FLOWDVFS: OUTPERFORMING GENERATIVE REPLAY FOR FEW-SHOT DVFS CONTROL

Anonymous authors
Paper under double-blind review

ABSTRACT

Few-shot reinforcement learning for dynamic voltage and frequency scaling (DVFS) must find, from a few dozen real transitions, the clock giving the highest stable frame rate without throttling. We propose FlowDVFS, a flow-matching replay generator for a few-shot Double DQN: a physics anchor backed up at every clock, plus a flow residual. We judge the generator by a learning curve over 8 to 32 tuples on a throughput reward whose thermal target never binds in the replayed runs. On held-out traces of a Jetson Orin NX and TX2, at MBPO's twenty updates per real transition, FlowDVFS needs fewer real tuples than SynthER, PGR (one update) and MBPO and is above all three at every budget, tested at each against MBPO. Neither replay method cuts the learner's real budget; FlowDVFS does, its anchor's argmax with no learner matching it on development, while the top clock, best on every run, stays above every learned method.

1 INTRODUCTION

The rapid advancements in large-scale generative models have transformed data generation, particularly in unstructured domains like computer vision for generating images and videos. Diffusion models Ramesh et al. (2022); Saharia et al. (2022); Croitoru et al. (2023) produce high-quality data and have drawn interest to their latent spaces. However, tabular generation Xu et al. (2019); Kotelnikov et al. (2023) assumes a fixed schema; semi-structured sensor data, which exhibits correlations and noise but lacks predefined relational structures, remains underexplored. This matters in few-shot reinforcement learning (RL), where models generalize from limited examples. It is essential for online RL over semi-structured sensor data, as in Dynamic Voltage and Frequency Scaling (DVFS), which sets the processor clock from readings of clock, temperature, power and frames per second (FPS) to hold throughput under a thermal limit. Such data are expensive: a transition is a second of board time under a workload that heats the die, so a learner needing thousands is not deployable on the device it controls.

We introduce FlowDVFS, a physically anchored flow-matching replay generator for a few-shot RL learner for DVFS. Unlike methods needing extensive datasets or a simulator, FlowDVFS is fitted to a few dozen real transitions and judged by how many the learner then needs. We achieve this through flow matching Lipman et al. (2023), which trains continuous normalizing flows (CNFs) simulation-free and is reported there to improve the efficiency and quality of generated data over diffusion models. By folding bootstrapping Efron (1992) into the fit as bagging over eight heads, we regularize a generator that sees a few dozen transitions, though the learning curve resolves no gain from the tuned fit. We also anchor the flow on a per-unit physical prior of frame rate, power and temperature against the clock, so the flow models only the residual. Though the Law of Large Numbers Hastie et al. (2009) inspires the distribution approximation, non-i.i.d. synthetic data limits strict convergence, so we rely on the policy's learning curve over real transitions as the test.

FlowDVFS addresses the limitation of generative replay Lu et al. (2023); Wang et al. (2025b), which adds a generator's draws to the replay buffer and, we find, does not make the learner few-shot. On Jetson Orin NX and TX2 held-out traces, SynthER (Synthetic Experience Replay) and PGR (Prioritized Generative Replay) leave the learner needing, on average, at least as many real tuples as unaugmented, on both boards. By backing up the learner's targets through the anchored generator at every clock, FlowDVFS is above SynthER, PGR and MBPO (Model-Based Policy Optimization) Janner et al. (2019) at every budget, tested at each against MBPO, and cuts the real tuples it needs on both boards. It stays below the top clock, best on every recorded run. This matters in DVFS, where the controller must hold the highest stable frame rate without thermal throttling, which a poorly

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

chosen action produces on the Orin NX and, under a lowered trip, on the TX2. At MBPO's and SynthER's twenty updates per real transition, FlowDVFS attains the unaugmented learner's full-budget reward with 9.2 and 12.3 fewer real tuples than SynthER, 24.7 and 14.6 fewer than PGR at one update, and 7.4 and 4.6 fewer than MBPO. The anchor's argmax with no learner gives the same curves, so the gain is the anchor's prior, not the flow or the learner.

2 PRELIMINARIES

This section introduces the concepts behind our approach, a flow matching generator filling a few-shot RL learner's replay, with emphasis on sample efficiency. We focus on semi-structured data, such as our DVFS case study's sensor readings, which exhibit complex, non-grid-like patterns. We cover Q-learning, regret bounds, the Law of Large Numbers (LLN), and the flow matching algorithm.

2.1 Q-LEARNING AND ONLINE RL

Q-learning learns $\mathcal{Q}(s, a)$, the expected *return* (cumulative discounted reward) for action a in state s under an optimal policy thereafter. For continuous states and discrete actions, a **Deep Q-Network (DQN)** Mnih et al. (2015) approximates $\mathcal{Q}(s, a; \mathcal{W})$ with network parameters $\mathcal{W}$. The update rule is: $\mathcal{Q}(s_t, a_t; \mathcal{W}) \leftarrow \mathcal{Q}(s_t, a_t; \mathcal{W}) + \alpha \left[r_t + \gamma \max_{a'} \mathcal{Q}(s_{t+1}, a'; \mathcal{W}^-) - \mathcal{Q}(s_t, a_t; \mathcal{W})\right]$ where r_t is the immediate reward, $\gamma \in (0, 1)$ is the discount factor, and $\mathcal{W}^-$ the parameters of a slower-moving *target network*.

Online Setting. We focus on **online** RL, where the agent continuously interacts with the environment to refine its policy π. At each time-step, it observes (s_t, a_t, r_t, s_{t+1}) and updates $\mathcal{Q}$; in our case study a_t is the CPU clock, and on the Jetson Orin NX the client exposes fan speed as a second lever. In applications like DVFS, frequent decisions and gradual environmental changes allow treating each time-step as a horizon of $\mathcal{H} = 1$, so our learner discounts at zero, its target network copied every ten steps. This simplification is common in resource-management tasks; our reward follows the frame rate under a temperature target, with a hard -1 at the target or when the thermal governor engages.

While Q-learning converges in tabular, finite-state settings under mild conditions, convergence in high-dimensional or partially observed environments is less assured. Deep networks as function approximators can bring instability or overfitting if sample diversity is limited.

2.2 REGRET BOUNDS AND SAMPLE EFFICIENCY

In RL, the **regret** after T steps is defined as:

$$\mathrm{Reg}(T) \;=\; \sum_{t=1}^{T} \Big[\, \mathcal{Q}^*(s, a) - \mathcal{Q}(s, a) \Big], \tag{1}$$

representing the gap between the agent's policy, using $\mathcal{Q}(s, a)$, and the optimal one, using $\mathcal{Q}^*(s, a)$. Sublinear regret, such as $O(\sqrt{T})$, is achievable in *finite-state*, well-explored environments Jin et al. (2018), indicating that per-step regret diminishes over time.

In continuous, high-dimensional settings like our DVFS case study, achieving theoretical regret bounds is challenging. If the function approximator or model class is misspecified (the true environment lies outside the assumed class), sublinear bounds may not hold. In practice, we rely on an empirical learning curve over real transitions rather than strict theoretical guarantees.

Model Realizability and LLN. Sublinear regret often depends on **realizability**—the assumption that the true dynamics or $\mathcal{Q}^*$ lies within the agent's hypothesis space. In practice, limited exploration (e.g., rarely sampled CPU frequencies in DVFS) or complex environments can keep neural networks from approximating the true dynamics, slowing or preventing convergence. The *Law of Large Numbers (LLN)* ensures that, with sufficiently *diverse* and *representative* data, empirical estimates (e.g., $\mathcal{Q}_T$ or transition model P_T) converge to their true values as $T \to \infty$. However, without effective exploration, convergence may remain incomplete.

2.3 FLOW MATCHING FOR GENERATIVE MODELING

Flow matching Lipman et al. (2023) models a continuous flow of probability density from a simple base distribution (e.g., $\mathcal{N}(0, I)$) at $\tau = 0$ to a target distribution at $\tau = 1$. The distribution evolves

via:

$$\frac{dx_\tau}{d\tau} \;=\; w(x_\tau, \tau), \tag{2}$$

where $w(\cdot, \cdot)$ is a time-dependent velocity field for sample vector x_τ at flow time τ. A parametric network v_θ approximates it, trained to minimize:

$$\min_\theta \mathbb{E}_{\tau\in[0,1],\, x_\tau\sim p_\tau}\Big[\, \|v_\theta(x_\tau, \tau) - w(x_\tau, \tau)\|^2\Big]. \tag{3}$$

Relevance to DVFS RL. We employ flow matching to draw synthetic successors for every real state at every clock from a few dozen real transitions, and back them up into the learner's targets. Anchored to a physics prior fitted on the same real transitions, the flow learns only the residual around it. Its training objective is a squared-error regression on the velocity field, simulation-free, and the generator is judged, anchor and residual alike, by how many real tuples the learner needs to match one trained on real data alone.

3 Design of FlowDVFS: Anchored Flow Matching for Replay Generation

In this section, we describe *FlowDVFS*, a physics-anchored flow matching generator that fills a few-shot RL learner's replay, so a DVFS controller reaches stable throughput under a thermal limit from few real transitions.

3.1 Redefinition of the Regret Function

To evaluate synthetic data's impact in RL, we redefine the cumulative regret to reflect the Bellman operator's estimation error under combined real and synthetic data, inspired by the model-free RL framework. Let $\mathcal{M}^*$ denote the true Markov Decision Process (MDP), with optimal Q-function $\mathcal{Q}^*$. Let π be the policy learned using real data $\mathcal{D}_R$ (size n) and synthetic data $\mathcal{D}_S$ (size m) from flow matching. The cumulative regret up to time T is defined as:

$$\mathrm{Reg}(T) = \sum_{t=1}^{T} [\mathcal{Q}^*(s_t, a_t) - \mathcal{Q}^\pi(s_t, a_t)] + T\, O\Big(\sigma\sqrt{\tfrac{\log(1/\delta)}{n+m}}\Big), \tag{4}$$

where $\mathcal{Q}^\pi$ is the Q-function under policy π, σ^2 bounds the variance of rewards and Q-values ($\mathrm{Var}[r + \gamma \max \mathcal{Q}] \le \sigma^2$), and δ is the confidence parameter. The second term accumulates over the T steps the estimation error of the empirical Bellman operator due to finite real and synthetic samples, at that rate only for independent synthetic draws. This makes synthetic data's role in policy-update variance explicit.

3.2 Flow Matching and Bootstrapping

We augment a few dozen real transitions with generated successors approximating the board's dynamics, so the learner needs fewer real transitions. While generated successors are not i.i.d. draws from the board, backing them up at every clock raises the learner's throughput reward at every budget on both boards of the development split.

