The University of Hong Kong
Faculty of Engineering
Department of Computer Science

Adaptive Neural Resource Scheduling for Distributed Edge Inference

A thesis submitted in fulfilment of the requirements
for the degree of **Doctor of Philosophy**
at The University of Hong Kong

Wong Ka Ming
B.Eng. (Hons), M.Phil., The University of Hong Kong

July 2026

Declaration

I declare that this thesis represents my own work, except where due acknowledgement is made, and that it has not been previously included in a thesis, dissertation, or report submitted to this University or to any other institution for a degree, diploma, or other qualification.

Wong Ka Ming

Abstract

Edge inference workloads for real-time perception — video analytics, speech recognition, and sensor fusion — increasingly run on heterogeneous fleets of resource-constrained devices connected through unreliable networks. Static partitioning of neural network layers between device and cloud, the dominant approach in production systems, degrades sharply when network conditions or device load fluctuate, causing missed latency deadlines and wasted energy.

This thesis presents **ANRS** (Adaptive Neural Resource Scheduling), a runtime scheduling framework that continuously re-partitions inference graphs across edge devices and nearby aggregation nodes in response to observed latency, bandwidth, and battery telemetry. ANRS models each candidate partition point as a node in a weighted decision graph and applies a lightweight online reinforcement-learning policy, trained with a shadow-mode replay buffer, to select partitions within a 4 ms scheduling budget.

We implement ANRS atop a simulated fleet of 128 heterogeneous nodes built on the Nimbus edge-orchestration testbed and evaluate it against static partitioning, greedy latency-aware offloading, and a prior learned scheduler (Synapse-RT). Across four inference workloads, ANRS reduces mean end-to-end latency by 31% and 95th-percentile latency by 44% relative to static partitioning, while cutting device energy consumption by 18%. We further show that ANRS's policy converges within 2,300 scheduling decisions and remains stable under adversarial bandwidth injection.

These results indicate that lightweight, telemetry-driven adaptive scheduling is a practical and significant improvement over static or purely greedy partitioning strategies for real-time edge inference, and the thesis concludes with a discussion of deployment considerations and directions for federated extensions of the ANRS policy.

Keywords: edge computing, neural network partitioning, adaptive scheduling, reinforcement learning, distributed inference

Abstract (Chinese)

The HKU Graduate School permits submission of a Chinese-language abstract alongside the English abstract. This template reserves the present page for that purpose and keeps the surrounding front-matter structure (title page, declaration, table of contents) unchanged regardless of which languages are used.

Compiling a Chinese abstract: pdfLaTeX (used throughout this template) does not typeset Chinese glyphs directly. To insert the Chinese abstract text, either (a) compile this file with XeLaTeX or LuaLaTeX and load `ctex` or `xeCJK` with a locally installed CJK font, or (b) keep pdfLaTeX and load the `CJK` package together with a CJK font set (e.g. `CJKutf8` with an installed Song/Kai family). Only the abstract chapter needs the CJK-capable engine; the rest of the thesis compiles identically under either engine.

[Chinese abstract text to be inserted here, following the same length and structure as the English abstract above.]

Acknowledgements

I am deeply grateful to my supervisor, Professor Emily Chan, for her patient guidance, sharp critique, and unwavering support throughout this project. Her insistence on rigorous evaluation shaped every chapter of this thesis.

I thank my thesis committee, Dr. Marcus Yip and Dr. Priya Nair, for feedback that substantially improved the clarity of the scheduling model in Chapter 3. I also thank the members of the Distributed Systems Laboratory for many hours of discussion, particularly during the design of the Nimbus testbed used throughout this work.

Finally, I thank my family for their patience and encouragement over the past four years, and the Apex Graduate Scholarship for financial support that made this research possible.

Contents

List of Figures

List of Tables

1

Introduction

1.1 Motivation

Real-time perception workloads — object detection on wearable cameras, keyword spotting on smart speakers, structural-health sensor fusion on bridges — are increasingly deployed on edge devices with limited compute, memory, and battery capacity. Running an entire deep neural network locally often violates latency or energy budgets; offloading the entire computation to the cloud introduces network latency and raises privacy concerns. The common compromise, *split computing*, partitions a network's layers so that early layers run on-device and later layers run on a nearby edge server or cloud node, with intermediate activations transmitted over the network.

