Flux: Online, Fine-Grained Data Scheduling for Training Machine Learning Interatomic Potentials

Yuanchang Zhou[1,2] **Chen Wang**[1,2] **Hongtao Xu**[1] **Mingzhen Li**[1,2,†]
Guangming Tan[1,2] **Weile Jia**[1,2,†]
[1] State Key Lab of Processors, Institute of Computing Technology, CAS
[2]University of Chinese Academy of Sciences
{zhouyuanchang23s, limingzhen, jiaweile}@ict.ac.cn

Abstract

Large-scale training of machine learning interatomic potentials (MLIPs) increasingly relies on data parallelism over heterogeneous atomistic data. In this setting, atom count captures an important part of the per-rank workload, but structures with similar atom counts can still induce different graph workloads through variations in graph count, cutoff edges, higher-order geometric features, and graph-collation overhead. This makes conventional atom-count batching and offline load tables both incomplete and inflexible, especially when the model architecture, cutoff radius, or data mixture changes. We present Flux, an online workload-aware scheduler for distributed MLIP training. Flux estimates graph-size features from lightweight physical and geometric priors during data loading. It uses calibrated runtime and memory models to form balanced local mini-batches across data-parallel ranks while respecting memory constraints. Without requiring precomputed graph metadata, Flux can be used as a drop-in replacement for the standard distributed sampler, making it flexible and portable across training setups. Across large-scale atomistic datasets and three representative MLIP architectures, eSEN, MatRIS, and AllScAIP, Flux improves training throughput by 2.23–4.76$\times$ without degrading convergence.

1 Introduction

As machine learning interatomic potentials (MLIPs) scale to larger datasets and models, distributed training across tens or hundreds of GPUs has become the standard [1–6]. However, atomic data is inherently heterogeneous. A dataset can contain a mix of small molecules with a few atoms and massive periodic systems with thousands of atoms. Consequently, standard data-parallel training suffers from severe load imbalance. Complex structures assigned to a specific GPU can cause significant slowdowns, forcing other GPUs to idle at synchronization barriers. As training scales, this data-induced straggler effect has emerged as a major bottleneck for efficient MLIP training.

To mitigate this bottleneck, many pioneering works have explored sample reordering, bucketed batching, and workload-aware scheduling [7–9, 4]. A common approach uses precomputed atom counts as a lightweight proxy for computational load. However, atom count alone does not fully capture the workload of many state-of-the-art (SOTA) MLIPs [1, 10, 11, 4, 5]. Training cost can also depend on the numbers of graphs, edges, and higher-order interactions in a batch. Obtaining these quantities by constructing every graph requires dataset preprocessing, and the resulting load metadata may need to be recomputed when graph construction settings change. This raises a critical question: *Can we sense the true workload of an MLIP online and build a flexible, metadata-free scheduling framework around it?*

[†]Corresponding authors.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

However, using fine-grained workload information for online scheduling poses several challenges. Edge and higher-order feature counts depend on atomic neighborhoods, but full graph construction during scheduling incurs data preparation overhead and can cause GPU starvation. These graph-size features contribute differently to runtime and memory costs across MLIP architectures, requiring the performance model to account for architectural differences. Fixed-boundary batching further restricts sample assignment, making it difficult to balance execution times across GPUs when each device can accommodate only a few structures.

To address these challenges, we propose **Flux**, an online workload-aware scheduler for distributed MLIP training. Flux estimates graph-size features from lightweight physical and geometric priors during data loading, then uses these features to predict runtime and memory costs. Based on these predictions, Flux distributes workloads across data-parallel ranks to reduce stragglers and improve training throughput. Our contributions are summarized as follows:

1. We characterize why atom-count balancing is insufficient for modern MLIP training. Different architectures incur costs that depend on graph, atom, edge, and higher-order feature counts, which can leave substantial imbalance even when atom counts are balanced.

2. We introduce low-overhead online estimators for graph-size features. The estimators use physical and geometric priors to predict edge and higher-order feature counts during data loading, avoiding full offline graph construction and adapting to cutoff or dataset changes.

3. We design a runtime-calibrated performance and memory model together with a memory-aware windowed scheduler. This allows Flux to assign samples according to predicted local cost while respecting per-GPU memory constraints.

4. We implement Flux as a drop-in replacement for the standard distributed sampler. On large mixed MLIP workloads, Flux improves throughput by 1.16–1.98$\times$ at fixed global batch size and by 2.23–4.76$\times$ in the max-batch setting, without affecting convergence behavior.

2 Related work

Large-scale MLIP training. Atomistic datasets have grown from small, task-specific collections to large datasets with millions to more than 100 million atomistic structures [12, 13, 1, 3, 14, 15]. This growth has increased training costs and driven the use of multi-GPU training, often across multiple nodes.

Recent MLIPs [16, 17, 4, 5] often use multi-task training, combining data from different domains. With such heterogeneous data, training efficiency depends on model design, GPU kernel performance [18–20], and how samples are drawn, batched, and assigned across GPUs. This has motivated work on load balancing and data scheduling for atomistic datasets.

Workload-aware scheduling for MLIP training. Existing approaches mitigate load imbalance in atomistic MLIP training by optimizing batch composition and sample assignment [2, 7, 4]. However, these methods still face two key limitations.

First, they mainly rely on atom count as the single proxy for computational load. This proxy works well for neighborhood-padded architectures (i.e., intermediate representations with shape [number of atoms, maximum number of neighbors, hidden dimension]) [21, 2, 22], but is too coarse to capture the cost of interaction-flattened architectures [10, 11, 4, 5]. In these models, intermediate representations are flattened into two-dimensional edge-level or angle-level tensors (e.g., [number of edges, hidden dimension] or [number of angles, hidden dimension]), shifting the computational bottleneck to variable-length many-body operations. Therefore, the actual workload is inherently more fine-grained, jointly determined by structural quantities such as atoms, edges, and three-body terms.

Second, existing methods often rely on precomputed workload metadata and static scheduling policies. Edge and higher-order interaction counts may need to be recomputed when the cutoff radius or graph construction rules change. The computational cost associated with these features also varies across MLIP architectures, requiring updates to the workload model.

To overcome these limitations, we propose a dynamic data scheduling system. It combines a low-overhead online estimator for real-time feature counting with a fine-grained performance model to quantify workloads. This enables adaptive data scheduling without relying on offline metadata.