Flow matching models how a sample's probability density evolves over a continuous-time parameter $\tau \in [0, 1]$, distinct from the decision step t. Like a learned environment model, FlowDVFS draws the successor s_{t+1} from a conditional distribution $p(s_{t+1} \mid s_t, a_t)$ centred on a physics anchor. In its *conditional* form, the model learns a velocity field $v_\theta(x_\tau; z)$ to match a target field $w(x_\tau; z)$ Tomczak (2024), where x_t is the anchor residual at flow time τ and $z = (s_t, a_t)$ the condition. Equation 3 is the baseline objective.

Bootstrapping by Bagged Heads. To mitigate overfitting on a few dozen rows, we introduce *bagged bootstrapping* of the fit, inspired by Efron (1992). For the n residual rows we draw, for each of B heads, a Dirichlet weight vector w^b with concentration α and total weight n:

$$w^b \sim n\,\mathrm{Dirichlet}(\alpha \mathbf{1}_n), \quad b = 1, \ldots, B. \tag{5}$$

Let $x_\tau^b = (1-\tau)\, x_0 + \tau\, x_1 + \sigma\epsilon$ denote a point on the straight path from a Gaussian source x_0 to a row x_1 drawn under w^b, with distribution $p_\tau^b(x)$ and target velocity $w(x_\tau^b) = x_1 - x_0$ Tong et al. (2024). The bootstrapped flow matching objective extends Equation 3:

$$\min_{\theta_1,\ldots,\theta_B} \frac{1}{B}\sum_{b=1}^{B} \mathbb{E}_{\substack{\tau\in[0,1] \\ x_\tau^b\sim p_\tau^b(x)}} \Big[\|v_{\theta_b}(x_\tau^b) - w(x_\tau^b)\|^2\Big]. \tag{6}$$

With $B = 8$ and $\alpha = 4$, the heads are routed in balance at sampling time, row i of a query integrated by head $i \bmod B$.

3.3 Physics Anchor and Conditional Flow Matching

Bootstrapping controls variance, but a *physics anchor* holds every generated successor to the board's fitted response to the clock, as required in our DVFS case study.

Physics Anchor. We fit, on the run's own rows, a physics anchor $\mu_\phi(s_t, f_a)$ for frame rate, power and temperature against the commanded clock f_a in GHz. The $d = 3$ anchored channels follow a saturating frame-rate law, a power law that bends at a critical frequency f_k, after CMOS voltage scaling and the critical-speed law Burd & Brodersen (1996); Jejurikar et al. (2004), and a first-order thermal response, while the clock channel is set to f_a:

$$\begin{aligned} \mathrm{fps} &= \frac{F_{\mathrm{sat}}\, f_a}{f_a + f_{1/2}}, \qquad \theta' = \theta + (\alpha_\theta + \beta_\theta P - \theta)\,\lambda, \\ P &= \sum_k w_k \big(p_{0,k} + [f_a,\ f_a h_k,\ f_a h_k^2]\, x_k\big), \quad h_k = \max(0, f_a - f_k),\ x_k \geq 0. \end{aligned} \tag{7}$$

The knee f_k ranges over zero and the run's clocks, each candidate is fitted by non-negative least squares, and the fits are averaged with weights $w_k \propto \mathrm{SSE}_k^{-n/2}$. The anchor is fitted at the first generation event, on the rows the learner holds. No channel or generated row is weighted by a fitted importance: a development proxy measured such weighting as harmful, so the feature-weighted bootstrap is off in the deployed generator.

Anchored Conditional Objective. We train the flow on the anchor's residual in a *conditional* setup, with context $z = (s_t, a_t)$, the standardised state and one-hot clock, following the real rows' distribution $q(z)$. Define $y_c = (s_{t+1,c} - \mu_{\phi,c}(s_t, f_a) - \bar{r}_c)/\sigma_c$ as the standardised residual of channel c, and x_τ^b its bootstrapped sample at flow time τ. The conditional flow matching objective on the residual extends equation 6:

$$\min_{\theta_1,\ldots,\theta_B} \frac{1}{B} \sum_{b=1}^{B} \mathbb{E}_{\substack{\tau \in [0,1] \\ x_\tau^b \sim p_\tau^b(y|z) \\ z \sim q(z)}} \Big[\big\| v_{\theta_b}(x_\tau^b; z + \eta\, \epsilon \odot m_s) - w(x_\tau^b; z) \big\|^2 \Big], \quad \epsilon \sim \mathcal{N}(0, I). \tag{8}$$

The condition is jittered by $\eta = 0.4$ standard deviations on the state channels only (the mask m_s), never on the clock, and the velocity weights are Polyak-averaged with decay 0.999.

3.4 Implications for RL: Regret Bounds under Model-Based Assumptions

We theoretically motivate synthetic data in a **model-based** RL framework. Let $\mathcal{M}^*$ be the true MDP that governs the DVFS transitions, and let $\widehat{\mathcal{M}}$ be the anchored flow, a model of its transitions fitted on the real rows. The model predicts s_{t+1} from (s_t, a_t), but at discount zero the learner regresses the one-step reward and the max over successor actions never enters the target, so the bounds below describe the discounted learner of the cited analyses and not the deployed one. Its successors fill the synthetic set, each real state backed up at every clock through M successors. The redefined regret accounts for policy suboptimality and finite-sample estimation error.

Enhanced Model Realizability. Generated successors augment the real rows to improve the Bellman backup's estimate. The empirical Bellman operator for $\widehat{\mathcal{M}}$ at a pair (s, a) averages the backed-up target over $\mathcal{D}_R(s, a)$ and $\mathcal{D}_S(s, a)$, the $n_{s,a}$ real and $m_{s,a}$ synthetic rows taken at that pair:

$$\begin{aligned} \widehat{\mathcal{T}}\mathcal{Q}(s,a) = \frac{1}{n_{s,a} + m_{s,a}} \Big(& \sum_{(s',r) \in \mathcal{D}_R(s,a)} [r + \gamma \max_{a'} \mathcal{Q}(s', a')] \\ & + \sum_{(s'',r) \in \mathcal{D}_S(s,a)} [r + \gamma \max_{a'} \mathcal{Q}(s'', a')] \Big). \end{aligned} \tag{9}$$

Under bounded variance ($\mathrm{Var}[r + \gamma \max \mathcal{Q}] \leq \sigma^2$), a Bernstein-type concentration bound Boucheron et al. (2013) gives, with probability $1 - \delta$:

$$\big| \widehat{\mathcal{T}}\mathcal{Q}(s,a) - \mathcal{T}\mathcal{Q}(s,a) \big| \leq O\left(\sigma \sqrt{\frac{\log(1/\delta)}{n_{s,a} + m_{s,a}}} \right) + O\left(\frac{\log(1/\delta)}{n_{s,a} + m_{s,a}} \right). \tag{10}$$

This matches the regret's second term only if the generated successors were independent draws from $\mathcal{M}^*$. Then synthetic data would reduce estimation error by a larger sample size, consistent with sublinear regret $\mathrm{Reg}(T) = O(\sqrt{T})$ under a realizable model class Fan & Ming (2021); otherwise a bias term is added.

Illustrative Sketch. A simplified argument is:

1. Each (s, a) pair is sampled with probability $\mu > 0$, so $n_{s,a} + m_{s,a} \approx \mu(n+m)$: the backup queries every clock at every real state.
2. The learned model $\widehat{\mathcal{M}}$ satisfies the above concentration bound.
3. By standard analyses Jin et al. (2018); Fan & Ming (2021), regret satisfies:

$$\mathrm{Reg}(T) = O(\sqrt{T}) + T\,O\left(\sigma\sqrt{\frac{\log(1/\delta)}{n+m}}\right). \tag{11}$$

Independent draws from $\mathcal{M}^*$ would make the contribution of m suggest sublinear regret at a reduced real-sample complexity.

Practical Caveats. In a thermally constrained setting like DVFS, exploration is limited, and mismatch may arise if the flow misses corner cases such as the onset of throttling. The anchor bounds that mismatch, and the throttling event enters from real rows only, as the hard -1.

3.5 Implications for RL: Regret Bounds with Synthetic Data in Model-Free RL

In a model-free RL framework, synthetic data augments real data to reduce Q-value variance, tightening the regret bound for independent draws. The learner never plans on the model: the successors are stored with their rewards in $\mathcal{D}_S$ and enter every Q-learning update, G per real transition (the update ratio), as a second loss term of equal weight.

Role of Synthetic Data via Flow Matching. The anchored flow aims at the regret bound by covering the state-action pairs a few dozen real transitions never visit. The bagged objective (Equation 6) controls variance, while the anchored conditional objective (Equation 8) preserves the board's frame-rate, power and temperature response to the clock. This shrinks the variance term only for independent draws. On the development split the anchor alone matches the anchored flow within a rung on both boards, so the sample-efficiency gain is attributed to the anchored backup, not the flow's residual.

Bias arises if synthetic tuples deviate from the true distribution. The anchor bounds that bias; on the development split the augmented learner reaches the control's full-budget quality sooner on both boards and a higher quality at every budget.

3.6 From LLN to Finite-Sample Guarantees

Under mild mixing, independent synthetic samples reduce the variance of empirical Bellman updates, yielding finite-sample bounds for both paradigms.

Let $\mathcal{D}_R$ be the n real transitions and $\mathcal{D}_S$ the m synthetic ones, backed up as in Equation 9. With $\mathrm{Var}[r + \gamma \max \mathcal{Q}] \le \sigma^2$, the Bernstein-type bound of Equation 10 Boucheron et al. (2013) holds at every state-action pair with probability $1 - \delta$. Adding m synthetic samples shrinks the variance term only for independent draws, which the flow's are not, so we judge the generator by the real budget n at which the learner reaches a stated quality.

3.7 Algorithmic Overview for Few-Shot Online RL

We integrate the anchored flow with a Double Deep Q-Network (DQN) whose action is the CPU clock (the Orin NX fan level held fixed here), using two replay buffers: $\mathcal{D}_{\mathrm{real}}$ for real tuples (s_t, a_t, r_t, s_{t+1}) and $\mathcal{D}_{\mathrm{syn}}$ for backed-up successors.