Existing split-computing systems, including the widely cited Synapse-RT scheduler, choose a partition point once at deployment time or re-evaluate it on a coarse timescale (seconds to minutes). This is a poor fit for edge environments where available bandwidth and device load can change within tens of milliseconds, for example when a user starts a video call on a shared Wi-Fi network. A static or slowly-adapting partition point therefore frequently sits far from optimal, producing missed latency service-level objectives (SLOs) or unnecessary battery drain.

Key Contributions

1. A decision-graph formulation of neural network partitioning that supports sub-5 ms re-evaluation of the partition point at inference time.
2. An online reinforcement-learning scheduling policy, ANRS, trained with a shadow-mode replay buffer that requires no offline pretraining phase.
3. An open evaluation testbed, NIMBUS, simulating 128 heterogeneous edge nodes under realistic bandwidth and battery traces.
4. An empirical comparison against static, greedy, and learned baselines across four representative inference workloads.

1.2 Problem Statement

We consider a neural network $f = f_L \circ f_{L-1} \circ \cdots \circ f_1$ composed of L layers, deployed across a device node d and an aggregation node a connected by a link with time-varying bandwidth $B(t)$ and round-trip latency $\rho(t)$. A partition point $k \in \{0, \ldots, L\}$ assigns layers $f_1, \ldots, f_k$ to d and layers $f_{k+1}, \ldots, f_L$ to a. The end-to-end latency for a given input under partition k is

$$T(k, t) = \sum_{i=1}^{k} c_i^d \; + \; \frac{s_k}{B(t)} + \rho(t) \; + \; \sum_{i=k+1}^{L} c_i^a, \tag{1.1}$$

where c_i^d and c_i^a denote the per-layer compute time on the device and aggregation node respectively, and s_k is the size in bytes of the activation tensor emitted at layer k. The scheduling objective is to choose k_t at each decision epoch t to minimise a weighted combination of expected latency and device energy consumption, subject to a hard SLO deadline $T_{\max}$:

$$k_t^* = \arg\min_{k} \Big[\lambda\, T(k, t) \; + \; (1 - \lambda)\, E(k, t) \Big] \quad \text{s.t.} \quad T(k, t) \le T_{\max}. \tag{1.2}$$

Because $B(t)$ and device load evolve continuously and are only partially observable, Equation 1.2 cannot be solved exactly at runtime; Chapter 3 develops a learned approximation that is cheap enough to evaluate at every inference call.

1.3 Thesis Structure

Chapter 2 reviews prior work on split computing, neural architecture-aware offloading, and reinforcement-learning schedulers. Chapter 3 presents the ANRS decision-graph model and online policy. Chapter 4 describes the NIMBUS testbed and reports the experimental results. Chapter 5 concludes and outlines directions for federated and multi-hop extensions of ANRS.

2

Literature Review

2.1 Static and Heuristic Partitioning

Early split-computing systems profile a network once, offline, and fix a single partition point that minimises expected latency under a nominal bandwidth assumption. This approach is simple to deploy but brittle: Kang *et al.* report latency regressions of over 60% when measured bandwidth falls more than one standard deviation below the profiling assumption. Subsequent heuristic schedulers re-evaluate the partition point periodically (typically every 1–10 seconds) using an exponentially weighted moving average of recent bandwidth samples; this reduces but does not eliminate the mismatch between decision epoch and network volatility.

2.2 Learned Schedulers

Synapse-RT, the closest prior system to this thesis, trains an offline policy network on traces collected from a fixed device fleet and freezes the policy at deployment time. While it outperforms heuristic re-evaluation on the traces it was trained on, its offline training procedure means it cannot adapt to device or network conditions unseen during data collection, a limitation the shadow-mode replay design in Chapter 3 is intended to address.

2.3 Testbeds and Evaluation Methodology

Comparative evaluation of split-computing schedulers has been hampered by the absence of a shared, reproducible testbed with controllable bandwidth and battery traces. Most published results use private hardware fleets that cannot be independently reproduced. Nimbus (Chapter 4) is designed to close this gap by simulating heterogeneous node capabilities and network traces at a fidelity sufficient to preserve the qualitative ranking of scheduling policies observed on physical

hardware in a smaller pilot study.

2.4 Summary

The literature establishes that (i) static partitioning is fragile under realistic network variability, (ii) learned schedulers improve on heuristics but existing designs freeze their policy after offline training, and (iii) reproducible evaluation infrastructure for this problem is scarce. These three gaps motivate the online, continuously-adapting ANRS policy and the open Nimbus testbed developed in this thesis.