3 Observation

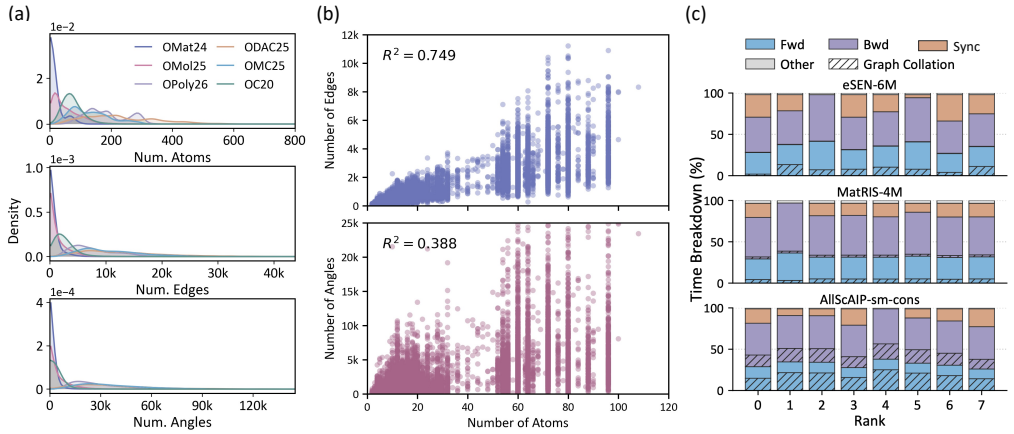

Figure 1: (a) Distributions of atoms, edges, and angles across large-scale atomistic datasets. The cutoff radius for edges and angles are 6.0 Å and 4.0 Å, respectively. (b) Correlation between atom count and the number of edges or angles, using 50k structures randomly sampled from OMat24. (c) Time breakdown of one training step for eSEN-6M, MatRIS-4M, and AllScAIP-sm-cons using standard atom-count-based load balancing. Here, "Graph Collation" denotes the graph preparation before model computation, including index shifting, setting gradient-related metadata, neighbor truncation, etc. "Other" includes loss computation and optimizer updates.

3.1 Heterogeneous atomic graph size distribution

As shown in Figure 1(a), large-scale atomistic datasets exhibit a long-tailed distribution in sample size. Even within a single data domain, the graph may mix small systems with only a few atoms and much larger periodic or molecular structures. For example, in OMat24 [1], about 50% of structures contain fewer than 12 atoms, whereas only 1% contain more than 88 atoms. Multi-task training [4, 16, 17, 6] further broadens this distribution by mixing domains with different length scales, such as ODAC25 [23].

Atom count, however, is only a coarse description of the workload seen by an MLIP. After cutoff graph construction, the number of edges and angles also depends on local density, cell geometry, periodic images, and the cutoff radius. Figure 1(b) shows that atom count is therefore an imperfect proxy for graph complexity: its R^2 with edge and angle counts is only 0.749 and 0.388, respectively. This mismatch motivates using finer graph-size features, rather than atom count alone, when modeling training load.

3.2 Training load goes beyond atom count

To examine whether the structural mismatch in atomistic graphs leads to system-level imbalance, we profile eSEN [24], MatRIS [5], and AllScAIP [22] under atom-balanced partitioning. As shown in Figure 1(c), balancing the number of atoms does not eliminate synchronization stalls. The maximum synchronization-time ratios still reach 31.8%, 16.6%, and 21.6% for eSEN, MatRIS, and AllScAIP, respectively. This indicates that atom count alone is insufficient to characterize the end-to-end training load.

The remaining load imbalance arises because the computational cost of MLIPs depends on structural properties beyond atom count, which vary across architectures. In eSEN, edgewise convolutions account for substantial computation, so batches with similar atom counts can have different runtimes

due to differences in edge count. MatRIS further processes three-body interactions through angle-indexed features, introducing additional workload associated with local neighbor pairs. AllScAIP combines neighborhood and all-to-all node attention with differentiable kNN graph construction [22, 25]. Its batch preparation also incurs per-graph overhead, which increases approximately with the number of graphs in a local batch (see Appendix A).

These differences make atom count alone an unreliable measure of training workload. Depending on the architecture, edge count, angular interactions, graph sizes, and per-graph preparation costs can contribute differently to execution time. Consequently, balancing batches using a single structural metric may still leave substantial workload differences across ranks.

These observations motivate a scheduler that combines structural features according to their contributions to each model's workload.

4 Methods

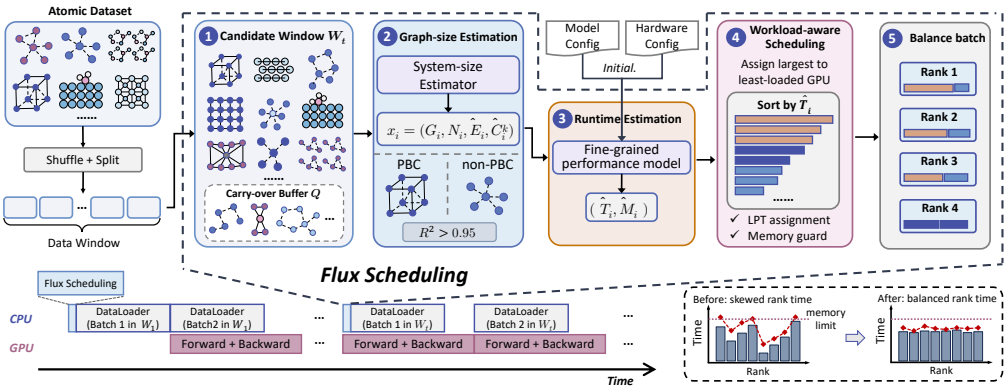

Figure 2: Overview of Flux. During data loading, Flux estimates graph-size features online, predicts runtime and memory with a calibrated performance model, and uses windowed scheduling to form balanced local mini-batches across DP ranks.

We propose Flux, an online workload-aware scheduler for distributed MLIP training. Figure 2 illustrates the overall workflow of Flux. During data loading, Flux estimates graph-size features (Section 4.2), maps them to runtime and memory costs (Section 4.3), and forms balanced mini-batches with workload-aware windowed scheduling (Section 4.4). Flux operates within the data-loading pipeline without modifying model computation. A PyTorch integration example is provided in Appendix G.