In each iteration, we send the clock and store the returned tuple (s_t, a_t, r_t, s_{t+1}) in $\mathcal{D}_{\mathrm{real}}$. The throughput reward is $r_t = 2(\mathrm{fps}_t/\mathrm{fps}^\star - 1) + 0.1\tanh(\theta_{\mathrm{t}} - \theta_t)$, with $\mathrm{fps}^\star = 30$ and $\theta_{\mathrm{t}} = 70°\mathrm{C}$ on the Orin NX and $60°\mathrm{C}$ on the TX2. Its thermal term is -1 when the temperature reaches θ_{t} (the learning-curve read sets the whole reward to -1 there), and the whole reward -1 when the governor engages (the throttling event, a non-zero cooling-device state). At the first generation event we fit the anchor and the heads on $\mathcal{D}_{\mathrm{real}}$ by Equation 8; at every event we draw $M = 16$ successors at every clock of every distinct real state, stored with their rewards in $\mathcal{D}_{\mathrm{syn}}$. Each real transition is

Algorithm 1 FlowDVFS as the replay generator of a few-shot Double DQN. The anchored flow is fitted at the first generation event and backed up at every clock of every real state at each event; the throttling event e_t (the governor engaging) is the hard -1.

Initialize: $\mathcal{D}_{\text{real}}, \mathcal{D}_{\text{syn}}$, Double DQN with update ratio G, anchor μ_ϕ, residual flow heads $\{v_{\theta_b}\}_{b=1}^{B}$
for $t = 1$ **to** T **do**
  Send the clock a_t (on the Orin NX with a fixed fan level); receive s_{t+1} and the throttling event e_t
  $r_t \leftarrow$ the throughput reward of s_{t+1}, its thermal term -1 if $\theta_{t+1} \geq \theta_{\text{t}}$; $r_t \leftarrow -1$ if $e_t = 1$
  Store (s_t, a_t, r_t, s_{t+1}) in $\mathcal{D}_{\text{real}}$
  if $t \in \{32, 64, 96\}$, a generation event **then**
    Fit μ_ϕ and the B heads by Equation 8 on $\mathcal{D}_{\text{real}}$ if not yet fitted; empty $\mathcal{D}_{\text{syn}}$
    for each distinct real state s in $\mathcal{D}_{\text{real}}$ and each clock a **do**
      Draw M successors $\hat{s}' = \mu_\phi(s, f_a) + \bar{r} + \sigma \odot \Delta_\psi(s, a)$, clock channel f_a ▷ anchor plus residual
      Store $(s, a, r(\hat{s}'), \hat{s}')$ in $\mathcal{D}_{\text{syn}}$
    end for
  end if
  if $t \geq 31$: take G DQN updates, each on a real mini-batch plus the mean loss over $\mathcal{D}_{\text{syn}}$, equal weight
end for

followed by $G = 20$ updates, each fitting the Q-function on a real mini-batch plus the mean loss over $\mathcal{D}_{\text{syn}}$, weighted equally. At discount zero the backed-up targets are computed once per event. The residual flow trains for 2,000 Adam steps at learning rate 3×10^{-4}, and the generator is judged only by the real transitions the learner needs.

4 EXPERIMENTS

We present the setup, the tuning, the learner, and comparisons with generative replay and MBPO.

Experimental Setup We built a Double Deep Q-Network (Double DQN) agent on the zTT framework Kim et al. (2021) to reach the maximum stable frame rate under a thermal constraint on one Jetson Orin NX and one TX2 (Appendix C), under one workload, H.264 decode. The goal is the clock, and later the Orin NX fan, maximising frame rate without throttling. The client reports $\{FPS, f, \rho, \theta\}$ (frame rate, clock, power, temperature) and the throttling flag; the server returns the clock and, on the Orin NX, a fixed fan level (Figure 1). A run trains for 100 one-second steps; its frozen policy scores on 90 rows. A step engaging the governor or reaching the temperature target (70°C on the Orin NX, 60°C on the TX2) pays -1; on the boards the thermal term carries it. In a trace-driven simulator replaying recorded runs, we train FlowDVFS, its anchor alone, the control, SynthER, PGR and MBPO on a run's first 8, 16, 24 and 32 real tuples. Updates and generation events are fixed across rungs; n^* is the smallest budget reaching the control's quality at 32.

Hyperparameter Tuning. We tuned the generator's fit on the development split to 2,000 Adam steps at 3e-4, Polyak 0.999, jitter 0.4 and eight Dirichlet-bagged heads (Table 10). Tuning does not change the read: tuned and untuned generators need the same budget on 57 of 60 Orin NX and 27 of 30 TX2 runs. Future work reads the ladder on the boards, where the thermal term binds and the fan level is learned.

Model Architecture and Training. The Double DQN agent van Hasselt et al. (2016) has two 128-unit hidden layers, trained with the Huber loss and Adam at 10^{-3} on standardised inputs. An ϵ-greedy strategy starts at 1.0 and decays; the policy freezes at step 100. Training runs in intervals: 20 Q-network updates at each of steps 31–99 (1,380, MBPO's and SynthER's count), and a generator fitted once at step 32 backing up every clock of every real state at 32, 64 and 96. Each head's velocity network has three 64-unit SELU layers. Quality is the mean throughput reward over 90 rows scored in a leave-run-out simulator from the board's other recorded runs, the thermal-violation fraction beside it; power is a diagnostic. The reward function, adapted from Kim et al. (2021), is:

$$R = \begin{cases} u + v & \text{if } \theta < \text{target_temp and no throttling,} \\ -1 & \text{otherwise,} \end{cases} \tag{12}$$

where

$$u = 2\left(\frac{FPS}{\text{target_FPS}} - 1\right), \tag{13}$$

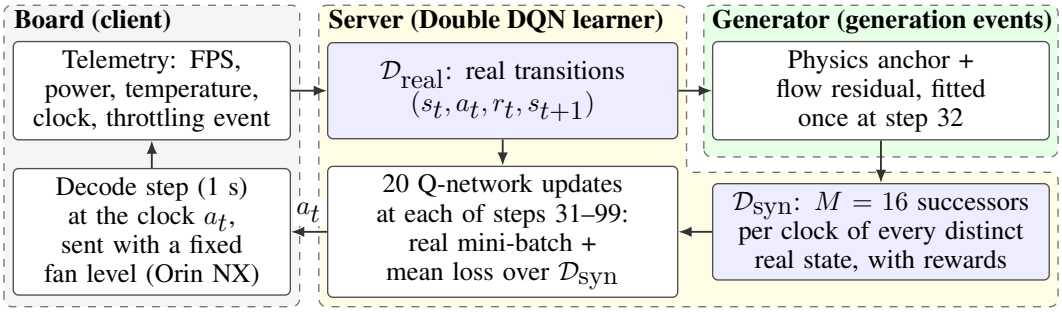

Figure 1: FlowDVFS in the deployed loop: at the generation events (steps 32, 64 and 96) the anchored flow backs up every clock of every distinct real state, $M = 16$ successors each into the learner's $\mathcal{D}_{\text{syn}}$; the learner takes 20 updates at each of steps 31–99 on $\mathcal{D}_{\text{real}}$ and $\mathcal{D}_{\text{syn}}$. The read replays recorded runs of this loop in a trace-driven simulator.

	Jetson Orin NX (60 runs)			Jetson TX2 (30 runs)		
Method	n^*	q_8	q_{32}	n^*	q_8	q_{32}
No augmentation (control)	20.1	15.77	18.04	21.1	6.10	6.86
Physics anchor (ours)	16.0†	19.52	19.67	9.1†	7.83	7.88
FlowDVFS, untuned fit (ours)	16.5†	19.48	19.67	12.8†	7.84	7.85
FlowDVFS (ours)	15.5†	19.48	19.66	10.9†	7.84	7.86
SynthER (matched)	23.7	15.67	18.15	21.6	6.12	6.98
PGR	36.5	14.42	16.51	25.3	5.99	6.76
MBPO	20.4	17.33	19.34	19.5	6.96	7.61
Anchor argmax, no learner (ours)	12.7	19.70	19.70	8.0	7.91	7.91
Top clock held (reference)	–	19.70	19.70	–	7.91	7.91

Table 1: Mean n^* in buffer tuples (64 when never reached) and mean quality q_n at $n = 8$ and 32 per method and board, development split; the threshold is the control's quality at 32 at the same update count; MBPO and SynthER run at their own update ratio of 20 per real transition, PGR at one; anchor argmax: the anchor's argmax clock held with no learner, tested against FlowDVFS only (Appendix A.3); reference: the top clock, the best fixed clock on every run, no tuples; untuned fit: the recipe before tuning; matched: SynthER's width set to the board trainer's earlier flow, 536,330 weights against the deployed generator's 79,384; †: n^* below the control's (exact one-sided Wilcoxon, Holm across boards, $p < 0.03$).

and

$$v = 0.1 \cdot \tanh(\text{target_temp} - \theta). \tag{14}$$

With target FPS 30 and target temperature 70°C on the Orin NX and 60°C on the TX2, this rewards frame rate only on a cool, unthrottled board; on the boards crossing the target sets $v = -1$ rather than R.

Model-Based and Model-Free Baselines. We compare FlowDVFS with SynthER Lu et al. (2023), diffusion-based generative replay; PGR Wang et al. (2025b), prioritized generative replay; MBPO Janner et al. (2019), model-based with an ensemble world model; the anchor alone; and the control. SynthER and MBPO run in their own integrations, PGR in ours at one update per real transition. SynthER carries 536,330 weights, matched to the board trainer's earlier flow rather than the deployed generator's 79,384, PGR and MBPO their own, at a tenth of the published diffusion schedule (Supplement Table 2), unchecked; the control uses no synthetic data. Every arm shares the learner, the reward, the run's tuples and the random numbers; contrasts are paired.

Results and Analysis. We fit a physics anchor and a *conditional flow matching* residual Lipman et al. (2023); Tomczak (2024) on the buffer's real transitions, then generate successors for every clock of every real state. We call the anchor without its residual "the physics anchor alone." We call the anchored flow with its tuned fit *FlowDVFS*; the untuned fit is a third arm. We judge each generator by its learner's learning curve (Figure 2), not its samples' fidelity. For an arm on run u

with quality $q(u, n)$ at n buffer tuples ($n - 1$ real transitions), the budget it needs is

$$n^*(u) = \min\{\, n \in \{8, 16, 24, 32\} : q(u, n) \geq q_{\text{ctrl}}(u, 32) \,\}. \tag{15}$$

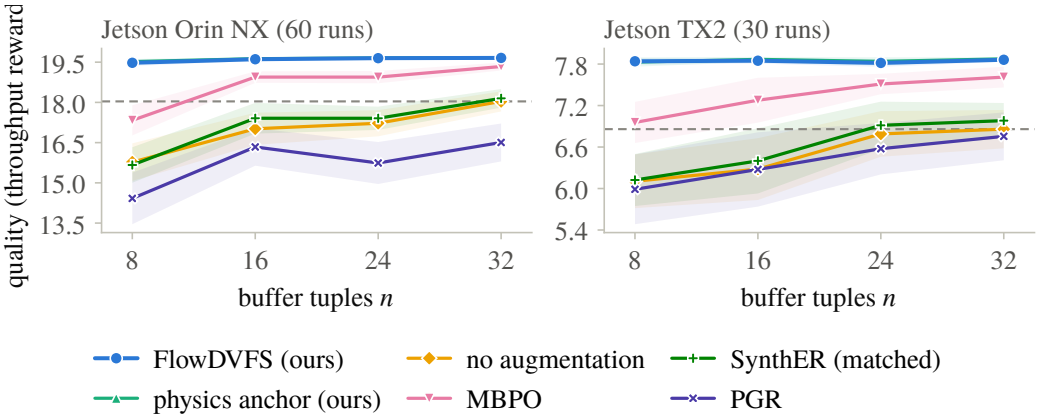

Figure 2: Learning curves on the development split, FlowDVFS against generative replay and MBPO: quality (throughput reward) against buffer tuples n, mean over runs with a 95% band, Orin NX (60 runs) and TX2 (30 runs); dashed: the control at $n = 32$, the threshold defining n^*; the anchor alone coincides with FlowDVFS at every rung. The untuned fit is in Supplement Figure 1.

