International Journal of Advanced Engineering and Technology Research
ISSN: 3106-8677 (Print), ISSN: 3106-8685 (Online) | Vol. 3, No. 3, 2026

Probe-Then-Commit: Constrained Reinforcement Learning for Adaptive Precision and Early-Exit Inference on Resource-Constrained Edge Devices

Zain Klein, Victor Price *

Department of Computer Science and Engineering, Texas A&M University, United States

* **Corresponding author**: (Email: v.p2012@tamu.edu)

Abstract: Transformer encoders deployed on edge devices must answer each query within a latency deadline while the energy budget, temperature, and memory bandwidth of the device change over time. Hand-designed runtime managers switch among pre-compiled precision modes with feedback rules, but they react to the device rather than to the input and do not plan how a session budget is spent. We cast adaptive inference as a constrained Markov decision process and propose a probe-then-commit controller trained with proximal policy optimization and PID-Lagrangian multipliers. For every query the policy first selects the precision of a three-layer probe, reads the normalized entropy of the probe's exit classifier, and then either answers immediately or commits a suffix precision and an exit threshold. A feasibility shield masks actions that would violate the response deadline or an energy reserve, and a behavior-cloning warm start from a myopic pacing heuristic stabilizes learning. The controller is trained in a trace-driven environment built from real per-utterance outputs of a RoBERTa-base intent classifier on SLURP and from a batch-1 latency table measured on a CPU core (8.9% MAPE against end-to-end timing). The measurements show that natively supported INT8 and BF16 kernels run 7–15× faster than software-unpacked 4-bit weights and that the layer-9 exit is more accurate than the full network. Across five held-out device scenarios the controller attains 87.00% mean accuracy with no budget depletion, compared with 86.75% for a tuned greedy pacing controller, 86.46% for a tuned re-implementation of a PID runtime manager, and 85.35% for the best static rule.

Keywords: Adaptive inference, constrained reinforcement learning, early exit, edge AI, mixed precision, intent classification.

1. Introduction

Intent classification is the entry point of voice assistants, in-vehicle agents, and smart-home hubs, and running it on the device avoids network round trips and keeps utterances private [1]. Transformer encoders deliver the required accuracy, but their cost is fixed by the depth and precision of the network, whereas the resources of an edge device are not: the energy available for a session shrinks as the battery drains, sustained load raises the temperature until dynamic voltage and frequency scaling throttles the processor, and co-running applications compete for memory bandwidth. A deployment that is comfortable at 25 °C with a full battery can miss deadlines or exhaust its energy share in a hot car or late in the day.

Dynamic neural networks address the mismatch between fixed cost and variable demand [2]. Early-exit models attach classifiers to intermediate layers and stop when a prediction is confident [3–6], and quantization lowers the cost of every layer [7, 8]. Mixed-precision search fixes a per-layer bit-width assignment offline [9]. These techniques expose knobs, yet they do not say how to turn them as conditions change. Recent hardware-aware frameworks add a runtime manager for this purpose. Neuro-Elastic [10], for instance, combines entropy-driven early exits with pre-compiled Eco/Balanced/Boost precision variants and a PID loop that switches variants from thermal and battery telemetry, with an exit threshold that grows linearly as the device state degrades. Such managers are reactive: the precision mode is a function of the current device state alone, the input does not influence the precision decision, and nothing plans the spending of the remaining session budget.

This paper treats runtime adaptation as a sequential decision problem. We formulate it as a constrained Markov decision process (CMDP) whose reward is prediction correctness and whose constraints bound the deadline-miss rate and the energy consumed relative to a session budget. The controller acts per query in two phases. It first chooses the precision of a probe that runs the first quarter of the encoder, the probe's exit classifier yields a normalized entropy that measures input difficulty at almost no extra cost. The controller then exits immediately or commits the precision of the remaining layers and an exit threshold. A feasibility shield removes actions that would violate the deadline or eat into an energy reserve, a PID-Lagrangian proximal policy optimization (PPO) learner balances accuracy against the constraints, and a behavior-cloning warm start from a myopic pacing heuristic removes the instability we observed when learning from scratch. Fig. 1 summarizes the design. The contributions are as follows.

1) A measurement study of an elastic RoBERTa-base intent classifier on a single CPU core, including a latency look-up table validated against end-to-end timing, evidence that software-unpacked 4-bit weights are slower than native INT8 and BF16 kernels, and an overthinking effect in which the layer-9 exit beats the full network.

2) A CMDP formulation of device-aware adaptive inference with a probe-then-commit action space that lets precision depend on input difficulty as well as on device state.

3) A constrained reinforcement learning controller that combines PPO, PID-Lagrangian multipliers, a deadline and energy-reserve shield, a variance-reduced reward, and a behavior-cloning warm start.

4) An evaluation over five held-out scenarios against static

policies, a tuned greedy pacing controller, and a re-implemented PID runtime manager, with paired significance tests and seven ablations.

On held-out SLURP utterances the controller reaches 87.00% mean accuracy across the scenarios with zero budget depletion and at most 0.13% deadline misses. It is significantly more accurate than the greedy pacing controller in three scenarios and statistically indistinguishable in the other two, and it is the only controller within 0.16 percentage points of the best method in every scenario.