4.1 Problem formulation

Consider distributed MLIP training with R DP ranks. At each step, a global batch is partitioned into local mini-batches $\{B_r\}_{r=1}^{R}$. With synchronous gradient updates, the step time is determined by the slowest rank:

$$T_{\text{step}} \approx \max_{r \in \{1,\dots,R\}} T_{\text{local}}(B_r) + T_{\text{sync}}, \tag{1}$$

where $T_{\text{local}}(B_r)$ is the local execution time of rank r, and T_{sync} is the synchronization cost. Flux therefore aims to reduce the rank-level skew in T_{local}.

To capture architecture-dependent workloads, we represent each atomic structure i by a vector of structural counts:

$$x_i = \left(1,\, n_i,\, e_i,\, c_i^{(3)},\, \dots,\, c_i^{(K)}\right), \tag{2}$$

where the first entry denotes one graph, n_i is the atom count, e_i is the cutoff-edge count, and $c_i^{(k)}$ counts the k-body terms for $k = 3, \dots, K$.

For a local mini-batch B_r, the corresponding batch-level feature is additive:

$$x(B_r) = \sum_{i \in B_r} x_i = \left(G_r,\, N_r,\, E_r,\, C_r^{(3)},\, \ldots,\, C_r^{(K)} \right), \tag{3}$$

with

$$G_r = |B_r|, \quad N_r = \sum_{i \in B_r} n_i, \quad E_r = \sum_{i \in B_r} e_i, \quad C_r^{(k)} = \sum_{i \in B_r} c_i^{(k)}, \quad k = 3, \ldots, K. \tag{4}$$

Here, G_r, N_r, E_r, and $C_r^{(k)}$ are the graph, atom, edge, and k-body counts of batch B_r.

At each scheduling round t, Flux maintains a candidate window $\mathcal{W}_t$ of pending samples. The window can contain more samples than required for a single global batch, allowing Flux to select and assign samples based on their estimated workloads. The objective is to minimize the maximum predicted execution time across R ranks: $\min_{\{B_r\}_{r=1}^{R}} \max_r \hat{T}(B_r)$, subject to

$$\bigcup_{r=1}^{R} B_r \subseteq \mathcal{W}_t, \quad \left| \bigcup_{r=1}^{R} B_r \right| = B_{\text{global}}, \quad B_r \cap B_j = \emptyset, \quad \hat{M}(B_r) \le M_{\max}. \tag{5}$$

Here, B_{global} is the target global batch size, $\hat{T}(B_r)$ is the predicted execution time of rank r, and $\hat{M}(B_r)$ estimates its peak memory usage. Unselected samples remain in the buffer for subsequent scheduling rounds.

Flux performs sample selection and assignment on the CPU while the GPUs execute the current training step. To avoid delaying the next step, scheduling should complete within the current step:

$$T_{\text{sched}}(\mathcal{W}_t) \le T_{\text{step}}, \tag{6}$$

where $T_{\text{sched}}(\mathcal{W}_t)$ includes feature estimation, cost prediction, and sample assignment.

4.2 Online system-size estimator

To satisfy Eq. 6, graph-size features must be estimated without full graph construction. Constructing the graphs solely for scheduling would be too expensive. For a global batch of 512 OMat24 structures with $r_{\text{cut}} = 6.0$ Å, KDTree-based edge construction [26] takes about 1.0 s, compared with 0.4 s for an eSEN-6M training step under the same setting. For radius-cutoff graphs, inexpensive structural and geometric quantities, such as atom count, estimated coordination, and occupied volume, can serve as proxies for graph size. Flux uses lightweight estimators for periodic and nonperiodic structures, as well as higher-order interaction counts.

PBC edge count. For PBC systems, exact edge counting requires accounting for interactions with periodic images, with the cost depending on the lattice geometry, cutoff radius, and number of periodic images. For bulk-like structures, Flux estimates the average neighbor count using a uniform-density approximation:

$$\bar{d} \approx \frac{N}{V_{\text{eff}}} \cdot \frac{4}{3} \pi r_{\text{cut}}^3, \quad \hat{e} = N\bar{d}. \tag{7}$$

Here, $V_{\text{eff}} = V_{\text{cell}}$ for bulk structures. Systems with large vacuum regions, such as OC20 [12], or porous structures, such as ODAC25 [23], violate the uniform-density assumption. Flux therefore applies the corrections described in Appendix B.1. For bulk-like systems, the estimator replaces periodic image materialization and neighbor-list construction with an $O(1)$ density computation, and the corrected version remains lightweight for slab- and MOF-like systems.

Non-PBC edge count. For non-PBC systems, cutoff edges can be counted directly from pairwise Euclidean distances. Flux computes the squared distance matrix using a vectorized operation over atom pairs (a, b):

$$d_{ab}^2 = \|\mathbf{r}_a\|^2 + \|\mathbf{r}_b\|^2 - 2\mathbf{r}_a^\top \mathbf{r}_b \tag{8}$$

Here, $\mathbf{r}_a$ is atom a's position and $\mathbf{1}[\cdot]$ is the indicator function. Applying the cutoff and excluding self-pairs gives

$$\hat{e} = \sum_{a,b} \mathbf{1}\left[d_{ab}^2 \le r_{\text{cut}}^2 \right] - N \tag{9}$$

This CPU-vectorized method counts edges without constructing the neighbor list and is over $60\times$ faster than full graph construction in our measurements.

For MLIPs with explicit higher-order interactions, such as three-body features, Flux estimates $\hat{c}^{(k)}$ from local degree statistics using the corresponding model-specific cutoff. Details are provided in Appendix B.2.

4.3 Fine-grained performance model

Following the observation in Sec. 3.2, Flux combines the workload features shown in Fig. 2 ❷. Graph count accounts for per-structure collation overhead, while atom, edge, and higher-order interaction counts capture the computational costs of node-, edge-, and many-body operations, respectively.