On the development split, FlowDVFS reaches the control's full-budget quality from 4.7 and 10.1 fewer real tuples on the Orin NX and TX2 (Holm $p < 0.01$; the simulator's median reward error is 0.05 and 0.01, Appendix A.1). Its quality exceeds the control's at every budget, by 3.70 at 8 and 1.62 at 32 on the Orin NX, 1.74 and 1.00 on the TX2 ($p < 0.0001$). FlowDVFS owes this to the anchor: its argmax alone, no learner, reaches the same budget on 52 of 60 Orin NX and 27 of 30 TX2 runs and a smaller one on 7 and 3 (Appendix A.3).

Table 1 contrasts the methods.

FlowDVFS reaches the control's quality from 8.3 and 10.7 fewer real tuples than SynthER on the Orin NX and TX2, 21.1 and 14.4 fewer than PGR at one update (uncorrected $p \leq 0.0002$). This holds at a 90% bar (Appendix A.6). Its quality exceeds SynthER's by 3.81 and 1.72 at 8, 1.51 and 0.88 at 32, and PGR's by 5.06 and 1.85, 3.15 and 1.11 ($p < 0.0001$). Neither replay method is few-shot here: SynthER needs 3.6 and 0.5 more tuples than the control, PGR 16.4 and 4.3 more. PGR's quality is below the control's at every budget; SynthER's mean passes it from 16 tuples on the Orin NX and at 8 on the TX2.

To examine the thermal half, which replayed transitions never bind, we drive each board into the event by stepping the Orin NX's fan down and heating the TX2 (Appendix B).

A poorly chosen action produces the event: at the top clock, fan at its floor, the Orin NX crossed its 99°C trip at step 163 and both governors engaged (Supplement Figure 2). Both clusters fell from 1.98 GHz to 1.15 and 1.11 GHz, the frame rate from 150 to about 100, the learner paid -1 on 287 of 450 rows. The TX2's own event is the GPU governor's alone, so a cut of the CPU clock needs the passive trip lowered to 85°C. Under that trip the GPU governor engaged, the A57 clock fell from 2.03 to 1.73 GHz, and the learner paid -1 on 79 of 300 rows.

Performance Comparisons. We swept the fan, the Orin NX's other lever, from 2987 to 6556 rpm at the top clock, each level held until settled or 300 s (Supplement Figure 4).

The fan is a lever: the CPU zone falls from 84.9°C to 62.9°C across its range, with all nine zones, and the module rail by 0.756 W, attributed to leakage. Its own draw, modelled at 0.75 W, reaches no rail; the net is within 0.1 W.

At the same 1,380 updates FlowDVFS needs 4.9 and 8.5 fewer real tuples than MBPO on development (Holm $p \leq 0.005$) and leads it at every budget, by 2.15 and 0.88 at 8, 0.32 and 0.25 at 32. The control at 20 updates stays below MBPO at 32 on both boards (Table 1), so the update count alone does not explain the gain. On 148 held-out Orin NX and 82 TX2 runs, the test fixed beforehand confirms every contrast (Appendix A.4). FlowDVFS needs 7.4 and 4.6 fewer real tuples than MBPO, 9.2 and 12.3 fewer than SynthER, 24.7 and 14.6 fewer than PGR at its one update, 6.8 and 9.9 fewer than the control (Holm $p < 0.0001$). Its quality at 32 is above MBPO's and below the

top clock's, best fixed clock on every run, by 0.04 and 0.05 (Appendix A.4). At a binding target the quality advantage holds, the budget advantage does not (Appendix A.3).

5 Related Work

Machine Learning for DVFS. Machine learning is widely used for Dynamic Voltage and Frequency Scaling (DVFS) and scheduling Yu et al. (2020); Kim & Wu (2020); Dinakarrao et al. (2019); Zhuo et al. (2021); Pagani et al. (2018); Shen et al. (2012); Wang et al. (2017); Yeganeh-Khaksar et al. (2020); Liu et al. (2021); ul Islam & Lin (2015); Wang et al. (2021); Bo et al. (2021). The survey of Pagani et al. (2018) covers learning-based power and thermal management, and zTT Kim et al. (2021), like Panda et al. (2024), optimises a model-free reinforcement learning (RL) policy for power-performance trade-offs. However, these methods rely on extensive data collection and rarely state how many real transitions they need. Hierarchical multi-agent Pivezhandi et al. (2026c) and graph-driven Pivezhandi et al. (2026b) approaches show promise for DAG workloads. Our work introduces FlowDVFS, a flow-matching replay generator for a few-shot clock controller under a thermal limit.

Statistical Learning for DVFS. Statistical learning evaluates hardware events and timing statistics of a task Sasaki et al. (2007); Cazorla et al. (2019); Liu et al. (2021). For example, Sasaki et al. (2007) selects the clock within a task from hardware events, while Cazorla et al. (2019) surveys probabilistic worst-case timing analysis. Liu et al. (2021) ranks compiler-level features for a thermal-aware scheduler that also sets the clock. These efforts model a task from its traces and leave open what a controller learns from a few dozen transitions. FlowDVFS instead fits a physics anchor to the learner's own few dozen real transitions.

Bounding Sample Efficiency. Sample-efficiency bounds are a focus of theory Foster & Rakhlin (2023); Auer et al. (2008); Fan & Ming (2021); Jin et al. (2018). Sublinear bounds exist for finite state-action spaces Auer et al. (2008); Jin et al. (2018); at our horizon of one the clock choice is a structured bandit, nearest to best-arm identification Audibert et al. (2010); Lattimore & Szepesvári (2020), and the learner is the deployed controller compared against replay generators. Bounds for continuous control Fan & Ming (2021) hold only under distributional assumptions, and continuous spaces need robust approximators. Deep Q-learning with experience replay and target networks carries few theoretical guarantees Foster & Rakhlin (2023). Unlike these, FlowDVFS measures sample efficiency as the real tuples a learner needs to reach a stated quality.

Few-Shot RL. Few-shot RL methods, transfer, meta-learning and model-based, aim to reduce data needs Wang et al. (2020); Nachum et al. (2018); Florensa et al. (2017), the first two from a source task one board and one workload lack, as do imitation and inverse RL Arora & Doshi (2021); Wulfmeier et al. (2015); Reddy et al. (2019). MAML Finn et al. (2017) adapts with minimal data, PILCO Deisenroth & Rasmussen (2011) is the classical data-efficient model-based controller, and the residual-physics line Johannink et al. (2019); Ramesh & Ravindran (2023); El Asri et al. (2024) corrects a physical model with a learned residual, as our anchor does. Model-based methods Moerland et al. (2023); Janner et al. (2019); Wu et al. (2022) and generative replay Lu et al. (2023); Wang et al. (2025b); Kong et al. (2025); Wang et al. (2025a); Lim et al. (2026) synthesise buffer transitions, which van Hasselt et al. (2019) finds no better than replay from observed states, while backing up unvisited actions, Dyna's pattern Sutton (1991), can pay off. DVFS works Lin et al. (2023); Kim et al. (2021); Zhou & Lin (2021); Zhang et al. (2024) build learning-based governors and rarely report their learner's real transitions. Others address task placement, alone Pivezhandi et al. (2026a) or jointly with the clock Pivezhandi et al. (2026d). FlowDVFS advances this with a physics-anchored replay generator that, at MBPO's update-to-data ratio Chen et al. (2021); Fedus et al. (2020); D'Oro et al. (2023), needs fewer real tuples than SynthER, MBPO or PGR at one update and leads all three at every budget, tested at each against MBPO, a gain the anchor alone delivers.

6 Conclusion

We presented FlowDVFS, a flow-matching replay generator for few-shot DVFS control under a thermal limit. Backing up a physics-anchored flow at every clock cuts the real tuples needed. On held-out traces it outperforms SynthER and MBPO at their update count and PGR at one, but stays below the top clock, best on every run. The anchor's argmax alone gives the same gain and the flow adds nothing. Future work trains the fan and reads that ladder on the boards.

Reproducibility Statement

Every number in this paper is computed from telemetry logged on the two boards or from the trace-driven replay built from it, except the fan's own draw, which no rail reports and which is modelled from its rating. The archive we release carries the recorded runs (the transitions table and each run's banked buffer with its seed), the replay harness that trains the deployed learner on a run's first n tuples and scores its frozen policy in the leave-run-out simulator, the generator code of FlowDVFS and of every compared method (SynthER and MBPO in their own integration, PGR in ours), the decision script that produced the numbers file behind the learning-curve tables and figure, and the clients and trainer that record the throttling event on the boards. The instrument, its ladder and its primary test were fixed and committed before the cells ran; the training reward was amended once, its commit landing as the first amended cells were already running, and disclosed before the read of record. The development split was read under the superseded energy reward, once as fixed beforehand and once post hoc against a 90% bar, and under the reward of record as its arms landed. The learner's update ratio, 20 per real transition, was then chosen on the development split against a test fixed before its cells ran: fewer real tuples than MBPO on each board under Holm, and quality at $n = 32$ not below MBPO's. The choice was confirmed once on the held-out split of 148 Orin NX and 82 TX2 runs under the same test, committed before any of FlowDVFS's cells ran, the comparators' held-out cells being those of the earlier confirmation, reused unchanged: every contrast, against MBPO, the control, SynthER and PGR, held on both boards (Appendix A.4). The comparison against replay is read at twenty updates per real transition on both splits; the earlier read at one update and its own held-out confirmation are reported in the supplement as history, and the archive carries the cells of both reads with their markers. The learning-curve and event figures are regenerated by archive scripts from the recorded data; Appendix A's learning-curve tables are transcribed from the decision script's numbers files, its ablation table from the earlier read's file and the cell records, its event table from the trainer records. An anonymized public mirror of the repository is planned; until it is public, the statement that the instrument and its tests were fixed before their cells ran rests on the repository's own history.