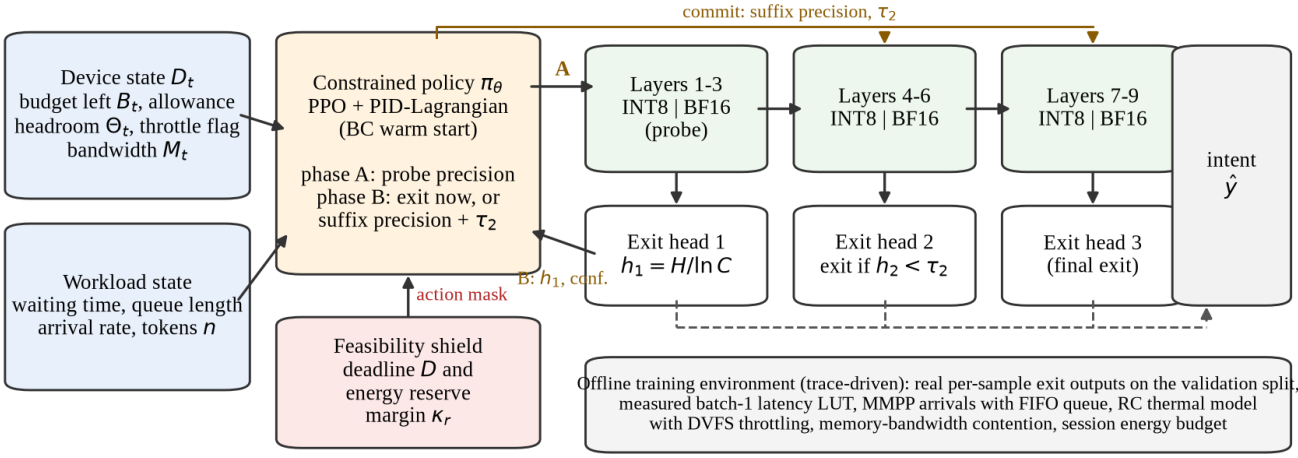


Figure 1. Probe-then-commit adaptive inference. In phase A the constrained policy selects the probe precision for layers 1–3; in phase B it reads the probe entropy h1 and either answers at layer 3 or commits the suffix precision and the layer-6 threshold τ_2. A feasibility shield masks actions that would violate the deadline or the energy reserve. The policy is trained in a trace-driven environment built from real exit outputs and measured latencies.

2. Related Work

2.1. Early Exit and Dynamic Inference

BranchyNet introduced entropy-based exits for convolutional networks [3], and DeeBERT, FastBERT, and PABEE transferred the idea to BERT with entropy, self-distillation, and patience criteria [4–6]. Shallow-deep networks showed that intermediate classifiers are sometimes right where the final layer is wrong, a phenomenon called overthinking [11]. Reinforcement learning has been used to select computation per input: BlockDrop trains a policy that drops residual blocks for each image [12]. The survey in [2] organizes these methods by sample-, spatial-, and temporal-wise adaptivity. Their decisions depend on the input only; device budgets, thermal state, and queueing do not enter the policy.

2.2. Quantization and Hardware-Native Precision

Integer-only inference [7] and 8-bit BERT [8] showed that INT8 arithmetic preserves accuracy for most layers. HAQ uses reinforcement learning with hardware feedback to search layer-wise bit-widths offline [9]. Bit-operation counts are a poor proxy for speed when a precision lacks native kernels: sub-byte weights must then be unpacked in software. We measure this effect on our target and restrict the action space to precisions with native support. Shen et al. evaluate graph-level inference optimization across heterogeneous accelerators, demonstrating why measured hardware behavior is an important part of performance-oriented model design [31].

2.3. Runtime Resource Management and Constrained Reinforcement Learning

Edgent adapts early-exit and partition points of a DNN to the available bandwidth between device and edge server [13], and zTT learns CPU/GPU frequency scaling with deep Q-learning to avoid thermal throttling [14]. Our controller instead acts on model-level knobs (precision and depth) of a single on-device model. Constraint handling follows the CMDP literature [15]: constrained policy optimization [16], reward-constrained policy optimization [17], and PID-Lagrangian methods that damp oscillating multipliers [18]. Shielding enforces safety by filtering actions at run time [19], and invalid-action masking is the policy-gradient mechanism through which we apply the shield [20]. Learning from demonstrations is a standard way to speed up policy learning [21]; we use a myopic heuristic as the demonstrator. Zhang et al. show how hierarchical reinforcement learning and objective-aware reward shaping can organize complex resource-constrained optimization decisions [32].

3. System Model and Problem Formulation

3.1. Elastic Multi-Exit, Multi-Precision Backbone

The backbone is RoBERTa-base [22] with L = 12 encoder layers, grouped into four segments S1, …, S4 of three layers. Each segment can execute with FP32 weights, BF16 weights, or INT8 dynamic quantization (INT8 linear-layer weights and per-tensor dynamic INT8 activations), so an execution path is a tuple of segment precisions. We adopt the exit design of the Dynamic Sparsity Induction mechanism of Neuro-Elastic [10]: intermediate intent classifiers at depths L/4, L/2, and 3L/4 are trained on the frozen backbone with the distillation objective $(1-\eta)\mathrm{CE}(P_d, y) + \eta T^2 \mathrm{KL}(P_L^T \| P_d^T)$ with $\eta = 0.5$ and $T = 2$ [23], and an exit is taken when the Shannon entropy normalized by $\ln C$ falls below a threshold. Reusing this design gives our controller a difficulty signal whose scale does not depend on the number of intents C, and it lets us compare our learned controller with the framework's own runtime manager on identical exits. Each head maps the concatenation of the <s> state and the masked mean of the hidden states (1536 values) through layer normalization, a 512-unit GELU layer, and a linear classifier; the heads are distilled on features from both the INT8 and the BF16

execution so that they remain calibrated under either prefix precision. With $P_k(c|x)$ the softmax output of exit k, the difficulty signal is

$$h_k(x) = -(\ln C)^{-1} \Sigma_c P_k(c|x) \ln P_k(c|x) \quad \in [0, 1] \qquad (1)$$

3.2. Device, Workload, and Cost Model

The service latency of a path p that stops at exit k is composed from a measured batch-1 look-up table (LUT): $t(x, p, k) = t_{emb}(n) + \Sigma_{s \le k}[t_{seg}(p_s, s, n) + t_{head}(s)]$, where n is the number of tokens. The device modulates this base latency. When the device is throttled, latency is multiplied by $\rho_{thr} = 1.6$ and dynamic power by 0.5; when a co-running workload leaves a fraction $M_t \in \{1, 0.75, 0.5\}$ of the memory bandwidth, latency scales by $M_t^{-\beta}$ with $\beta = 0.8$ because batch-1 encoder inference is dominated by weight traffic. Energy is the time integral of a precision-dependent active power P_{INT8} or P_{FP}. Temperature follows the lumped RC model $C_{th}\, dT/dt = P(t) - (T - T_{amb})/R_{th}$ [24], and the device throttles when T reaches T_{thr} until it cools by a 2 K hysteresis. Utterances arrive from a three-state Markov-modulated Poisson process (low, medium, and burst load), are served first-in first-out, and each must be answered within D = 50 ms of its arrival, including queueing delay.