Runtime estimation. We first estimate the computation time $\hat{T}(B)$ of a local batch B. For many MLIPs, such as MACE [27], eSEN, and MatRIS, the dominant computation consists of linear layers, normalization, element-wise operations, and scatter/gather aggregation. Under a fixed model configuration, the cost of these operators is approximately linear in the number of processed features. For MLIPs with self-attention modules, such as PET [2], EScAIP [21], and AllScAIP [22], local attention typically operates over the nearest M neighbors, with M usually less than 40. This keeps the quadratic cost of local attention relatively small compared with the overall computation. Flux therefore uses a calibrated linear model to predict the execution time of a local batch from its workload features:

$$\hat{T}(B) = \alpha_0 + \alpha_G G(B) + \alpha_N N(B) + \alpha_E \hat{E}(B) + \sum_{k=3}^{K} \alpha_k \hat{C}^{(k)}(B) \tag{10}$$

Here, α_0 captures batch-size-independent costs, including kernel launch and optimizer overhead. The coefficients α_G, α_N, α_E, and α_k correspond to graph, atom, edge, and k-body workloads, respectively. For each model and hardware configuration, Flux estimates these coefficients using linear least squares with a short profiling run, typically fewer than 200 iterations.

Memory estimation. Flux also estimates the memory required by each local batch to avoid exceeding the per-GPU memory budget. We decompose training memory into a static component and an input-dependent dynamic component. The static component includes model parameters, gradients, and optimizer states, and is approximately constant for a fixed model and optimizer. The dynamic component is dominated by activation tensors and intermediate features indexed by atoms, edges, and higher-order geometric terms.

For a local batch B, Flux estimates memory as

$$\hat{M}(B) = M_{\text{static}} + \beta_N N(B) + \beta_E \hat{E}(B) + \sum_{k=3}^{K} \beta_k \hat{C}^{(k)}(B). \tag{11}$$

Here, M_{static} denotes the static model memory, while β_N, β_E, and β_k capture the dynamic memory contribution of atom-, edge-, and k-body-indexed features. These coefficients are fitted from a short profiling phase using linear least squares, as in the runtime model. During scheduling, a candidate assignment is feasible only when $\hat{M}(B_r) \le M_{\max}$. We derive graph-size and memory-model bounds in Appendix C.2 and report memory usage on heterogeneous batches in Appendix C.3.

4.4 Windowed workload-aware scheduling

Flux uses a windowed greedy scheduler to balance workloads across DP ranks based on predicted runtime $\hat{T}$ and memory usage $\hat{M}$. As illustrated in Fig. 2 ❹, the scheduler selects samples from a candidate window $\mathcal{W}_t$ with a target size of $W_{\text{win}} \ge B_{\text{global}}$ and assigns them to DP ranks. Samples that are not assigned in the current step are retained in a carry-over buffer for subsequent scheduling rounds. Algorithm 1 describes the scheduling procedure.

At each scheduling round, Flux initializes the candidate window with samples carried over from the previous round and fills it with new samples from the shuffled stream (lines 1–2). It then estimates the workload features of each sample and sorts the candidates by decreasing predicted execution time (lines 3–4). Following the longest-processing-time-first (LPT) heuristic, Flux processes samples in

Algorithm 1 Windowed workload-aware scheduling with carry-over buffering

Require: Shuffled stream $\mathcal{D}$, carry-over buffer $\mathcal{Q}$, world size R, global batch size B_{global}, window size $W_{\text{win}} \geq B_{\text{global}}$, performance model $\hat{T}$, memory guard $\hat{M}$

1: $\mathcal{W} \leftarrow \mathcal{Q}$ and clear $\mathcal{Q}$
2: Fill $\mathcal{W}$ from $\mathcal{D}$ until $|\mathcal{W}| \geq W_{\text{win}}$ or $\mathcal{D}$ is exhausted
3: Estimate workload features x_i for each sample $i \in \mathcal{W}$
4: Sort samples in $\mathcal{W}$ by descending single-sample cost $\hat{T}(\{i\})$
5: Initialize $B_r \leftarrow \emptyset$ for each rank $r = 1, \ldots, R$
6: **for** sample i in sorted $\mathcal{W}$ **do**
7: **if** $\sum_{r=1}^{R} |B_r| = B_{\text{global}}$ **then**
8: Add i and the remaining sorted samples to $\mathcal{Q}$
9: **break**
10: **end if**
11: $\mathcal{R}_i \leftarrow \{r \mid \hat{M}(B_r \cup \{i\}) \leq M_{\max}\}$
12: **if** $\mathcal{R}_i$ is empty **then**
13: Add i to $\mathcal{Q}$
14: **else**
15: $r^\star \leftarrow \arg\min_{r \in \mathcal{R}_i} \hat{T}(B_r \cup \{i\})$
16: $B_{r^\star} \leftarrow B_{r^\star} \cup \{i\}$
17: **end if**
18: **end for**
19: **return** $\{B_r\}_{r=1}^{R}$ and updated carry-over buffer $\mathcal{Q}$

descending order of predicted cost. By assigning large samples first, the scheduler can better balance the remaining workload across ranks.

For each sample i, Flux selects the rank with the lowest predicted execution time after assignment, subject to the memory constraint:

$$r^\star = \arg \min_{r: \hat{M}(B_r \cup \{i\}) \leq M_{\max}} \hat{T}(B_r \cup \{i\}).$$

If no rank can accommodate the sample, it is moved to the carry-over buffer (lines 11–13). Once the target global batch size is reached, the remaining candidates are also retained for subsequent scheduling rounds (lines 7–8). After graph construction, Flux rechecks the memory constraints using exact graph counts (see Appendix C.2).

The window size controls the number of samples considered in each scheduling round. When $W_{\text{win}} = B_{\text{global}}$, the target window size matches one global batch, while a larger window provides more flexibility in workload balancing. The carry-over buffer retains unassigned samples for subsequent rounds. This flexibility is useful when each GPU can accommodate only a few structures. We discuss the scheduling overhead and window-size trade-off in Appendix D.

5 Evaluation setup

Dataset and model. We use a mixture of datasets used in Batatia et al. [16], Wood et al. [4], Kim et al. [17] (including OMat24, OMol25, OMC25, etc.). Specifically, we selected three representative state-of-the-art (SOTA) MLIPs for experiments. eSEN relies on edge-level message passing, MatRIS focuses more on high-order geometric features, while AllScAIP represents an attention-based architecture, whose neighborhood construction and data preparation overhead is not negligible. Model configurations are shown in Appendix E.