Ethics Statement

This work involves no human subjects and no personal data; all experiments run an H.264 video-decode workload on the authors' own embedded development boards, with GPU inference workers added as heat in one event run. The boards' measured CPU-rail draw spans roughly 4–9 W and their total-board draw about 13–19 W during the event runs (Appendix B); we do not characterise the footprint further because the cell records carry the host's training time and not its energy, and we decline to call the total modest without measuring it. We foresee no harmful application specific to this method beyond generic DVFS control, and fixing the instrument's primary test before its cells ran, disclosing each amendment, and reporting every null beside the result it accompanies are intended to support research-integrity norms.

AI Use Statement

In this work, we used generative AI tools (large-language-model, LLM, assistants) for code implementation and refactoring of the training and experiment infrastructure, experiment orchestration, statistical-analysis scripting, diagnostic analyses and the exploration of model variants on units held out for that purpose, literature-search assistance, and manuscript drafting and editing, all under author direction. We did not use generative AI to generate, impute, or modify experimental data: every reported number derives either from telemetry logged on physical hardware or from the trace-driven replay built from that telemetry and validated against held-out board runs, with marker-verified provenance; the primary numbers of Appendix A follow the instrument fixed before the cells ran and its replay comparison the read that the confirmation registers, and the supplement's earlier studies, some from exploratory diagnostics, are descriptive. AI-assisted code and analyses were reviewed by the authors, the figures were checked against the numbers file by a separate pass, and the event numbers were reconciled against the trainer records. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

References

Saurabh Arora and Prashant Doshi. A survey of inverse reinforcement learning: Challenges, methods and progress. *Artificial Intelligence*, 297:103500, 2021.

Jean-Yves Audibert, Sébastien Bubeck, and Rémi Munos. Best arm identification in multi-armed bandits. In *COLT*, pp. 41–53, 2010.

Peter Auer et al. Near-optimal regret bounds for reinforcement learning. *NeurIPS*, 21, 2008.

Zitong Bo et al. Developing real-time scheduling policy by deep reinforcement learning. In *RTAS*, pp. 131–142, 2021.

Stéphane Boucheron, Gábor Lugosi, and Pascal Massart. *Concentration Inequalities: A Nonasymptotic Theory of Independence*. Oxford University Press, 2013.

Thomas D. Burd and Robert W. Brodersen. Processor design for portable systems. *Journal of VLSI Signal Processing*, 13(2–3):203–221, 1996.

Francisco J Cazorla et al. Probabilistic worst-case timing analysis: Taxonomy and comprehensive survey. *CSUR*, 52(1):1–35, 2019.

Xinyue Chen et al. Randomized ensembled double Q-learning: Learning fast without a model. In *ICLR*, 2021.

Florinel-Alin Croitoru et al. Diffusion models in vision: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(9):10850–10869, 2023.

Marc Peter Deisenroth and Carl Edward Rasmussen. PILCO: A model-based and data-efficient approach to policy search. In *ICML*, pp. 465–472, 2011.

Sai Manoj Pudukotai Dinakarrao et al. Application and thermal-reliability-aware reinforcement learning based multi-core power management. *JETC*, 15(4):1–19, 2019.

Pierluca D'Oro et al. Sample-efficient reinforcement learning by breaking the replay ratio barrier. In *ICLR*, 2023.

Bradley Efron. Bootstrap methods: another look at the jackknife. In *Breakthroughs in statistics: Methodology and distribution*, pp. 569–593. 1992.

Zakariae El Asri et al. Physics-informed model and hybrid planning for efficient dyna-style reinforcement learning. *arXiv preprint arXiv:2407.02217*, 2024.

Ying Fan and Yifei Ming. Model-based reinforcement learning for continuous control with posterior sampling. In *ICML*, pp. 3078–3087, 2021.

William Fedus et al. Revisiting fundamentals of experience replay. In *ICML*, pp. 3061–3071, 2020.

Chelsea Finn et al. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, pp. 1126–1135, 2017.

Carlos Florensa et al. Stochastic neural networks for hierarchical reinforcement learning. *arXiv preprint arXiv:1704.03012*, 2017.

Dylan J Foster and Alexander Rakhlin. Foundations of reinforcement learning and interactive decision making. *arXiv preprint arXiv:2312.16730*, 2023.

Trevor Hastie et al. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. 2009.

Michael Janner et al. When to trust your model: Model-based policy optimization. In *NeurIPS*, volume 32, pp. 12519–12530, 2019.

Ravindra Jejurikar et al. Leakage aware dynamic voltage scaling for real-time embedded systems. In *DAC*, pp. 275–280, 2004.

Chi Jin et al. Is q-learning provably efficient? *NeurIPS*, 31, 2018.

Tobias Johannink et al. Residual reinforcement learning for robot control. In *ICRA*, pp. 6023–6029, 2019.

Seyeon Kim et al. zTT: Learning-based DVFS with zero thermal throttling for mobile devices. In *Annual International Conference on Mobile Systems, Applications, and Services (MobiSys '21)*, pp. 41–53, 2021.

Young Geun Kim and Carole-Jean Wu. Autoscale: Energy efficiency optimization for stochastic edge inference using reinforcement learning. In *MICRO*, pp. 1082–1096, 2020.

Lingkai Kong et al. Composite flow matching for reinforcement learning with shifted-dynamics data. In *NeurIPS*, 2025.

Akim Kotelnikov, Dmitry Baranchuk, Ivan Rubachev, and Artem Babenko. TabDDPM: Modelling tabular data with diffusion models. In *International Conference on Machine Learning*, pp. 17564–17579, 2023.

Tor Lattimore and Csaba Szepesvári. *Bandit Algorithms*. Cambridge University Press, 2020.

Soon Hoe Lim et al. Is flow matching just trajectory replay for sequential data? *arXiv preprint arXiv:2602.08318*, 2026.

Chengdong Lin et al. A workload-aware DVFS robust to concurrent tasks for mobile devices. In *Annual International Conference on Mobile Computing and Networking (ACM MobiCom '23)*, 2023.

Yaron Lipman et al. Flow matching for generative modeling. In *ICLR*, 2023.

Di Liu et al. Cartad: Compiler-assisted reinforcement learning for thermal-aware task scheduling and dvfs on multicores. *IEEE TCAD*, 2021.

Cong Lu et al. Synthetic experience replay. In *NeurIPS*, 2023.

Volodymyr Mnih et al. Human-level control through deep reinforcement learning. *Nature*, 518 (7540):529–533, 2015.

Thomas M Moerland et al. Model-based reinforcement learning: A survey. *Foundations and Trends® in Machine Learning*, 16(1):1–118, 2023.

Ofir Nachum et al. Data-efficient hierarchical reinforcement learning. *NeurIPS*, 31, 2018.

Santiago Pagani et al. Machine learning for power, energy, and thermal management on multicore processors: A survey. *IEEE TCAD*, 39(1):101–116, 2018.

Prasanta Panda et al. Reinforcement learning-based dynamic voltage and frequency scaling for energy-efficient computing. In *ICDCECE*, 2024.

Mohammad Pivezhandi et al. Feature-aware task-to-core allocation in embedded multi-core platforms via statistical learning. In *RTCSA*, pp. 102–113, 2026a.

Mohammad Pivezhandi et al. Graphperf-rt: A graph-driven performance model for hardware-aware scheduling of openmp codes. *arXiv preprint arXiv:2512.12091*, 2026b.

Mohammad Pivezhandi et al. Hidvfs: A hierarchical multi-agent dvfs scheduler for openmp dag workloads. *arXiv preprint*, 2026c.

Mohammad Pivezhandi et al. Zerodvfs: Zero-shot llm-guided core and frequency allocation for embedded platforms. *arXiv preprint arXiv:2601.08166*, 2026d.

Adithya Ramesh and Balaraman Ravindran. Physics-informed model-based reinforcement learning. In *Annual Learning for Dynamics and Control Conference*, volume 211, pp. 26–37, 2023.

Aditya Ramesh et al. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.

Siddharth Reddy et al. Sqil: Imitation learning via reinforcement learning with sparse rewards. *arXiv preprint arXiv:1905.11108*, 2019.

Chitwan Saharia et al. Photorealistic text-to-image diffusion models with deep language understanding. *NeurIPS*, 35:36479–36494, 2022.

Hiroshi Sasaki et al. An intra-task dvfs technique based on statistical analysis of hardware events. In *international conference on Computing frontiers*, pp. 123–130, 2007.

Hao Shen et al. Learning based dvfs for simultaneous temperature, performance and energy management. In *ISQED*, pp. 747–754, 2012.

Richard S Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *ACM SIGART Bulletin*, 2(4):160–163, 1991.

Jakub M Tomczak. *Deep Generative Modeling*. 2024.

Alexander Tong et al. Improving and generalizing flow-based generative models with minibatch optimal transport. *TMLR*, 2024.

Fakhruddin Muhammad Mahbub ul Islam and Man Lin. Hybrid dvfs scheduling for real-time systems based on reinforcement learning. *IEEE Systems Journal*, 11(2):931–940, 2015.

Hado van Hasselt et al. Deep reinforcement learning with double Q-learning. In *AAAI*, volume 30, pp. 2094–2100, 2016.

Hado van Hasselt et al. When to use parametric models in reinforcement learning? In *NeurIPS*, volume 32, 2019.

Bin Wang et al. Controllable flow matching for online reinforcement learning. *arXiv preprint arXiv:2511.06816*, 2025a.

Renhao Wang et al. Prioritized generative replay. In *ICLR*, 2025b.

Yaqing Wang et al. Generalizing from a few examples: A survey on few-shot learning. *ACM computing surveys (csur)*, 53(3):1–34, 2020.

Yiming Wang et al. Online power management for multi-cores: A reinforcement learning based approach. *IEEE Transactions on Parallel and Distributed Systems*, 33(4):751–764, 2021.

Zhe Wang et al. Modular reinforcement learning for self-adaptive energy efficiency optimization in multicore system. In *DAC*, pp. 684–689, 2017.

Philipp Wu et al. Daydreamer: World models for physical robot learning. In *CoRL*, 2022.

Markus Wulfmeier et al. Deep inverse reinforcement learning. *CoRR, abs/1507.04888*, 2015.

Lei Xu, Maria Skoularidou, Alfredo Cuesta-Infante, and Kalyan Veeramachaneni. Modeling tabular data using conditional GAN. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Amir Yeganeh-Khaksar et al. Ring-dvfs: Reliability-aware reinforcement learning-based dvfs for real-time embedded systems. *IEEE Embedded Systems Letters*, 13(3):146–149, 2020.