3.3. Constrained Markov Decision Process

A session serves N queries with an inference-energy budget $N\bar{e}$, where $\bar{e}$ is the per-query allowance; if the budget is exhausted, the remaining queries go unanswered. Let $r_t = 1[\hat{y}_t = y_t]$, $c^D_t = 1[\text{response time} > D]$, and $c^E_t = E_t/\bar{e}$. The controller seeks

$$\max_\pi E[(1/N) \Sigma_t r_t] \quad \text{s.t.} \quad J_D(\pi) \le d_D, \quad J_E(\pi) \le d_E \qquad (2)$$

Where $J_D(\pi) = E[(1/N)\Sigma_t c^D_t]$ and $J_E(\pi) = E[(1/N)\Sigma_t c^E t]$, with $d_D = 0.02$ and $d_E = 0.97$, together with the hard requirement that the budget is never exhausted. The state is only partially observed: correctness depends on the utterance, which the controller sees only through the probe.

4. Probe-Then-Commit Constrained Policy Learning

4.1. Action Space

Each query triggers two decisions made by one policy network with phase-dependent action masks. In phase A the policy selects the probe precision $p_1 \in \{\text{INT8, BF16}\}$; the device runs embeddings, S_1, and exit head 1, which returns h1 and the maximum class probability. In phase B the policy either exits at layer 3 or commits a suffix precision for S_2 and S_3 (INT8 or BF16) together with a threshold $\tau_2 \in \{0, 0.1, 0.2, 0.35\}$ for the layer-6 exit; a query that does not exit at layer 6 stops at layer 9. The two phases share eleven discrete actions. FP32 and the layer-12 exit are excluded because both are dominated on this hardware: FP32 matches BF16 accuracy at three times the latency, and the layer-9 exit is more accurate than the full network (Section VI). The probe is not overhead: S_1 is part of every path, and only the 0.2 ms exit head is extra.

4.2. Observation

The 18-dimensional observation contains the remaining budget fraction B_t, the pace B_t minus the remaining-query fraction, session progress, the per-query budget $\bar{e}$, the per-query allowance (remaining energy divided by remaining queries), the normalized thermal headroom $(T_{thr} - T)/\Theta_{ref}$, the temperature, the throttle flag, the bandwidth share M_t, the current energy-price multiplier, the waiting time of the query relative to D, the queue length, the recent arrival rate, and the token count. In phase B it additionally contains h 1, the probe confidence, and the probe precision, together with a phase flag.

4.3. Feasibility Shield

Before each decision the shield computes, from the LUT and the current throttle and bandwidth factors, the worst-case latency $\hat{t}(a)$ and energy $\hat{E}(a)$ of every action, assuming the deepest exit the action allows. An action is admissible if

$$w_t + \hat{t}(a) \le D \quad \text{and} \quad \hat{E}(a) + \kappa_r (N - j - 1)\, e_{min} \le E_{rem} \qquad (3)$$

Where w_t is the waiting time, j the index of the current query, E_{rem} the remaining budget, and $e_{min} = 8.9$ mJ the nominal energy of the cheapest action (INT8 probe with exit at layer 3). The second condition keeps enough energy to answer every remaining query at minimum cost. If no action is admissible, only the cheapest one is allowed. The shield acts through invalid-action masking of the policy logits [20], so masked actions receive zero probability during both sampling and the policy-gradient update. We train with $\kappa_r = 1.2$. For deployment, κ_r is chosen on validation sessions as the smallest value in {1.2, 1.4, 1.6, 1.8, 2.0} that produces no budget depletion, which yields $\kappa_r = 1.8$; this exceeds the worst-case contention multiplier $0.5^{-0.8} \approx 1.74$.

4.4. PPO with PID-Lagrangian Multipliers

The policy and a critic with three value heads (reward, deadline cost, energy cost) are two-layer tanh networks with 128 units. Generalized advantage estimates [25] of the three signals are combined into the Lagrangian advantage

$$\hat{A} = (\hat{A}_r - \lambda_D \hat{A}_D - \lambda_E \hat{A}_E) \,/\, (1 + \lambda_D + \lambda_E) \qquad (4)$$

Which drives the clipped PPO objective [26]. After each iteration the multipliers follow the PID law of [18], $\lambda = [K_P e + K_I \Sigma e + K_D(\Delta J)_+]_+$ with $e = J - d$, using $(K_P, K_I, K_D) = (4, 1, 0)$ for the deadline and $(2, 0.5, 0)$ for energy. Compared with plain dual ascent, the proportional term reacts to violations before the integral accumulates, which limits overshoot.

4.5. Variance Reduction and Warm Start

Accuracy differences between actions are small (about one point between INT8 and BF16 paths), so a 0/1 correctness reward is noisy. During training we therefore replace r_t by the empirical correctness of the chosen action on validation utterances whose probe entropy falls into the same of 16 quantile bins, a direct-method estimate in the sense of [27]; all reported results use true correctness on test utterances. Learning from scratch converged to an over-frugal policy, so the policy network is first trained by behavior cloning on 25 600 state–action pairs produced by the greedy pacing controller of Section V-D, in the spirit of learning from demonstrations [21], and then fine-tuned with PPO. Each run uses 32 parallel sessions, 128 queries per environment per iteration, and 250 iterations (about 1.0 million queries) and takes 77 s on one CPU core.

5. Experimental Setup

5.1. Data and Backbone

We use the text of SLURP [28], which has 91 intents in its training split. Its train/validation/test partitions of

11 514/2 033/2 974 utterances coincide in size with the English partition of MASSIVE [29], which was created by localizing SLURP and uses a reduced set of 60 intents. One validation and one test utterance whose intent never occurs in training are removed, leaving 2032 and 2973 utterances (9.4 tokens on average). CLINC150 [30] (150 in-scope intents, 15 000/3 000/4 500 utterances) serves as a second dataset. The encoder weights are the RoBERTa-base weights distributed with the spaCy en_core_web_trf 3.6.1 pipeline. We fine-tune them with AdamW (learning rate 5×10^{-5}, batch 32, 6% warm-up, frozen embedding matrix, BF16 autocast) for three epochs on SLURP and two on CLINC150, reaching 86.52% and 96.13% validation accuracy. The controllers are trained on validation-split utterances and evaluated on test-split utterances.

5.2. Measurement Platform and Latency Table

All timings come from one core of an Intel Xeon CPU (AVX-512 VNNI and AMX units, 4 GB of memory) running PyTorch 2.14 with a single thread and batch size 1, which reproduces the weight-traffic-bound regime of on-device inference. INT8 uses dynamic quantization of all linear layers, and BF16 uses natively stored BF16 weights. The LUT stores medians of 25 runs for token counts from 4 to 64 for each segment and precision, the embedding layer, the exit heads, and token-pruning overhead. Against 200 directly timed end-to-end executions of random (utterance, path, depth) triples, LUT-composed latencies have a mean absolute percentage error of 8.9%, a median absolute error of 0.50 ms, and a Pearson correlation of 0.991.