Hardware and environment. Unless otherwise stated, experiments were conducted on a cluster with 192 Intel Xeon Platinum CPU cores and eight NVIDIA H100 SXM 80 GB GPUs per node. We used the software environments specified by the respective official repositories.

We observed that the default automatic garbage collection (GC) introduced additional overhead during MLIP training. To reduce this overhead, we disabled automatic GC and triggered garbage collection manually every 10,000 iterations in all experiments.

6 Evaluation

We first evaluate the accuracy of the online estimator and performance model (Section 6.1), followed by ablation studies of the workload features and scheduling components (Section 6.2). We then measure end-to-end training performance (Section 6.3) and examine the effect of Flux on model convergence (Section 6.4).

6.1 Evaluation of estimator and performance model

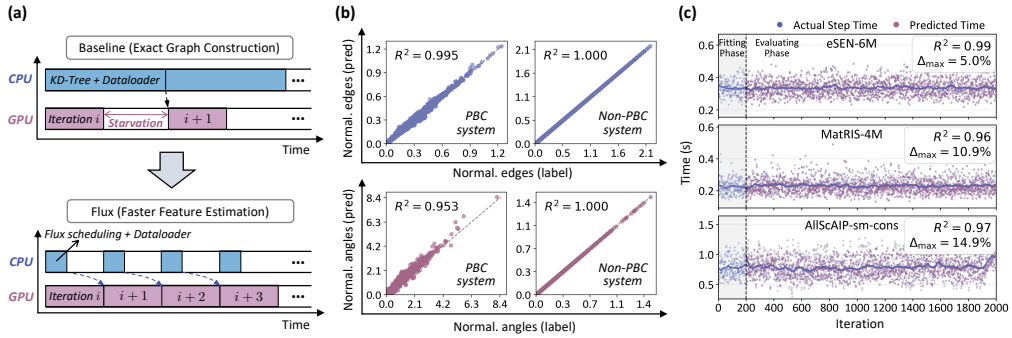

Figure 3: (a) Task execution pipeline using exact graph construction (KD-Tree based) and Flux estimation. (b) Comparison between Flux estimates and exact counts. For PBC systems, we randomly sample 50k structures from OMat24, while non-PBC systems are sampled from OMol25. (c) The comparison between the predicted time and the measured time of the fine-grained performance model. The curves are obtained by smoothing the points.

Flux estimator. To evaluate the Flux estimator, we trained eSEN-6M on OMat24 with a global batch size of 512, comparing the KD-Tree based exact graph construction against Flux. As shown in Figure 3 (a), exact graph construction took about 1.0 s for scheduling and data loading, while the GPU runtime was only 0.4 s, leading to GPU starvation. In contrast, the bulk-PBC estimator uses an $O(1)$ density calculation and reduces CPU overhead by roughly two orders of magnitude. Hence, data scheduling and loading can fully overlap with GPU execution without hindering training.

We also tested the accuracy of the Flux estimator on both PBC and non-PBC systems, as shown in Figure 3(b). For non-PBC systems, Flux estimates match the exact values, with $R^2 = 1$. For PBC systems, the R^2 values for edges and angles are 0.995 and 0.953, respectively. The slight degradation in PBC systems comes from the global average-density approximation: when the true coordination environment deviates from uniformity, edge estimation incurs small errors, which then accumulate and reduce the accuracy of angle estimates. Additional estimator results on OC20, OMC25, and ODAC25 are provided in Appendix B.3. We further evaluate the estimator under more extreme structural heterogeneity using a mixture of molecules, MOFs, and large MOF supercells in Appendix B.4.

Fine-grained performance model. To verify the accuracy of the Flux performance model component, we tested its performance on eSEN-6M, MatRIS-4M, and AllScAIP-sm-cons, as shown in Figure 3 (c). In practice, the model coefficients can be fitted using fewer than 200 profiling iterations. Across all three models, the R^2 values exceed 0.95, and the maximum relative error $\Delta_{\max}$ remains below 15%. The smoothed prediction curves closely follow the measured step-time trends, indicating that the performance model captures variations in training-step execution time. Appendix C.1 extends this analysis to additional MLIPs and includes memory prediction.

6.2 Ablation study

Ablation on performance model. To assess the effectiveness of fine-grained workload features, we conduct ablation studies on eSEN and AllScAIP. For each setting, the performance model is fitted using only the listed features in Table 1(a), while the remaining feature weights are set to zero. The results show that any single feature (Index 1/2/3) is insufficient to accurately predict end-to-end

Table 1: Ablation studies on feature. *: the weight for the edge feature is zero for AllScAIP, as its intermediate representation does not rely on edge count.

Index	Graph num	Atom count	Edge count	$R^2 \uparrow$	
				eSEN	AllScAIP
1	✓	✗	✗	0.255	0.474
2	✗	✓	✗	0.593	0.627
3	✗	✗	✓*	0.941	–
4	✓	✓	✓*	**0.999**	**0.967**

(a) Effectiveness of fine-grained feature modeling in the performance model.

Index	Atom counts	Exact edge counts	Estimated edge counts (Estimator)	Flux Perf. model	Speedup ↑	
					eSEN	AllScAIP
1	✓	✗	✗	✗	1.00×	1.00×
2	✗	✓	✗	✗	0.58×	–
3	✗	✗	✓*	✗	1.07×	–
4	✗	✗	✓*	✓	**1.09**×	**1.06**×

(b) Ablation study on fine-grained performance modeling for end-to-end training (32 GPUs).

iteration time. In contrast, using all relevant features (Index 4) yields the best fit, with R^2 values of 0.999 for eSEN and 0.967 for AllScAIP, substantially outperforming all single-feature variants.

Ablation on Flux components. To further analyze the contribution of each Flux component, we compare four scheduling methods: (1) using only atom counts (Baseline), (2) using exact edge counts obtained from KD-Tree construction, (3) using online estimated edge counts, and (4) the full Flux performance model. The baseline relies on atom counts alone for load balancing. Results are presented in Table 1(b). For eSEN, using exact edge counts (Index 2) leads to a longer total step time because exact edge computation and KD-Tree construction incur a substantial CPU overhead (see Figure 3(a)). In contrast, Flux's online estimator combined with the fine-grained performance model achieves the best training throughput for both eSEN and AllScAIP. We further examine edge-count-based scheduling across different architectures by comparing Flux with edge-level binning on eSEN and MatRIS in Appendix F.2.