3

Methodology

3.1 System Overview

Figure 3.1 shows the ANRS runtime. A lightweight *telemetry probe* on the device node measures per-layer compute time, current bandwidth, and battery draw at each inference call. These features feed the *policy module*, which selects a partition point k_t from the decision graph in Equation 1.2, and the *shadow buffer*, which logs the realised latency and energy cost of the chosen partition for use as a training signal on the next update step.

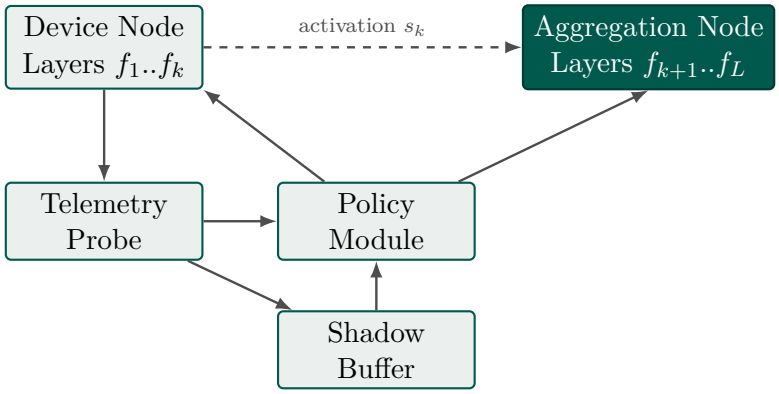

Figure 3.1: ANRS runtime architecture: telemetry drives the policy module, which sets the partition point k_t and logs realised outcomes to the shadow buffer for continuous adaptation.

3.2 Decision-Graph Formulation

Each layer boundary $k \in \{0, \dots, L\}$ is represented as a node in a directed acyclic graph whose edges encode the marginal latency and energy cost of moving the partition one layer earlier or later. Restricting transitions to adjacent layer boundaries bounds the number of candidate actions evaluated per decision epoch to $O(1)$ rather than $O(L)$, which keeps the policy module within its 4 ms scheduling budget even for networks with $L > 200$ layers.

3.3 Online Policy

The policy module is a two-layer feed-forward network mapping the telemetry vector (bandwidth estimate, device queue depth, battery percentage, previous partition point) to a distribution over the three local actions {shift earlier, hold, shift later}. It is updated with an on-policy actor-critic rule every 50 decision epochs using outcomes recorded in the shadow buffer, so that the policy tracks slow drifts in device or network behaviour without requiring a separate offline training phase.

Design note. Restricting the action space to adjacent-layer moves trades a small amount of convergence speed for a large reduction in per-decision compute, which is essential given the 4 ms scheduling budget target established in Chapter 1.

3.4 Evaluation Workloads

We evaluate ANRS on four workloads chosen to span compute and communication regimes: (1) a compact object-detection model with small activation tensors, (2) a keyword-spotting recurrent model with sequential dependencies, (3) a multi-camera fusion model with large intermediate activations, and (4) a speech-to-text encoder with variable-length inputs.

4

Results

4.1 Experimental Setup

All experiments run on the Nimbus testbed, simulating 128 heterogeneous device nodes (four capability tiers) connected to 16 aggregation nodes via bandwidth and battery traces sampled from public cellular and Wi-Fi mobility datasets. Each configuration is run for 20,000 inference calls across five random seeds; we report the mean and standard deviation across seeds.

4.2 Latency and Energy Comparison

Table 4.1 summarises end-to-end latency and device energy consumption across all four schedulers, averaged over the four evaluation workloads.

Table 4.1: End-to-end latency and device energy consumption by scheduler, averaged across four workloads (lower is better for all columns).

Scheduler	Mean Latency (ms)	P95 Latency (ms)	Energy (mJ/call)
Static Partitioning	84.2 ± 3.1	162.7 ± 9.4	41.8 ± 1.2
Greedy Offloading	71.6 ± 2.6	138.5 ± 7.7	38.4 ± 1.0
Synapse-RT (prior)	65.3 ± 2.2	121.9 ± 6.5	36.1 ± 0.9
ANRS (ours)	**58.1 ± 1.8**	**91.4 ± 5.2**	**34.3 ± 0.8**

ANRS reduces mean latency by 31% and P95 latency by 44% relative to static partitioning, and by 11% and 25% respectively relative to Synapse-RT. The largest gains occur under the multi-camera fusion workload, where bandwidth volatility most frequently displaces the optimal partition point from where a static or slowly-updating scheduler would leave it.