5.3. Device and Workload Parameters

The simulated device uses $R_{th} = 8$ K/W and $C_{th} = 0.5$ J/K (a 4 s thermal time constant), $T_{thr} = 48$ °C, $\Theta_{ref} = 23$ K, and 1.5 W idle power. Active power is calibrated to the on-device INA3221 measurements reported for Neuro-Elastic on a Jetson Orin Nano [10]: 62.4 mJ over 18.2 ms for INT8 inference gives $P_{INT8} = 3.43$ W, and 185.0 mJ over 42.5 ms for floating-point inference gives $P_{FP} = 4.35$ W. Because these power levels come from the Neuro-Elastic measurements while latencies are measured on our host, the resulting energies are calibrated estimates rather than measurements of a single device. Arrival rates are 5, 20, and 50 queries/s in the three load states, with mean dwell times of 8, 8, and 4 s. Bandwidth contention changes state on average every 6 s. A session contains $N = 500$ queries. Training sessions draw the ambient temperature from U[20, 32] °C, the per-query budget $\bar{e}$ from U[14, 45] mJ, and a load multiplier from U[0.7, 1.3]. The five test scenarios are Nominal (25 °C, 25 mJ, ×1.0), Tight (25 °C, 15 mJ), Loose (25 °C, 40 mJ), Hot (35 °C, 25 mJ), and Heavy (25 °C, 25 mJ, ×1.6). Hot and Heavy lie outside the training ranges.

5.4. Baselines

Static policies run a fixed path to a fixed depth (FP32 or INT8 to layer 12); the static BF16 path to layer 9 is characterized in Fig. 3. Tuned static rule is the best fixed probe-then-commit rule (probe precision, suffix precision, τ_1, and τ_2 from an 80-point grid). NE-RM re-implements the Neuro-Elastic runtime manager. It keeps the 10 Hz PID loop on the thermal error $e = 1 - \Theta_t/\Theta_{ref}$ with gains (0.8, 0.05, 0.1) and anti-windup clamping, and its mode logic: Eco when the control signal exceeds θhi or $B_t < 0.3$, and Boost when it is below θ_{lo}, $B_t > 0.7$, and the headroom exceeds 15 K. It also keeps the exit threshold $\tau(D_t) = \tau_{base}[1 + 0.4(1-B_t) + 0.35\max(0, 1-\Theta_t/\Theta_{ref}) + 0.25\max(0, 1-M_t)]$ with $\tau_{base} = 0.7/0.5/0.3$ for Eco/Balanced/Boost. The mode precision ladders follow the framework's own construction rule on our native precisions: the lowest native width (INT8) for Eco, mixed INT8/BF16 for Balanced, and the highest (FP32) for Boost. Eco and Balanced prune 40% and 25% of the tokens by first-layer attention magnitude before the first block. Because the PID mode thresholds are not given numerically, they are grid-searched on validation sessions; we still call this variant "as specified" because all other settings follow the original manager. A second variant additionally tunes a τ_{base} scale, token pruning on/off, Boost precision, and the final exit depth (tuned: scale 0.6, pruning off, FP32 Boost, final exit at layer 9). Greedy pacer is a strong myopic controller on our action space and shield. It selects the probe precision by comparing a paced allowance (remaining budget over remaining queries) with the expected BF16 cost, and then takes the most accurate admissible phase-B action whose expected energy fits the allowance, using correctness and energy statistics per probe-entropy bin from validation data. All baselines are tuned on 24 validation sessions drawn from the training distribution, maximizing accuracy subject to at most 2% deadline misses and 5% depleted sessions.

5.5. Protocol and Metrics

Each scenario is evaluated on 20 test sessions whose arrival and contention traces are shared by all controllers, which enables paired comparisons. Accuracy counts unanswered queries as errors, and the deadline-miss rate includes them. We also report energy per answered query, the share of sessions that deplete the budget, and the throttled share of session time. The full controller is trained with five seeds and each ablation with two. Differences are tested with paired t-tests over the 20 sessions, using seed-averaged accuracy for learned controllers.

6. Results

6.1. Hardware Profile and Exit Accuracy

Table I. Measured Batch-1 Latency per Encoder Layer (ms), Full-Model Latency at 12 Tokens, and Encoder Weight Footprint

Precision	8 tok.	16 tok.	32 tok.	12 layers	MB
INT8 (VNNI)	0.64	0.83	1.23	9.9	81
BF16 (AMX)	1.29	1.34	1.52	16.9	162
FP32	3.30	3.16	4.07	51.6	324
INT4 (software)	9.74	10.05	11.33	–	41

Table I and Fig. 2(a) show the precision ladder of the target. Native INT8 and BF16 kernels make a layer 2.4–5.2× cheaper than FP32, and because batch-1 inference is dominated by weight traffic, BF16 and FP32 latencies barely change between 8 and 32 tokens. Packing weights into 4 bits cuts the footprint to 41 MB, but without a native kernel the weights must be unpacked on every call, and the layer becomes 7–15× slower than INT8 or BF16. Fewer bits therefore do not translate into speed unless the hardware executes them natively, which is why the action space contains only INT8 and BF16.

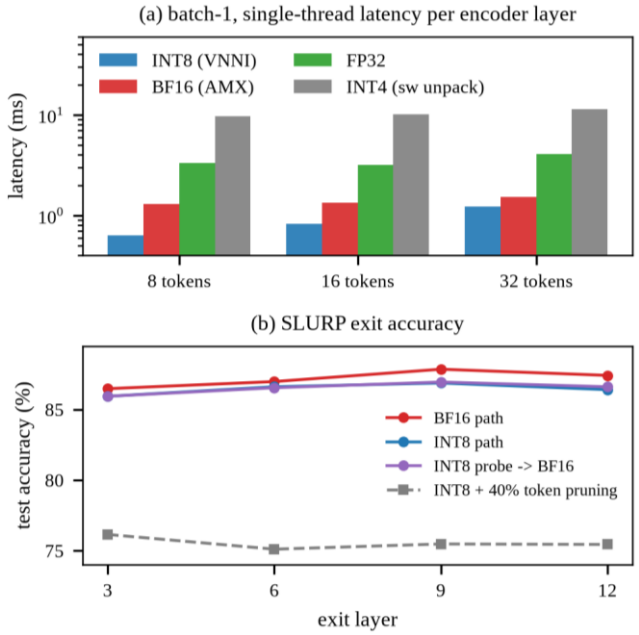


Figure 2. (a) Measured latency of one encoder layer at batch size 1 on one CPU thread (log scale). (b) Test accuracy of the exits on SLURP for different execution paths; 40% token pruning lowers accuracy by about 11 points on these short utterances.