6.3 Overall performance

Figure 4 reports the end-to-end speedup across different MLIPs. Under the same global batch size as the static baseline, Flux improves throughput by 1.16–1.98× on AllScAIP, eSEN, and MatRIS. These gains primarily reflect reduced synchronization stalls with the global batch size held fixed. Under the Max Batch setting, dynamic scheduling brings larger gains. The reason is that Flux can use a larger memory-feasible global batch, which reduces the number of iterations needed to process the same amount of data. Flux achieves 2.62×, 4.76×, and 2.23× speedup on eSEN, MatRIS, and AllScAIP, respectively. In comparison, FairChem reaches 2.58×, 4.15×, and 2.10×. Additional results for six SOTA MLIP architectures are reported in Appendix F.1.

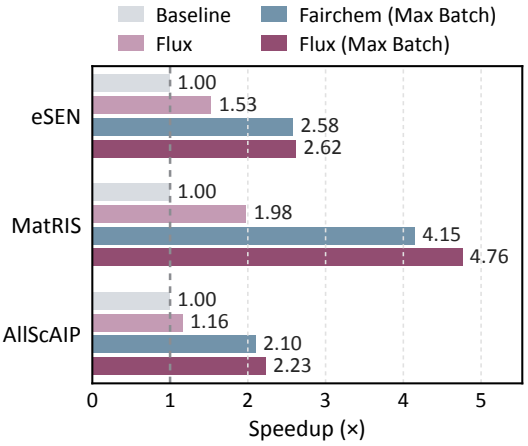

Figure 4: Overall performance of Flux.

FairChem is a strong dynamic scheduling baseline. On eSEN, atom-count balancing already captures much of the workload variation, leaving limited room for further gains. Flux achieves larger improvements on AllScAIP by accounting for per-graph collation overhead and on MatRIS by capturing higher-order interaction costs that atom count alone cannot fully describe. Beyond these performance gains, Flux operates online without requiring precomputed workload metadata, making it easier to adapt to changes in model architecture, cutoff radius, or data composition.

6.4 Convergence validation

To evaluate the effect of Flux on model convergence, we train MatRIS on the MatPES-r2SCAN dataset [28]. Flux and the standard distributed sampler use the same model configuration, optimizer,

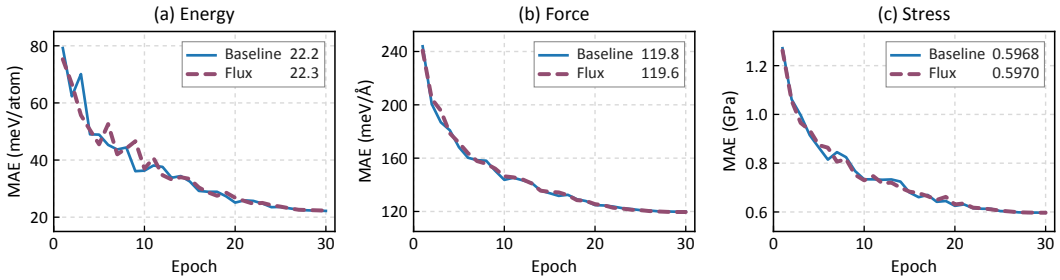

Figure 5: Validation MAE for energy, forces, and stress during MatRIS training with Flux and the standard distributed sampler (Baseline).

learning-rate schedule, global batch size, and random seed, with additional training details provided in Appendix E.1. As shown in Figure 5, the two methods exhibit comparable validation curves, indicating that Flux does not degrade model accuracy.

Flux changes how samples are assigned to DP ranks, which can affect local tensor shapes, collation order, and the order of graph operations and floating-point reductions. Since floating-point arithmetic is not strictly associative, such reordering can introduce small numerical differences [29, 30]. These differences may lead to slight variations in the intermediate validation curves, while both methods converge to comparable final validation errors.

7 Conclusion

In this work, we propose Flux, an online, fine-grained scheduling system for distributed MLIP training. It addresses a practical bottleneck: samples with similar atom counts can still incur substantially different computation and collation costs due to variations in graph size, edge count, and higher-order geometric features. Flux does not rely on offline graph metadata; instead, it estimates workload features during data loading, calibrates lightweight runtime and memory models, and forms more balanced local mini-batches across data-parallel ranks via windowed scheduling. Experiments on multiple datasets and MLIP architectures show that Flux effectively mitigates load imbalance.

Flux is designed for synchronous data-parallel MLIP training and targets load imbalance caused by variations in per-sample workload. Extending Flux to pipeline parallelism, expert parallelism, or highly asynchronous execution may require additional coordination. Flux also complements model- and kernel-level optimizations that reduce per-step computational costs.

Acknowledgments

This work is supported by the following funding: National Science Foundation of China (92270206, 62372435, T2125013, 62502501), Beijing Natural Science Foundation (4254087), National Key Research and Development Program of China (2025YFB4507000), and the Innovation Funding of ICT, CAS. Part of the training is performed on the robotic AI-Scientist platform of Chinese Academy of Sciences.

The authors declare no competing interests.

References

[1] Luis Barroso-Luque, Muhammed Shuaibi, Xiang Fu, Brandon M. Wood, Misko Dzamba, Meng Gao, Ammar Rizvi, C. Lawrence Zitnick, and Zachary W. Ulissi. Open materials 2024 (omat24) inorganic materials dataset and models, 2024. URL https://arxiv.org/abs/2410.12771.

[2] Filippo Bigi, Paolo Pegolo, Arslan Mazitov, Jonathan Schmidt, and Michele Ceriotti. Pushing the limits of unconstrained machine-learned interatomic potentials, 2026. URL https://arxiv.org/abs/2601.16195.

[3] Daniel S. Levine, Muhammed Shuaibi, Evan Walter Clark Spotte-Smith, Michael G. Taylor, Muhammad R. Hasyim, Kyle Michel, Ilyes Batatia, Gábor Csányi, Misko Dzamba, Peter Eastman, Nathan C. Frey, Xiang Fu, Vahe Gharakhanyan, Aditi S. Krishnapriyan, Joshua A. Rackers, Sanjeev Raja, Ammar Rizvi, Andrew S. Rosen, Zachary Ulissi, Santiago Vargas, C. Lawrence Zitnick, Samuel M. Blau, and Brandon M. Wood. The open molecules 2025 (omol25) dataset, evaluations, and models, 2026. URL https://arxiv.org/abs/2505.08762.