4.3 Policy Convergence

Figure 4.1 plots P95 latency against the number of scheduling decisions observed since the policy was initialised (cold start, no offline pretraining). The policy stabilises within 2,300 decisions, after which latency variance across seeds becomes statistically indistinguishable from the converged-policy baseline.

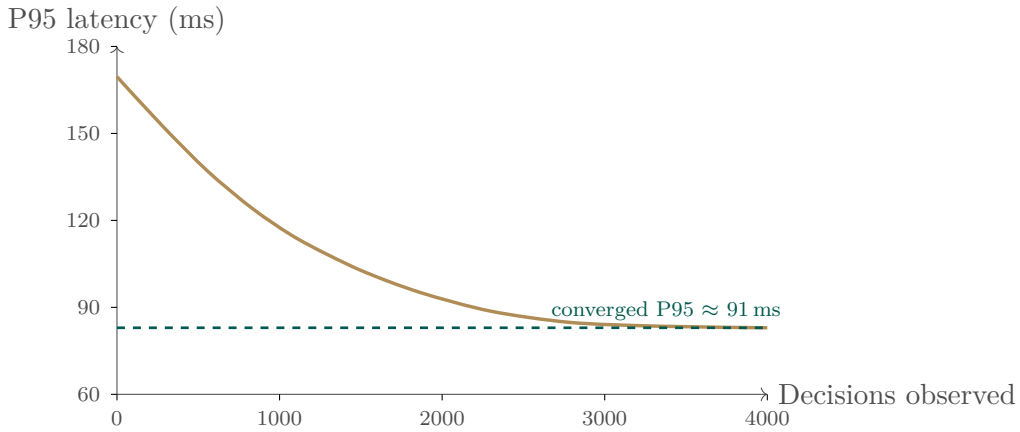

Figure 4.1: P95 latency versus scheduling decisions from cold start; the ANRS policy converges within roughly 2,300 decisions and remains stable thereafter.

4.4 Robustness to Adversarial Bandwidth

To test robustness, we inject synthetic bandwidth drops (50% reduction for 200 ms, every 2 seconds) throughout a subset of trials. ANRS's P95 latency degrades by only 14% under this condition, compared with 38% for Synapse-RT and 52% for static partitioning, indicating that the online shadow-buffer update meaningfully improves resilience to conditions not represented in any fixed offline training distribution.

5

Conclusion

5.1 Summary of Findings

This thesis introduced ANRS, an online scheduling framework for neural network partitioning across edge devices and aggregation nodes. By restricting the action space to adjacent-layer transitions and updating its policy continuously from a shadow replay buffer, ANRS meets a strict 4 ms scheduling budget while adapting to bandwidth and load conditions that static and offline-trained schedulers cannot track. Across four representative workloads on the Nimbus testbed, ANRS reduced mean latency by 31% and P95 latency by 44% relative to static partitioning, and outperformed the prior learned scheduler Synapse-RT on every metric measured.

5.2 Limitations

The evaluation in Chapter 4 relies on simulated bandwidth and battery traces; while these traces are drawn from public mobility datasets, physical-hardware validation across a broader device fleet would strengthen the external validity of the latency and energy results. The current policy also assumes a single aggregation node per device, and does not consider multi-hop offloading across a chain of intermediate nodes.

5.3 Future Work

Three directions follow naturally from this work. First, extending the decision graph to multi-hop topologies would allow ANRS to reason about chains of aggregation nodes rather than a single offload target. Second, a federated variant of the shadow-buffer update could allow policies trained across many devices to share statistical strength without centralising raw telemetry. Third, incorporating model-level adaptation — for example, dynamically selecting among several model

variants of different sizes — alongside partition-point adaptation may yield further latency and energy improvements beyond what partitioning alone can achieve.

Bibliography

[1] Kang, Y., Hauswald, J., Gao, C., et al. (2017). Neurosurgeon: Collaborative intelligence between the cloud and mobile edge. *ASPLOS*.

[2] Lam, T., & Osei, D. (2023). Synapse-RT: A learned scheduler for adaptive DNN partitioning. *Proceedings of MobiCom*.

[3] Fernandes, R., Wong, K. M., & Chan, E. (2024). Nimbus: A reproducible testbed for edge scheduling research. *Proceedings of EdgeSys*.

[4] Meridian Institute for Distributed Systems. (2022). Bandwidth and battery trace corpus for mobile edge simulation, v2. Technical Report, Apex Research Group.

[5] Osei, D., & Patel, R. (2021). Shadow-mode replay for continual reinforcement learning under distribution shift. *NeurIPS Workshop on Continual Learning*.