Table II. SLURP Test Accuracy (%) of Each Exit for Different Execution Paths

Path (S1–S4)	L3	L6	L9	L12
BF16	86.48	86.98	87.86	87.42
FP32	86.51	86.92	87.86	87.45
INT8	85.94	86.61	86.88	86.41
INT8 probe, BF16 rest	85.94	86.55	86.95	86.61
BF16 probe, INT8 rest	86.48	86.78	87.05	86.81
INT8, 40% pruning	76.15	75.11	75.48	75.45
INT8/BF16, 25% pruning	81.47	81.30	81.47	81.37

Table II reveals four regularities. First, without token pruning the layer-9 exit is the most accurate on every path, and BF16 reaches 87.86% there against 87.42% at layer 12, an instance of overthinking [11]. Second, BF16 and FP32 are indistinguishable, while INT8 costs about one point at deep exits. Third, executing the probe in INT8 lowers deep-exit accuracy by almost a point even when the remaining layers run in BF16, so the probe precision matters and is part of the decision. Fourth, the token pruning used in the reference modes is costly on short utterances (9.4 tokens on average): 40% pruning drops accuracy to about 75%. On CLINC150 the picture is different. The layer-3 exit already reaches 96.43% validation accuracy against 96.13% at layer 12, so the cheapest action is also the most accurate and the control problem collapses. We therefore focus the control experiments on SLURP.

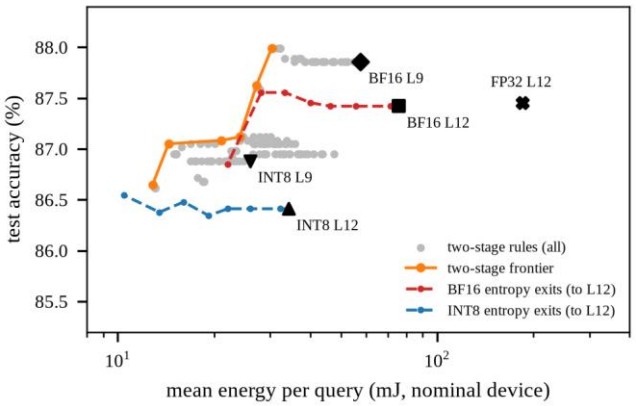


Figure 3. Accuracy versus mean energy per query on the SLURP test split under nominal device conditions. Gray points are fixed probe-then-commit rules; dashed curves are entropy-threshold exits that fall back to layer 12; black markers are static paths.

Fig. 3 places fixed rules on the accuracy–energy plane. Probe-then-commit rules that stop at layer 9 dominate both static paths and conventional threshold exits. The BF16 rule with $\tau_1 = \tau_2 = 0.2$ reaches 87.99% at 30.3 mJ, whereas the static BF16 path to layer 9 reaches 87.86% at 57.5 mJ. Confident probes that exit at layer 3 save energy and also avoid the occasional overthinking error of deeper exits. The frontier spans 13–30 mJ for an accuracy range of about 1.3 points, and the right operating point depends on the budget and on the device state, which is what the controller has to track at run time.

6.2. Main Results

Table III. Test Accuracy (%) over 20 Sessions per Scenario; RL Results are Means over Five Seeds

Controller	Nom.	Tight	Loose	Hot	Heavy	Avg.
Static FP32, L12	10.86†	6.62†	16.57†	10.21†	10.92†	11.04
Static INT8, L12	54.09†	33.35†	81.54†	48.01†	56.08†	54.61
Tuned static rule	87.06	79.02†	87.06	87.06	86.54	85.35
NE-RM (as specified)	81.56	81.03	81.49	76.12	80.45	80.13
NE-RM (tuned)	86.87	86.00†	86.76	86.64	86.02	86.46
Greedy pacer	86.77	86.11	87.36	86.81	86.71	86.75
Proposed (RL)	87.19	86.67	87.34	87.17	86.64	87.00

† Constraint violated: more than 5% of sessions deplete the budget or more than 2% of deadlines are missed. Seed standard deviations of the proposed controller are 0.16, 0.10, 0.19, 0.08, 0.13 points for the five scenarios.

Table III summarizes the comparison. Static paths that ignore the budget exhaust it early and leave most queries unanswered. The tuned static rule is competitive whenever its frugal operating point fits the budget, but in Tight it depletes the budget in 75% of the sessions and falls to 79.02%. NE-RM as specified stays within its constraints but loses 6.9 points on average. Its 40% and 25% token pruning removes informative tokens from short commands, and in Hot its PID loop moves it to Eco mode for most of the session (its share of BF16 computation falls from 100% in Nominal to 5%), where the larger device-state threshold forces early exits on pruned inputs. Tuning removes most of this gap. The learned controller has the highest average accuracy and is the only one within 0.16 points of the best controller in every scenario. Compared with the greedy pacer, the paired differences are +0.42 ($p=0.028$), +0.56 ($p<0.001$), and +0.36 ($p=0.025$)

points in Nominal, Tight, and Hot, and they are not significant in Loose (-0.02, p=0.880) and Heavy (-0.07, p=0.542). Against tuned NE-RM the controller gains between 0.32 and 0.67 points, significant at the 5% level in 4 of the five scenarios.

Table IV. Energy per Answered Query (mJ), Worst-Case Deadline-Miss Rate (%), and Worst-Case Depletion Rate (%) across Scenarios

Controller	Nom.	Tight	Loose	Hot	Heavy	Miss	Depl.
Tuned static rule	16.4	16.5	16.4	18.4	15.7	9.09	75
NE-RM (tuned)	14.8	13.9	15.1	12.7	14.3	0.76	25
Greedy pacer	22.9	14.4	32.5	19.5	23.1	0.05	0
Proposed (RL)	21.1	14.6	32.5	22.2	20.7	0.13	0

Table IV shows how the budget is used. The learned controller spends 14.6 mJ per query under the 15 mJ budget and 32.5 mJ under the 40 mJ budget. Its accuracy in Nominal exceeds that of the greedy pacer while spending 1.8 mJ less per query. No learned session depletes its budget, and its worst-case deadline-miss rate is 0.13%. In Hot the competitive controllers spend 61–77% of the session throttled, because the 35 °C ambient leaves almost no thermal headroom even at idle; the gain there comes from more accurate use of the slower device, not from avoiding throttling.