[4] Brandon M Wood, Misko Dzamba, Xiang Fu, Meng Gao, Muhammed Shuaibi, Luis Barroso-Luque, Kareem Abdelmaqsoud, Vahe Gharakhanyan, John R. Kitchin, Daniel S. Levine, Kyle Michel, Anuroop Sriram, Taco Cohen, Abhishek Das, Sushree Jagriti Sahoo, Ammar Rizvi, Zachary Ward Ulissi, and C. Lawrence Zitnick. UMA: A family of universal models for atoms. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=SvopaNxYWt.

[5] Yuanchang Zhou, Siyu Hu, Xiangyu Zhang, Hongyu Wang, Guangming Tan, and Weile Jia. MatRIS: Toward reliable and efficient pretrained machine learning interatomic potentials. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=5xBT5Ziute.

[6] Yuanchang Zhou, Hongyu Wang, Yiming Du, Yan Wang, Mingzhen Li, Siyu Hu, Xiangyu Zhang, Weijian Liu, Chen Wang, Zhuoqiang Guo, Long Wang, Jingde Bu, Yutong Lu, Guangming Tan, and Weile Jia. Breaking the training barrier of billion-parameter universal machine learning interatomic potentials, 2026. URL https://arxiv.org/abs/2604.15821.

[7] Jesun Sahariar Firoz, Franco Pellegrini, Mario Geiger, Darren Hsu, Jenna A. Bilbrey, Han-Yi Chou, Maximilian Stadler, Markus Höhnerbach, Tingyu Wang, Dejun Lin, Emine Kucukbenli, Henry W. Sprueill, Ilyes Batatia, Sotiris S. Xantheas, Mal-Soon Lee, Christopher Mundy, Gabor Csanyi, Justin S. Smith, Ponnuswamy Sadayappan, and Sutanay Choudhury. Optimizing data distribution and kernel performance for efficient training of chemistry foundation models: A case study with mace. HPDC '25, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400718694. doi: 10.1145/3731545.3731594. URL https://doi.org/10.1145/3731545.3731594.

[8] Mark Neumann, James Gin, Benjamin Rhodes, Steven Bennett, Zhiyi Li, Hitarth Choubisa, Arthur Hussey, and Jonathan Godwin. Orb: A fast, scalable neural network potential, 2024. URL https://arxiv.org/abs/2410.22570.

[9] Yuanchang Zhou, Siyu Hu, Chen Wang, Lin-Wang Wang, Guangming Tan, and Weile Jia. Fastchgnet: Training one universal interatomic potential to 1.5 hours with 32 gpus. In *2025 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 236–246, 2025. doi: 10.1109/IPDPS64566.2025.00029.

[10] Yutack Park, Jaesun Kim, Seungwoo Hwang, and Seungwu Han. Scalable Parallel Algorithm for Graph Neural Network Interatomic Potentials in Molecular Dynamics Simulations. *Journal of Chemical Theory and Computation*, 20(11):4857–4868, June 2024. ISSN 1549-9618. doi: 10.1021/acs.jctc.4c00190. URL https://doi.org/10.1021/acs.jctc.4c00190.

[11] Duo Zhang, Anyang Peng, Chun Cai, Wentao Li, Yuanchang Zhou, Jinzhe Zeng, Mingyu Guo, Chengqian Zhang, Bowen Li, Hong Jiang, Tong Zhu, Weile Jia, Linfeng Zhang, and Han Wang. A graph neural network for the era of large atomistic models, 2026. URL https://arxiv.org/abs/2506.01686.

[12] Lowik Chanussot, Abhishek Das, Siddharth Goyal, Thibaut Lavril, Muhammed Shuaibi, Morgane Riviere, Kevin Tran, Javier Heras-Domingo, Caleb Ho, Weihua Hu, et al. Open catalyst 2020 (oc20) dataset and community challenges. *Acs Catalysis*, 11(10):6059–6072, 2021.

[13] Bowen Deng, Peichen Zhong, KyuJung Jun, Janosh Riebesell, Kevin Han, Christopher J. Bartel, and Gerbrand Ceder. Chgnet as a pretrained universal neural network potential for charge-informed atomistic modelling. *Nature Machine Intelligence*, 2023. doi: 10.1038/s42256-023-00716-3.

[14] Vahe Gharakhanyan, Luis Barroso-Luque, Yi Yang, Muhammed Shuaibi, Kyle Michel, Daniel S. Levine, Misko Dzamba, Xiang Fu, Meng Gao, Xingyu Liu, Haoran Ni, Keian Noori, Brandon M. Wood, Matt Uyttendaele, Arman Boromand, C. Lawrence Zitnick, Noa Marom, Zachary W. Ulissi, and Anuroop Sriram. Open molecular crystals 2025 (omc25) dataset and models, 2025.

[15] Daniel S. Levine, Nicholas Liesen, Lauren Chua, James Diffenderfer, Helgi Ingolfsson, Matthew P. Kroonblawd, Nitesh Kumar, Amitesh Maiti, Supun S. Mohottalalage, Muhammed Shuaibi, Brian Van Essen, Brandon M. Wood, C. Lawrence Zitnick, Samuel M. Blau, and Evan R. Antoniuk. The open polymers 2026 (opoly26) dataset and evaluations, 2026.

[16] Ilyes Batatia, Chen Lin, Joseph Hart, Elliott Kasoar, Alin M. Elena, Sam Walton Norwood, Thomas Wolf, and Gábor Csányi. Cross learning between electronic structure theories for unifying molecular, surface, and inorganic crystal foundation force fields, 2025.

[17] Jaesun Kim, Jinmu You, Yutack Park, Yunsung Lim, Yujin Kang, Jisu Kim, Haekwan Jeon, Suyeon Ju, Deokgi Hong, Seung Yul Lee, Saerom Choi, Yongdeok Kim, Jae W. Lee, and Seungwu Han. Optimizing cross-domain transfer for universal machine learning interatomic potentials. *Nature Communications*, March 2026. ISSN 2041-1723. doi: 10.1038/s41467-026-70195-8.