Zheqi Yu et al. Energy and performance trade-off optimization in heterogeneous computing via reinforcement learning. *Electronics*, 9(11):1812, 2020.

Ziyang Zhang et al. DVFO: Learning-based DVFS for energy-efficient edge-cloud collaborative inference. *IEEE Transactions on Mobile Computing*, 23(10):9042–9059, 2024.

Ti Zhou and Man Lin. Deadline-aware deep-recurrent-q-network governor for smart energy saving. *IEEE Transactions on Network Science and Engineering*, 9(6):3886–3895, 2021.

Cheng Zhuo et al. A dvfs design and simulation framework using machine learning models. *IEEE Design & Test*, 2021.

A THE LEARNING-CURVE INSTRUMENT AND ITS READ

A.1 THE INSTRUMENT

A *run* is one recorded 100-step episode of the deployed client on a board, with its seed; the development split holds 60 Orin NX runs and 30 TX2 runs, and the held-out split 148 and 82. A *cell* is the exact replay of the deployed learner on one run: 20 Double-DQN updates at each of the 69 steps from 31 to 99, 1,380 in all, the same three planning events at steps 32, 64 and 96, and the same real minibatch order. Its frozen policy at step 100 is evaluated by replaying the trainer's own evaluation state machine over the same scored rows the board records, three 30-step episodes, in a trace simulator that excludes the run's own rows. Each successor is drawn from the recorded rows of the board's other runs: one of the eight nearest in weighted standardized temperature, frame rate and power among the rows with the same action and clock, failing that with the same action alone. In the model-backup integration of FlowDVFS and of the anchor alone, each update is one real minibatch plus the mean loss over the backed-up set at equal weight, and the backed-up targets are computed once per planning event, since the discount is zero. SynthER and MBPO take the same 20 updates per step in their own integrations; PGR, in ours, takes one, 69 in all. The learning curve truncates the run's replay buffer to its first $n \in \{8, 16, 24, 32\}$ tuples and the $n = 32$ rung is the full cell; the learner's input is standardized with the run's own real states, its discount is zero, and the frozen policy acts greedily.

The *quality* of a cell is the mean over its 90 scored evaluation rows of the throughput reward $2(\mathrm{FPS}/30 - 1) + 0.1\tanh(\theta_{\mathrm{t}} - \theta)$, set to -1 where $\theta \geq \theta_{\mathrm{t}}$, with $\theta_{\mathrm{t}} = 70$°C on the Orin NX and 60°C on the TX2. On the boards the trainer replaces the commanded clock by a random lower one at or above the temperature target, a case no recorded transition reached. The thermal-violation fraction is reported beside it and never averaged in; it was 0.00 in every cell of the read, because the recorded transitions never cross the target, which is why the throttling event is measured on the boards (Section B). On the throughput reward itself, the simulator's median absolute error is 0.046 on the Orin NX (maximum 0.108, mean signed error -0.006, rank correlation 0.89) and 0.009 on the TX2 (0.039, $+0.008$, 0.98). That read, made once, replayed ten development runs per board whose scored rows hold a single action, that action fixed from the run's own start, so it bounds the instrument's error under a constant policy and not under the learned policies. The record's earlier validation is on CPU-rail energy per frame, the superseded objective: simulating ten development runs per board, five under the replayed policy and five under the run's final network, reproduced their board endpoints with a median absolute relative error of 0.62% on the Orin NX and 0.42% on the TX2. The budget $n^*(u)$ of a method on run u is the smallest rung at which its quality reaches the control's quality at $n = 32$ on the same run at the same update count, and 64 if it never does. The primary test, fixed before any cell ran, is whether n^* is below the control's per board (exact one-sided Wilcoxon on the per-run differences, zeros dropped, Holm across the two boards); the secondary reads the quality difference to the control at every rung, paired by run, with the same test. The comparison against generative replay uses the same tests on the same runs; SynthER, PGR and MBPO were named as arms when the instrument was fixed, but that contrast entered the decision script only at the read, after every cell had run. Its sixteen p-values in Table 6 are therefore uncorrected and descriptive, and the held-out read, whose test was fixed before its cells ran, decides it under Holm (Section A.4). Every cell of every method shares the run's real tuples, the network initialization and the evaluation random numbers.

A.2 THE READ ON THE DEVELOPMENT SPLIT

Every method here is read at 20 learner updates per real transition, the count MBPO and SynthER take in their own recipes; PGR, in our integration, takes one. Table 2 gives the mean quality per rung, Table 3 the budget n^* and its distribution, Table 4 the primary test, Table 5 the quality against the control at $n = 8$, $n = 32$ and over the ladder, and Table 6 the comparison against generative replay. The per-run standard deviation of quality at $n = 32$ is 0.13 and 0.07 for the tuned fit and 0.11 and 0.07 for the anchor alone on the Orin NX and TX2. The physics anchor alone, FlowDVFS with the untuned fit and FlowDVFS with its tuned fit are three separate methods in the read; the paper's FlowDVFS is the tuned fit. The untuned fit trains the residual flow for 20,000 Adam steps at learning rate 10^{-3} with one head, no Polyak averaging and no conditioning jitter; the tuned fit is 2,000 steps at 3×10^{-4} with Polyak 0.999, jitter 0.4 and Dirichlet(4) bagging over eight heads. The ablations of the flow's recipe, read at the same count, are in Table 10 (Section A.5) and, per rung,

in the supplement; each ties the tuned fit within a rung on both boards. The earlier read of the same instrument at one update per real transition, and its held-out confirmation, are in the supplement.

Table 2: Mean quality (throughput reward) of the frozen policy per budget n of buffer tuples at 20 updates per real transition (PGR at one), development split (60 Orin NX runs, 30 TX2 runs). Higher is better; the best fixed clock per run averages 19.70 on the Orin NX and 7.91 on the TX2.

Method	Board	$n = 8$	$n = 16$	$n = 24$	$n = 32$
No augmentation (control)	Orin NX	15.772	17.016	17.222	18.037
	TX2	6.105	6.281	6.789	6.859
Physics anchor alone	Orin NX	19.520	19.629	19.658	19.670
	TX2	7.828	7.868	7.839	7.875
FlowDVFS, untuned fit	Orin NX	19.481	19.541	19.657	19.673
	TX2	7.844	7.830	7.825	7.850
FlowDVFS, tuned fit (ours)	Orin NX	19.477	19.609	19.649	19.659
	TX2	7.840	7.848	7.816	7.862
MBPO	Orin NX	17.331	18.944	18.943	19.338
	TX2	6.956	7.277	7.512	7.611
SynthER (matched)	Orin NX	15.666	17.409	17.408	18.151
	TX2	6.123	6.400	6.914	6.982
PGR	Orin NX	14.420	16.338	15.740	16.506
	TX2	5.988	6.276	6.576	6.755

Table 3: The budget n^* at 20 updates per real transition: its mean over runs, the number of runs that never reach the control's full-budget quality (censored at 64), and the number of runs reaching it at each rung. Development split.

Method	Board	Mean n^*	Censored	Runs at 8 / 16 / 24 / 32 / never
No augmentation (control)	Orin NX	20.13	0	17 / 15 / 8 / 20 / 0
	TX2	21.07	0	8 / 5 / 7 / 10 / 0
Physics anchor alone	Orin NX	16.00	8	49 / 2 / 1 / 0 / 8
	TX2	9.07	0	27 / 2 / 1 / 0 / 0
FlowDVFS, untuned fit	Orin NX	16.53	8	46 / 4 / 2 / 0 / 8
	TX2	12.80	2	26 / 0 / 2 / 0 / 2
FlowDVFS, tuned fit (ours)	Orin NX	15.47	7	48 / 3 / 2 / 0 / 7
	TX2	10.93	1	27 / 1 / 0 / 1 / 1
MBPO	Orin NX	20.40	8	26 / 18 / 5 / 3 / 8
	TX2	19.47	4	17 / 5 / 2 / 2 / 4
SynthER (matched)	Orin NX	23.73	6	14 / 18 / 8 / 14 / 6
	TX2	21.60	2	9 / 6 / 8 / 5 / 2
PGR	Orin NX	36.53	24	9 / 14 / 7 / 6 / 24
	TX2	25.33	6	9 / 9 / 4 / 2 / 6

A.3 ATTRIBUTION

The tuned fit against the untuned fit, on development: on the Orin NX its n^* is lower on 3 runs and tied on 57 of 60 ($p = 0.125$), on the TX2 lower on 2 and tied on 27 of 30 ($p = 0.375$). The tuned fit against the physics anchor alone: lower on 1 run and tied on 58 on the Orin NX ($p = 0.50$), lower on none and tied on 28 on the TX2 ($p = 1.0$). The anchor against the tuned fit is lower on 1 and on 2 runs ($p = 0.75$ and 0.25). The flow residual therefore adds nothing the instrument measures over the anchor it is fitted around. These are ties at the ladder's resolution, not equivalence: with k non-tied pairs the exact one-sided test cannot return p below 2^{-k}, 0.125 with three and 0.25 with two, and no equivalence margin was declared. The isolating control, registered and read once on the development split, is the anchor fitted on the unit's first n tuples, its predicted throughput reward maximised over the clock table, and the argmax clock held with no learner and no synthetic buffer. From 8 tuples on, that argmax is the top clock on every unit (60 of 60 and 30 of 30), so the arm's quality is the top clock's, 19.70 and 7.91, above every learned arm at every rung; at 4 tuples it is the top clock on 55 of 60 and 25 of 30. Its budget is at or below FlowDVFS's on 59 of 60 Orin NX runs (lower on 7, tied on 52, higher on 1; the test that it is lower, on average ranks since the paired differences tie in magnitude, gives $p = 0.035$) and on all 30 TX2 runs (lower on 3, tied on 27; $p = 0.125$). The gain is therefore the anchor's prior: a monotone response law fitted on a few rows names the top clock, and the learner adds nothing the instrument resolves. The held-out half of this read was not made. A second read of the same recorded runs asks whether the result survives

Table 4: Primary test at 20 updates per real transition: n^* of the method below the control's, per board. Mean paired difference, runs where the method's budget is lower, ties, exact one-sided Wilcoxon p and its Holm adjustment across the two boards; development split.