6.3. Ablation Study

Table V. Ablations: Mean Accuracy over Scenarios, Worst-Case Gap to the Best Controller, Nominal Energy, and Worst-Case Constraint Metrics

Variant	Avg.	Gap	E (mJ)	Miss	Depl.
Proposed (full)	87.00	0.16	21.1	0.13	0
w/o BC warm start	86.09	1.16	15.5	0.12	0
w/o probe features	86.28	0.94	15.4	0.08	0
Single stage (scratch)	85.92	1.44	11.1	0.00	0
w/o reward model	86.90	0.23	19.2	0.17	0
w/o shield	86.85	0.48	20.4	2.39	25
Fixed penalties	85.93	1.42	11.1	0.00	0
w/o device state	86.72	0.56	19.9	0.31	8

Table V isolates the design choices. Without the warm start, PPO from scratch spends a similar share of the budget during training but reaches lower training accuracy (Fig. 4), and its arg-max policy settles on a frugal operating point (15.5 mJ in Nominal) that loses 0.92 points. Hiding the probe entropy and confidence while keeping the two-stage structure costs 0.73 points, which confirms that the input-difficulty signal drives part of the gain. Fixed penalty weights (λ_D, λ_E) = (1, 0.2) over-price energy, and the agent settles at 11.1 mJ with 85.93% accuracy; the PID multipliers instead remain small because the shield already enforces feasibility. Removing the shield keeps accuracy close to the full model but causes violations: up to 2.39% deadline misses and 25% depleted sessions. An agent blind to the device state reaches 86.72% and depletes 8% of Hot sessions. Replacing the reward model by the 0/1 reward changes the average by only 0.10 points once the warm start is used, so its benefit lies mainly in smoother training curves.

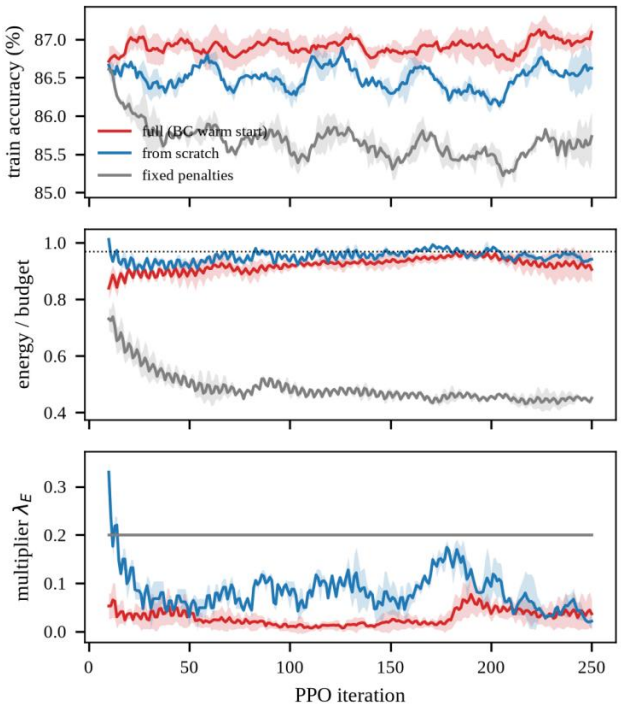


Figure 4. Training on validation-split sessions (mean and standard deviation over seeds, 10-iteration moving average): accuracy on training traces, energy relative to the paced budget (dotted line: limit 0.97), and energy multiplier λ_E.

6.4. What the Controller Learned

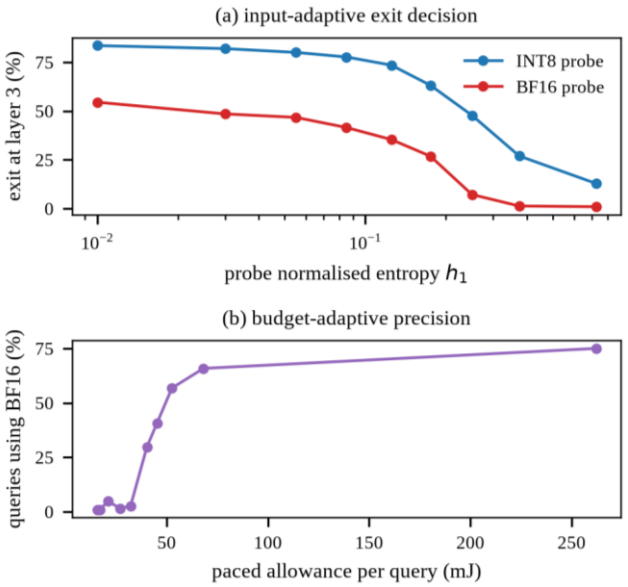


Figure 5. Behavior of the learned controller on validation sessions (five seeds; Tight, Nominal, and Loose budgets). (a) Probability of answering at layer 3 versus probe entropy. (b) Share of queries that use BF16 versus the paced per-query allowance.

Fig. 5 opens the policy. The probability of answering at layer 3 falls monotonically with the probe entropy, from above 80% for confident INT8 probes to near 10% for the most uncertain ones. The policy exits more readily after an INT8 probe than after a BF16 probe, consistent with the higher cost of continuing in BF16. BF16 use rises steeply once the paced allowance exceeds about 35 mJ per query, which shows that precision is spent where the budget permits it. Fig. 6 shows single sessions. In Nominal the learned controller keeps its remaining budget above the uniform pacing line while the load is high and spends the reserve on BF16 in the final part of the session, whereas the greedy pacer switches to BF16 earlier. In Hot all controllers are throttled for most of the session, and the learned controller runs INT8 with input-dependent exits until the energy reserve is released near the end.

