0	15.0	453	16.0	**14.0**	14.0	119	120	349	225	**119**	119
%Miss↓	0.0	0.0	88.5	0.4	**0.0**	**0.0**	0.0	0.0	90.6	0.1	0.0	**0.0**	0.0	0.0	74.4	1.7	0.0	**0.0**

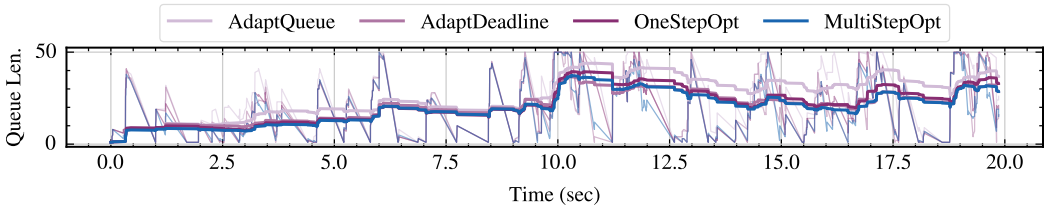

Figure 3: **Queue Length.** Visualization of queue length Q over time. The fainter lines are the real-time queue length, whereas the deeper colored line are their exponential moving averages (EMAs). One-Step and Multi-Step optimal maintains an overall shorter length compared to adaptive methods.

4 Accuracy-Latency Tradeoff

The image on the right demonstrates the accuracy-latency tradeoff of dispatchers and admission controls. The □ and △ sign denotes the best performing combination of admission strategy and dispatcher (i.e., DropOld + Multi-Step Optimal and DropOld + One-Step Optimal); collectively, they form the Pareto frontier (shown by the line connecting the two).

The different colors denotes different dispatchers. As illustrated, MSO and 1SO are top performers, followed by FixedEdge (FxE) and FixedOffload (FxO). FxE and FxO represents the "high accuracy" extreme of the accuracy-latency tradeoff, since they always dispatch samples to the most accurate model disregarding the queue pressure. AdD, on the other hand, trades accuracy for slightly better latency, but is no where close to matching the 1SO and MSO dispatchers.