Method	Board	Mean Δn^*	Lower	Ties	p	Holm
Physics anchor alone	Orin NX	-4.13	35	16	0.0278	0.0278
	TX2	-12.00	21	9	< 0.0001	< 0.0001
FlowDVFS, untuned fit	Orin NX	-3.60	35	15	0.0294	0.0294
	TX2	-8.27	20	8	0.0022	0.0043
FlowDVFS, tuned fit (ours)	Orin NX	-4.67	36	16	0.0091	0.0091
	TX2	-10.13	20	9	0.0002	0.0004
MBPO	Orin NX	$+0.27$	25	23	0.3104	0.5432
	TX2	-1.60	13	10	0.2716	0.5432
SynthER (matched)	Orin NX	$+3.60$	6	43	0.9857	0.9857
	TX2	$+0.53$	5	23	0.4531	0.9062
PGR	Orin NX	$+16.40$	14	6	1.0000	1.0000
	TX2	$+4.27$	9	11	0.8245	1.0000

Table 5: Secondary read at 20 updates per real transition: quality of the method minus the control's, paired by run, at $n = 8$, at $n = 32$ and averaged over the ladder (mean difference, exact one-sided Wilcoxon p after Holm across boards). Development split.

		$n = 8$		$n = 32$		ladder mean	
Method	Board	Δq	p	Δq	p	Δq	p
Physics anchor alone	Orin NX	$+3.7474$	< 0.0001	$+1.6333$	< 0.0001	$+2.6075$	< 0.0001
	TX2	$+1.7231$	< 0.0001	$+1.0166$	< 0.0001	$+1.3439$	< 0.0001
FlowDVFS, tuned fit (ours)	Orin NX	$+3.7044$	< 0.0001	$+1.6218$	< 0.0001	$+2.5866$	< 0.0001
	TX2	$+1.7353$	< 0.0001	$+1.0033$	< 0.0001	$+1.3332$	< 0.0001
MBPO	Orin NX	$+1.5592$	< 0.0001	$+1.3009$	< 0.0001	$+1.6272$	< 0.0001
	TX2	$+0.8507$	< 0.0001	$+0.7525$	< 0.0001	$+0.8303$	< 0.0001
SynthER (matched)	Orin NX	-0.1062	1.0000	$+0.1139$	0.0098	$+0.1465$	0.0559
	TX2	$+0.0183$	1.0000	$+0.1231$	0.0411	$+0.0962$	0.0559
PGR	Orin NX	-1.3523	0.9904	-1.5309	1.0000	-1.2607	1.0000
	TX2	-0.1164	0.9660	-0.1040	1.0000	-0.1099	1.0000

a constraint that binds. Scoring every fixed clock of every development unit at a grid of thermal targets shows the best fixed clock leaving the top clock when the target falls to 60°C on the Orin NX and 47°C on the TX2; the rule fixed before the read takes the highest whole-degree target at which that happens on at least a third of the units with the top clock violating on under a quarter of scored rows. At those targets the per-unit optimum is no longer a constant across runs: the modal best clock is 11 of 13 and 10 of 12, and the best fixed clock's mean quality is 18.963 and 7.616. Read there, the tuned fit's quality is above the control's at every rung on the Orin NX (by 2.71 at 8 and 0.74 at 32, Holm $p \leq 0.004$) and at 8 and 16 on the TX2 (0.92 and 0.91, $p \leq 0.0008$), and it reaches the control's full-budget quality from 8 tuples on 44 of 60 and 18 of 30 runs. Its budget advantage does not survive: it never reaches that quality on 12 and 6 runs, where the control always does, so its mean n^*, 20.40 and 20.80, is not below the control's 20.13 and 19.47. The anchored arms violate the constraint on 0.04 to 0.12 of scored rows against the control's 0.01 to 0.05. The gain is therefore in throughput, and under a binding thermal constraint it does not transfer to the budget the paper measures. Against MBPO, at the same 1,380 updates, Table 7 reads the two pre-stated contrasts. In it, d is the mean paired difference of n^* (negative favours FlowDVFS), tested by the exact one-sided Wilcoxon test that FlowDVFS's budget is lower, Holm across the two boards. q_{32} is the quality at $n = 32$, and p (lower) is the same test that FlowDVFS's q_{32} is below MBPO's, which the claim needs not to reject. FlowDVFS needs 4.93 and 8.53 fewer tuples than MBPO, and its quality at 32 is not below MBPO's on either board. Against the threshold of the earlier read, the one-update control's quality at 32, the budgets are 9.33 against 13.33 on the Orin NX and 12.27 against 21.33 on the TX2 (Holm $p < 0.0001$ and $p = 0.0001$); this reading is disclosed beside the other. Per rung, FlowDVFS's quality is above MBPO's by 2.145, 0.665, 0.706 and 0.321 at $n = 8$, 16, 24 and 32 on the Orin NX and by 0.885, 0.571, 0.305 and 0.251 on the TX2, paired one-sided $p \leq 0.0003$ at every rung. The physics anchor alone stands in the same relation to MBPO, 16.00 against 20.40 tuples and 9.07 against 19.47 (Holm $p < 0.0001$ and $p = 0.0001$), with its quality at 32 above MBPO's by 0.332 and 0.264.

Table 6: FlowDVFS (tuned fit) against generative replay on the same runs at 20 updates per real transition (PGR at one): the budget n^*, the budget $n^*_{0.9}$ to reach 90% of the run's best fixed-clock quality, and the quality at $n = 8$ and $n = 32$ (mean paired difference, runs in FlowDVFS's favour, exact one-sided Wilcoxon p, uncorrected across the sixteen contrasts). Development split.

		against SynthER (matched)			against PGR		
Measure	Board	mean	in favour	p	mean	in favour	p
Δn^*	Orin NX	-8.27	39	0.0001	-21.07	43	< 0.0001
	TX2	-10.67	19	0.0002	-14.40	19	< 0.0001
$\Delta n^*_{0.9}$	Orin NX	-16.13	44	< 0.0001	-20.80	44	< 0.0001
	TX2	-29.33	26	< 0.0001	-23.20	23	< 0.0001
Δq_8	Orin NX	$+3.811$	57	< 0.0001	$+5.057$	58	< 0.0001
	TX2	$+1.717$	28	< 0.0001	$+1.852$	30	< 0.0001
Δq_{32}	Orin NX	$+1.508$	46	< 0.0001	$+3.153$	57	< 0.0001
	TX2	$+0.880$	28	< 0.0001	$+1.107$	26	< 0.0001

Table 7: FlowDVFS against MBPO on the development split, both at 20 updates per real transition: mean n^* (buffer tuples, 64 when never reached), the mean paired difference d and the Holm-adjusted exact one-sided Wilcoxon p that FlowDVFS's is lower; mean quality at $n = 32$ and the p that FlowDVFS's is lower.

	n^*				q_{32}		
Board	FlowDVFS	MBPO	d	Holm p	FlowDVFS	MBPO	p (lower)
Orin NX	15.47	20.40	-4.93	< 0.0001	19.659	19.338	1.0000
TX2	10.93	19.47	-8.53	0.0050	7.862	7.611	0.9997

A.4 The confirmation on the held-out split

The test was fixed before any held-out cell ran and read once on the held-out runs (148 Orin NX, 82 TX2), with the instrument as read on development at 20 updates per real transition; PGR takes one. It asks that n^* of FlowDVFS be below MBPO's on each board (exact one-sided Wilcoxon, Holm across boards) and its quality at $n = 32$ not below MBPO's; it reads the same contrasts against the control, SynthER and PGR. No minimally important effect was declared, so each rejection establishes a direction, not a size, and every contrast holds (Tables 8 and 9). FlowDVFS needs 7.35 and 4.59 fewer real tuples than MBPO on the Orin NX and the TX2, 12.54 against 19.89 and 8.78 against 13.37 (Holm $p < 0.0001$ on both boards). Its quality at 32 is 19.662 against MBPO's 19.300 and 7.865 against 7.595; the test that it is lower gives $p = 1.0000$ and the test that it is higher $p < 0.0001$. Per rung it is above MBPO by 2.202, 0.631, 0.414 and 0.362 at $n = 8$, 16, 24 and 32 on the Orin NX and by 1.222, 0.368, 0.284 and 0.270 on the TX2 ($p < 0.0001$ at every rung). Against the control at the same 1,380 updates it needs 6.81 and 9.85 fewer tuples (Holm $p < 0.0001$). Against SynthER it needs 9.19 and 12.29 fewer and against PGR 24.70 and 14.63 fewer, $p < 0.0001$ on each board and under Holm across the boards. Its quality is above each of the three at $n = 8$ and at $n = 32$ under the test, and in the mean at every rung; the mean paired differences, Δq_8 and Δq_{32}, are in Table 9. The thermal-violation fraction was 0.00 in every cell. Two reads made after the confirmation, and not registered, complete the picture. The tuned fit is above SynthER and PGR at every rung on both boards (mean paired differences from 1.48 to 5.48 on the Orin NX and 1.06 to 2.15 on the TX2; $p < 0.0001$ at each rung). The best fixed clock is the top clock on every one of the 230 held-out runs; the tuned fit is below it by 0.041 and 0.050 at $n = 32$ (the Orin NX gap inside the simulator's median error of 0.046, the TX2's above its 0.009), at or above it on 83 of 148 and 39 of 82 runs, and the test that it is lower rejects ($p = 0.0056$ and $p < 0.0001$). The budgets are means of a score censored at 64, the runs never reaching the bar numbering 9 of 148 and 1 of 82 for the tuned fit, 21 and 2 for MBPO, 14 and 9 for SynthER and 61 and 12 for PGR. The censoring constant cannot move a median while fewer than half the runs are censored: the median n^* is 8 on both boards for the tuned fit, 16 and 8 for MBPO, 16 and 16 for SynthER, and 32 and 16 for PGR. Counted at 32 instead of 64 the tuned fit needs 4.76 and 4.20 fewer tuples than MBPO, 8.11 and 9.17 fewer than SynthER and 13.46 and 10.34 fewer than PGR, so 11.24 of the 24.70 against PGR come from the constant.

The attribution holds on these runs as on development. The tuned fit against the physics anchor alone is lower on 5 of the 148 Orin NX runs, tied on 137 and higher on 6; the test that it is lower gives $p = 0.7803$ and the test that the anchor is lower $p = 0.2490$. On the 82 TX2 runs it is lower

on none, tied on 81 and higher on 1 ($p = 1.0000$ and 0.5000). Against the untuned fit it is lower on 1 Orin NX run, tied on 142 and higher on 5; the test that it is lower gives $p = 0.9844$ and the test that the untuned fit is lower Holm $p = 0.0938$. On the TX2 it is lower on none, tied on 81 and higher on 1 ($p = 1.0000$ and 0.5000). The mean budgets are 12.38 for the anchor and 11.73 for the untuned fit against the tuned fit's 12.54 on the Orin NX, and 8.10, 8.10 and 8.78 on the TX2. The anchor alone stands in the same relation to MBPO as the tuned fit, needing 7.51 and 5.27 fewer tuples at Holm $p < 0.0001$, and its quality is above MBPO's at every rung ($p < 0.0001$). The control at the same 20 updates per real transition needs no fewer tuples than MBPO, 19.35 against 19.89 on the Orin NX and 18.63 against 13.37 on the TX2 (Holm $p = 1.0000$ on both). It is below MBPO at 32, 18.014 against 19.300 and 6.741 against 7.595, so the update count alone does not explain FlowDVFS's gain. One reading differs from development: on the TX2 held-out runs MBPO itself needs fewer tuples than the control, with a mean paired difference of n^* of -5.27 and Holm $p < 0.0001$, whereas on the Orin NX it does not ($+0.54$, Holm $p = 0.4985$). FlowDVFS's margin over MBPO stands on both boards. At the exploratory 90% bar of Section A.6, where the best fixed clock's quality on these runs is 19.703 and 7.914, FlowDVFS needs 8.05 and 8.10 tuples against the control's 20.81 and 39.32 and MBPO's 11.57 and 14.73 ($p < 0.0001$ against the control).