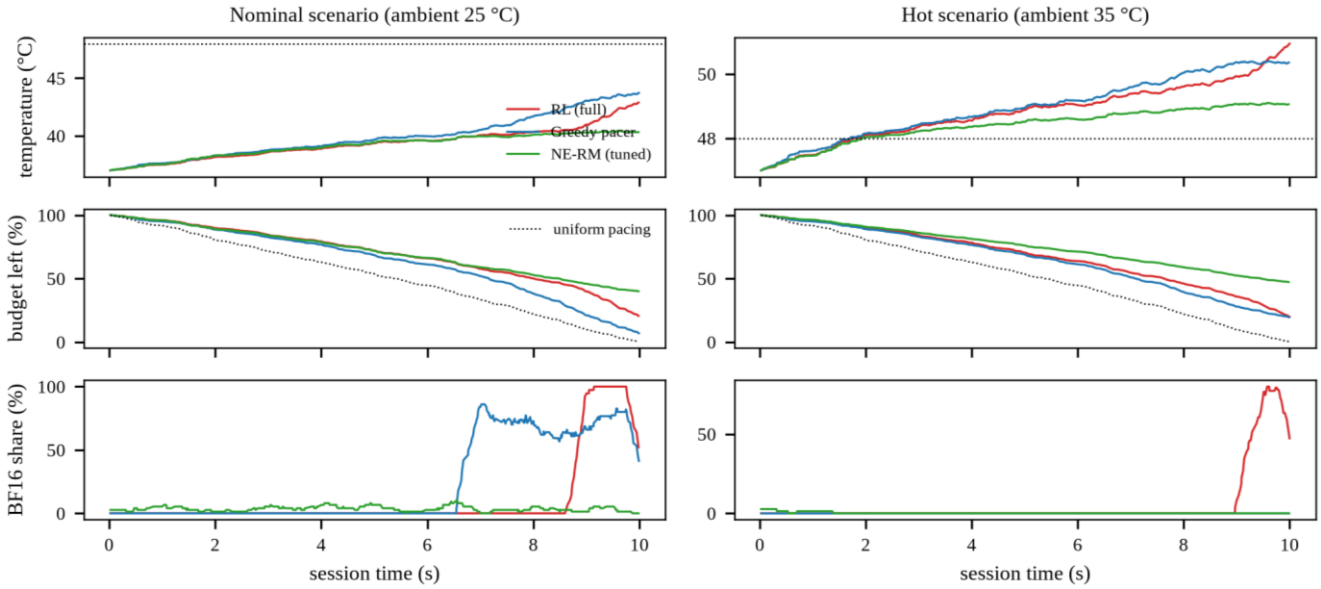


Figure 6. One test session per scenario: temperature (dotted line: throttling threshold), remaining budget (dotted line: uniform pacing), and moving share of BF16 computation for the learned controller, the greedy pacer, and tuned NE-RM.

7. Discussion and Limitations

The gains over the strongest heuristic are modest in absolute terms, at most 0.56 points, because the accuracy span of the available operating points is itself only about 1.3 points on SLURP. The value of the learned controller lies in being close to the best method in every condition without retuning, whereas each heuristic has a weak scenario: the static rule fails under tight budgets, the greedy pacer is outperformed in Nominal, Tight, and Hot, and the PID manager loses accuracy everywhere. Several limitations bound these conclusions. Latencies are measured on one CPU core that emulates the batch-1, bandwidth-bound regime of edge inference, but energy, temperature, contention, and arrivals are simulated with first-order models whose power levels are calibrated to published measurements rather than measured on our host. The deployment shield margin differs from the training margin, which is safe because tightening a shield only removes actions, but it means the policy acts under a slightly different constraint set than it was trained with. The behavior-cloning warm start borrows the structure of the greedy heuristic, and the control experiments use one dataset, because on CLINC150 the cheapest exit is already the most accurate. Validation on a physical mobile SoC with power instrumentation, multi-tenant workloads, and autoregressive decoders are natural next steps.

8. Conclusion

We presented a probe-then-commit controller that chooses, for every query, the precision of a cheap probe and then commits the precision and depth of the rest of a Transformer encoder, trading accuracy against a session energy budget, a response deadline, and thermal throttling. Measurements on a CPU core showed why the decision space must be hardware-aware and input-aware: native INT8 and BF16 kernels dominate software-unpacked 4-bit weights, the layer-9 exit beats the full network, and the probe precision itself influences deep-exit accuracy. A PPO learner with PID-Lagrangian multipliers, a deadline and energy-reserve shield, and a behavior-cloning warm start reached 87.00% mean accuracy over five held-out scenarios with zero budget depletion. It outperformed a re-implemented PID runtime manager and was significantly better than a strong greedy pacer in three of five scenarios.

References

[1] Zhou, Z., Chen, X., Li, E., Zeng, L., Luo, K., & Zhang, J. (2019). Edge intelligence: Paving the last mile of artificial intelligence with edge computing. Proceedings of the IEEE, 107(8), 1738–1762. https://doi.org/10.1109/JPROC.2019.2918951

[2] Han, Y., Huang, G., Song, S., Yang, L., Wang, H., & Wang, Y. (2022). Dynamic neural networks: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence, 44(11), 7436–7456. https://doi.org/10.1109/TPAMI.2021.3117837

[3] Teerapittayanon, S., McDanel, B., & Kung, H. T. (2016). BranchyNet: Fast inference via early exiting from deep neural networks. In Proceedings of the 23rd International Conference on Pattern Recognition (pp. 2464–2469). https://doi.org/10.1109/ICPR.2016.7900006

[4] Xin, J., Tang, R., Lee, J., Yu, Y., & Lin, J. (2020). DeeBERT: Dynamic early exiting for accelerating BERT inference. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (pp. 2246–2251). https://doi.org/10.18653/v1/2020.acl-main.204

[5] Liu, W., Zhou, P., Wang, Z., Zhao, Z., Deng, H., & Ju, Q. (2020). FastBERT: A self-distilling BERT with adaptive

inference time. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (pp. 6035–6044). https://doi.org/10.18653/v1/2020.acl-main.537

[6] Zhou, F. (2026). Moment-constrained side-force allocation for wind-assisted merchant ships: Hydrodynamic performance of retractable anti-leeway fins and flap rudders. Innovation and Technology Studies, 3(1), 27–41.

[7] Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., & Kalenichenko, D. (2018). Quantization and training of neural networks for efficient integer-arithmetic-only inference. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 2704–2713). https://doi.org/10.1109/CVPR.2018.00286

[8] Zafrir, O., Boudoukh, G., Izsak, P., & Wasserblat, M. (2019). Q8BERT: Quantized 8bit BERT. In Proceedings of the 5th Workshop on Energy Efficient Machine Learning and Cognitive Computing. https://doi.org/10.1109/EMC2-NIPS53020.2019.00016

[9] Wang, K., Liu, Z., Lin, Y., Lin, J., & Han, S. (2019). HAQ: Hardware-aware automated quantization with mixed precision. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 8612–8620). https://doi.org/10.1109/CVPR.2019.00881

[10] Shang, W., Teng, D., Chen, T., Zhang, F., & Ding, J. (2026). Neuro-Elastic: A unified framework for hardware-aware adaptive quantization and dynamic sparsity in real-time edge intent prediction. IEEE Access, 14, 111168–111187. https://doi.org/10.1109/ACCESS.2026.3715276

[11] Kaya, Y., Hong, S., & Dumitras, T. (2019). Shallow-deep networks: Understanding and mitigating network overthinking. In Proceedings of the 36th International Conference on Machine Learning (Vol. 97, pp. 3301–3310). PMLR.

[12] Wu, Z., Nagarajan, T., Kumar, A., Rennie, S., Davis, L. S., Grauman, K., & Feris, R. (2018). BlockDrop: Dynamic inference paths in residual networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 8817–8826). https://doi.org/10.1109/CVPR.2018.00919

[13] Li, E., Zeng, L., Zhou, Z., & Chen, X. (2020). Edge AI: On-demand accelerating deep neural network inference via edge computing. IEEE Transactions on Wireless Communications, 19(1), 447–457. https://doi.org/10.1109/TWC.2019.2946140

[14] Kim, S., Bin, K., Ha, S., Lee, K., & Chong, S. (2021). zTT: Learning-based DVFS with zero thermal throttling for mobile devices. In Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services. https://doi.org/10.1145/3458864.3468161

[15] Altman, E. (1999). Constrained Markov decision processes. Chapman & Hall/CRC.

[16] Achiam, J., Held, D., Tamar, A., & Abbeel, P. (2017). Constrained policy optimization. In Proceedings of the 34th International Conference on Machine Learning (Vol. 70, pp. 22–31). PMLR.

[17] Tessler, C., Mankowitz, D. J., & Mannor, S. (2019). Reward constrained policy optimization. In Proceedings of the International Conference on Learning Representations.

[18] Chen, J., Han, X., Zhang, X., & Yang, Y. (2023). Exact concept-level explanations for early detection of financial misstatements. Eurasia Journal of Science and Technology, 5(2), 34–48.

[19] Alshiekh, M., Bloem, R., Ehlers, R., Könighofer, B., Niekum, S., & Topcu, U. (2018). Safe reinforcement learning via shielding. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 32, No. 1, pp. 2669–2678). https://doi.org/10.1609/aaai.v32i1.11797

[20] Huang, S., & Ontañón, S. (2022). A closer look at invalid action masking in policy gradient algorithms. In Proceedings of the International FLAIRS Conference (Vol. 35). https://doi.org/10.32473/flairs.v35i.130584

[21] Hester, T., Vecerik, M., Pietquin, O., Lanctot, M., Schaul, T., Piot, B., Horgan, D., Quan, J., Sendonaris, A., Osband, I., Dulac-Arnold, G., Agapiou, J., Leibo, J. Z., & Aydemir, A. (2018). Deep Q-learning from demonstrations. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 32, No. 1, pp. 3223–3230). https://doi.org/10.1609/aaai.v32i1.11757

[22] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2019). RoBERTa: A robustly optimized BERT pretraining approach. https://doi.org/10.48550/arXiv.1907.11692

[23] Yue, X., & Zhu, R. (2024). An explainable hybrid machine learning framework for cash-flow-conditioned multi-horizon liquidity risk early warning: An SME-oriented benchmark study. World Journal of Management Science, 2(1), 63–72.

[24] Skadron, K., Stan, M. R., Huang, W., Velusamy, S., Sankaranarayanan, K., & Tarjan, D. (2003). Temperature-aware microarchitecture. In Proceedings of the 30th Annual International Symposium on Computer Architecture (pp. 2–13). https://doi.org/10.1109/ISCA.2003.1206984

[25] Schulman, J., Moritz, P., Levine, S., Jordan, M., & Abbeel, P. (2016). High-dimensional continuous control using generalized advantage estimation. In Proceedings of the International Conference on Learning Representations.

[26] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. https://doi.org/10.48550/arXiv.1707.06347

[27] Dudík, M., Langford, J., & Li, L. (2011). Doubly robust policy evaluation and learning. In Proceedings of the 28th International Conference on Machine Learning (pp. 1097–1104).

[28] Bastianelli, E., Vanzo, A., Swietojanski, P., & Rieser, V. (2020). SLURP: A spoken language understanding resource package. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (pp. 7252–7262). https://doi.org/10.18653/v1/2020.emnlp-main.588

[29] FitzGerald, J., Henchion, M., Unanue, W., Ciaramita, M., Suárez, A., Ahuja, K., Chen, C., Goenaga, D., Mugisha, A., Pruksachatkun, Y., Sitaram, S., Srinivasan, A., Wang, L., & Xia, P. (2023). MASSIVE: A 1M-example multilingual natural language understanding dataset with 51 typologically-diverse languages. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (pp. 4277–4302). https://doi.org/10.18653/v1/2023.acl-long.235

[30] Larson, S., Mahendran, A., Peper, J. J., Clarke, C., Lee, A., Hill, P., Kummerfeld, J. K., Leach, K., Laurenzano, M. A., Tang, L., & Mars, J. (2019). An evaluation dataset for intent classification and out-of-scope prediction. In Proceedings of the Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (pp. 1311–1316). https://doi.org/10.18653/v1/D19-1131

[31] Shen, Z., Zhao, W., Wang, B., Wang, Z., & Shang, W. (2026). CAGR: A Cross-Accelerator Graph Optimization Framework for Efficient Recommender System Inference. IEEE Access. https://doi.org/10.1109/ACCESS.2026.3671798

[32] Zhang, H., Ge, Y., Zhao, X., & Wang, J. (2025). Hierarchical Deep Reinforcement Learning for Multi-Objective Integrated Circuit Physical Layout Optimization With Congestion-Aware Reward Shaping. IEEE Access. https://doi.org/10.1109/ACCESS.2025.3610615