Table 8: FlowDVFS against MBPO on the held-out split (148 Orin NX runs, 82 TX2 runs), both at 20 updates per real transition: mean n^* (buffer tuples, 64 when never reached), the mean paired difference d and the Holm-adjusted exact one-sided Wilcoxon p that FlowDVFS's is lower; mean quality at $n = 32$ and the p that FlowDVFS's is lower.

	n^*				q_{32}		
Board	FlowDVFS	MBPO	d	Holm p	FlowDVFS	MBPO	p (lower)
Orin NX	12.54	19.89	-7.35	< 0.0001	19.662	19.300	1.0000
TX2	8.78	13.37	-4.59	< 0.0001	7.865	7.595	1.0000

Table 9: FlowDVFS against the control, SynthER (matched) and PGR on the held-out split (148 Orin NX runs, 82 TX2 runs) at 20 updates per real transition (PGR at one): mean paired difference of n^* (d) and of the quality at $n = 8$ and $n = 32$ (Δq_8, Δq_{32}); every exact one-sided Wilcoxon p is below 0.0001.

	against the control			against SynthER (matched)			against PGR		
Board	d	Δq_8	Δq_{32}	d	Δq_8	Δq_{32}	d	Δq_8	Δq_{32}
Orin NX	-6.81	$+3.388$	$+1.648$	-9.19	$+3.514$	$+1.477$	-24.70	$+5.484$	$+2.633$
TX2	-9.85	$+1.915$	$+1.124$	-12.29	$+1.897$	$+1.060$	-14.63	$+2.149$	$+1.207$

A.5 The ablation

Table 10 reads each change to the tuned fit on the same runs at 20 updates per real transition, the count of the paper's read, with the anchor alone and the untuned fit beside it. Polyak averaging keeps a running average of the velocity network's weights in place of its last iterate. Dirichlet bagging fits eight heads on Dirichlet-weighted copies of the residual rows. The two optimal-transport (OT) couplings pair the noise and residual rows of a minibatch by its transport plan, under a condition-aware cost or in residual space alone. The physics-consistency term (a resistor-capacitor, RC, consistency penalty) penalises generated temperature and power residuals whose one-step endpoint disagrees with the anchor's thermal law. Transport-cost admission draws twice the successors and keeps the half of lowest transport cost. The sampler steps are the Euler steps that integrate the residual from noise. Each contrast is the exact one-sided Wilcoxon test that the arm's n^* is below the tuned fit's, on the paired runs, with Holm across the two boards. Every variant ties the tuned fit: the mean paired differences of n^* lie within 1.87 tuples, 56 to 59 of 60 Orin NX runs and 27 to 29 of 30 TX2 runs tie, and no p is below 0.1875 (Holm 0.3750). Every variant, the anchor alone and the untuned fit included, needs fewer tuples than MBPO on both boards (Holm $p \leq 0.0073$); the tuned fit's own contrast with MBPO is Table 7. The quality at $n = 32$ lies between 19.625 and 19.673 on the Orin NX and between 7.836 and 7.881 on the TX2 for every arm in the table. None of the six changes to the flow moves the read, and the anchor alone ties the tuned fit on both boards ($p = 0.7500$ and 0.2500), so the gain over MBPO is already present with the anchor. The per-rung qualities, the contrasts against MBPO and the compute per cell are in the supplement, beside the same ablation at one update per real transition.

Table 10: Ablation of FlowDVFS at 20 updates per real transition, development split (60 Orin NX runs, 30 TX2 runs): mean n^* (buffer tuples, 64 when never reached), throughput reward q_{32} at $n = 32$, and the exact one-sided Wilcoxon p that the arm's n^* is below the tuned fit's.

		Orin NX		TX2		p vs. tuned	
Method	What changes	n^*	q_{32}	n^*	q_{32}	Orin	TX2
physics anchor (ours)	the anchor's mean successor only: no flow residual, deterministic	16.00	19.670	9.07	7.875	0.7500	0.2500
FlowDVFS, untuned fit (ours)	anchor plus flow residual, untuned: 20,000 Adam steps at 10^{-3}, one head, 50 Euler steps	16.53	19.673	12.80	7.850	1.0000	0.8750
FlowDVFS (ours)	the tuned fit: 2,000 Adam steps at 3×10^{-4}, Polyak 0.999, conditioning jitter 0.4 SD, eight bagged heads (Dirichlet 4)	15.47	19.659	10.93	7.862	ref.	ref.
tuned fit + OT coupling (condition-aware)	regression pairs from the minibatch's exact optimal-transport plan under a condition-aware cost ($\eta = 1$)	15.33	19.665	12.00	7.859	0.5000	1.0000
tuned fit + plain OT coupling	regression pairs from the exact optimal-transport plan in residual space alone ($\eta = 0$)	15.47	19.671	10.93	7.881	0.7500	0.6250
tuned fit + physics-consistency term	an RC-consistency penalty between the generated temperature and power residuals on the one-step endpoint ($\lambda = 0.1$)	14.67	19.648	11.47	7.878	0.5000	1.0000
tuned fit + transport-cost admission	$2n$ draws per query, the n of lowest transport cost kept (50th percentile), at sampling only	15.33	19.634	12.00	7.866	0.6250	1.0000
tuned fit, one sampler step	the same fit sampled with one Euler step instead of 50	14.67	19.625	11.47	7.840	0.5000	1.0000
tuned fit, four sampler steps	the same fit sampled with four Euler steps instead of 50	13.60	19.644	12.00	7.836	0.1875	1.0000

A.6 THE 90% READING

Because the control's full-budget quality is a bar that 17 of 60 Orin NX runs and 8 of 30 TX2 runs reach from 8 tuples, the same curves were also read against 90% of each run's best fixed-clock quality. That quality is the best over every fixed clock from the run's own start under its own random numbers, banked once per run. At that bar the tuned fit needs 8.13 tuples on average against the control's 23.73 on the Orin NX (40 runs lower, 20 tied, $p < 0.0001$; 0 and 10 runs censored). That mean is the first rung on all but one run, so this bar does not resolve the tuned fit's budget below 8. A rung added below the ladder, $n = 4$ (3 real transitions), read once on the development split at the primary bar, puts the tuned fit at the control's full-budget quality on 45 of 60 Orin NX and 21 of 30 TX2 runs, the anchor alone on 47 and 21, the control on 4 and 12; the tuned fit's mean quality there is 18.59 and 7.29 against the control's 12.47 and 5.82, so its budget is at or below 3 real transitions on three quarters of the runs and between 3 and 7 on most of the rest. On the TX2 it needs 8.00 against 36.00 (25 lower, 5 tied, $p < 0.0001$; 0 and 12 censored). MBPO needs 11.87 and 13.07, SynthER 24.27 and 37.33, PGR 28.93 and 31.20. This reading was added after a first pass of the curve had been read and before the cells reported here were read; it is exploratory.

B THE THROTTLING EVENTS ON THE BOARDS

The thermal term of the objective is not bound by the recorded transitions, so the event was provoked on both boards with clients that record, at every step, both cooling governors' states, the achieved

clock of every CPU cluster, the GPU load, the fan level (an index on a five-step pulse-width ladder, PWM, with its duty and speed) and the trip temperature, and a trainer that pays -1 on every step with a governor engaged. Table 11 summarizes the two induced runs, indexed by the trainer's one-second steps; the Supplement's figures show the ramp collector's own runs, in seconds, and the TX2's event under its factory trip is Supplement Figure 3. On the Orin NX the action was the top clock with the fan pinned at its lowest level under eight decode workers; the die reached the 99°C trip at step 163, both governors engaged, and the two clusters fell from 1.98 GHz to 1.15 and 1.11 GHz by the end of the run, the frame rate from 150 to about 100. On the TX2, whose CPU cooling device did not engage under its factory trip in any run, the passive trips were lowered to a labelled 85°C for the run; the GPU governor engaged at step 217, the CPU governor's state stayed 0, and the achieved clock of the A57 cluster nevertheless fell from 2.035 GHz to 1.73–1.88 GHz, so on this board the governor's state under-reports throttling of the CPU and the achieved clock is the second signal. Both clients restored stressors and governors at exit, the Orin NX its fan and the TX2 its trips. The fan on the Orin NX, swept from 40 to 100% of full PWM at the top clock under eight decode workers (Supplement Figure 4), moves the CPU zone from 84.9°C to 62.9°C, a 22.0°C range, with all nine zones falling together, while the module rail falls by 0.756 W, from 17.866 to 17.109 W as rounded, each level held until the temperature drifted below 0.25°C per minute or a 300 s cap; it is therefore an action on that board.

Table 11: The two throttling-event runs of the clients: the action that produced the event, the trip in force, the onset, the governors that engaged, the CPU clock under the event and the steps on which the learner paid -1.

	Jetson Orin NX	Jetson TX2
Action	top clock, fan at lowest level	top clock (2.035 GHz)
Load	eight decode workers	two decode workers per Denver core, two GPU workers
Trip in force	factory, 99°C	lowered to 85°C for the run
Onset	step 163	step 217
Governors engaged	CPU and GPU	GPU only (CPU state stayed 0)
CPU clock under the event	1.98 to 1.15 and 1.11 GHz, the two clusters	2.035 to 1.73–1.88 GHz, A57 cluster
Steps at -1	287 of 450	79 of 300

C THE BOARDS

The Jetson Orin NX carries two CPU clusters, an integrated GPU and a fan whose level is a commanded pulse-width value; its thermal target in the reward is 70°C and its factory trip 99°C. The Jetson TX2 carries a Denver cluster and an A57 cluster with an integrated GPU; its thermal target is 60°C and its factory trips 95.5°C. On both boards the action is an index into the board's discrete CPU clock table, thirteen levels on the Orin NX and twelve on the TX2, and the rendering application (H.264 decode) is the workload of every run in the read and of the events above.