[18] Vivek Bharadwaj, Austin Glover, Aydin Buluc, and James Demmel. An efficient sparse kernel generator for o(3)-equivariant deep networks, 2025. URL https://arxiv.org/abs/2501.13986.

[19] Yuanchang Zhou, Mingzhen Li, Chen Wang, Xiangyu Zhang, Guangming Tan, and Weile Jia. Lunatp: Enabling efficient and scalable tensor product for equivariant mlips, 2026. URL https://arxiv.org/abs/2610.18393.

[20] Chen Wang, Yuanchang Zhou, Guangming Tan, Mingzhen Li, and Weile Jia. Trimso(2): Fast and memory-efficient training of equivariant interatomic potentials, 2026. URL https://arxiv.org/abs/2610.20029.

[21] Eric Qu and Aditi S. Krishnapriyan. The importance of being scalable: Improving the speed and accuracy of neural network interatomic potentials across chemical domains. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=Y4mBaZu4vy.

[22] Eric Qu, Brandon M. Wood, Aditi S. Krishnapriyan, and Zachary W. Ulissi. A recipe for scalable attention-based mlips: unlocking long-range accuracy with all-to-all node attention, 2026. URL https://arxiv.org/abs/2603.06567.

[23] Anuroop Sriram, Logan M. Brabson, Xiaohan Yu, Sihoon Choi, Kareem Abdelmaqsoud, Elias Moubarak, Pim de Haan, Sindy Löwe, Johann Brehmer, John R. Kitchin, Max Welling, C. Lawrence Zitnick, Zachary Ulissi, Andrew J. Medford, and David S. Sholl. The open dac 2025 dataset for sorbent discovery in direct air capture, 2025. URL https://arxiv.org/abs/2508.03162.

[24] Xiang Fu, Brandon M Wood, Luis Barroso-Luque, Daniel S. Levine, Meng Gao, Misko Dzamba, and C. Lawrence Zitnick. Learning smooth and expressive interatomic potentials for physical property prediction. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=R0PBjxIbgm.

[25] Ryan Liu, Eric Qu, Tobias Kreiman, Samuel M. Blau, and Aditi S. Krishnapriyan. From evaluation to design: Using potential energy surface smoothness metrics to guide machine learning interatomic potential architectures, 2026. URL https://arxiv.org/abs/2602.04861.

[26] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, September 1975. ISSN 0001-0782. doi: 10.1145/361002.361007. URL https://doi.org/10.1145/361002.361007.

[27] Ilyes Batatia, David Peter Kovacs, Gregor N. C. Simm, Christoph Ortner, and Gabor Csanyi. MACE: Higher order equivariant message passing neural networks for fast and accurate force fields. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=YPpSngE-ZU.

[28] Aaron D. Kaplan, Runze Liu, Ji Qi, Tsz Wai Ko, Bowen Deng, Janosh Riebesell, Gerbrand Ceder, Kristin A. Persson, and Shyue Ping Ong. A foundational potential energy surface dataset for materials, 2025. URL https://arxiv.org/abs/2503.04070.

[29] Caroline Collange, David Defour, Stef Graillat, and Roman Iakymchuk. Numerical reproducibility for the parallel reduction on multi- and many-core architectures. *Parallel Comput.*, 49(C):83–97, November 2015. ISSN 0167-8191. doi: 10.1016/j.parco.2015.09.001. URL https://doi.org/10.1016/j.parco.2015.09.001.

[30] Sanjif Shanmugavelu, Mathieu Taillefumier, Christopher Culver, Oscar Hernandez, Mark Coletti, and Ada Sedova. Impacts of floating-point non-associativity on reproducibility for hpc and deep learning applications, 2024. URL https://arxiv.org/abs/2408.05148.

[31] Chi Chen and Shyue Ping Ong. A universal graph deep learning interatomic potential for the periodic table. *Nature Computational Science*, 2(11):718–728, November 2022. ISSN 2662-8457.

[32] Yuzhi Liu, Duo Zhang, Anyang Peng, Weinan E, Linfeng Zhang, and Han Wang. Scaling machine learning interatomic potentials with mixtures of experts, 2026. URL https://arxiv.org/abs/2603.07977.

[33] Yi-Lun Liao, Alexander J. Hoffman, Sabrina C. Shen, Alexandre Duval, Sam Walton Norwood, and Tess Smidt. Equiformerv3: Scaling efficient, expressive, and general se(3)-equivariant graph attention transformers, 2026. URL https://arxiv.org/abs/2604.09130.

[34] Benjamin Rhodes, Sander Vandenhaute, Vaidotas Šimkus, James Gin, Jonathan Godwin, Tim Duignan, and Mark Neumann. Orb-v3: atomistic simulation at scale, 2025. URL https://arxiv.org/abs/2504.06231.

[35] Jinzhe Zeng, Duo Zhang, Anyang Peng, Xiangyu Zhang, Sensen He, Yan Wang, Xinzijian Liu, Hangrui Bi, Yifan Li, Chun Cai, Chengqian Zhang, Yiming Du, Jia-Xin Zhu, Pinghui Mo, Zhengtao Huang, Qiyu Zeng, Shaochen Shi, Xuejian Qin, Zhaoxi Yu, Chenxing Luo, Ye Ding, Yun-Pei Liu, Ruosong Shi, Zhenyu Wang, Sigbjørn Løland Bore, Junhan Chang, Zhe Deng, Zhaohan Ding, Siyuan Han, Wanrun Jiang, Guolin Ke, Zhaoqing Liu, Denghui Lu, Koki Muraoka, Hananeh Oliaei, Anurag Kumar Singh, Haohui Que, Weihong Xu, Zhangmancang Xu, Yong-Bin Zhuang, Jiayu Dai, Timothy J. Giese, Weile Jia, Ben Xu, Darrin M. York, Linfeng Zhang, and Han Wang. Deepmd-kit v3: A multiple-backend framework for machine learning potentials. *Journal of Chemical Theory and Computation*, 21(9):4375–4385, 2025. doi: 10.1021/acs.jctc.5c00340. URL https://doi.org/10.1021/acs.jctc.5c00340. PMID: 40315155.

[36] Tsz Wai Ko, Bowen Deng, Marcel Nassar, Luis Barroso-Luque, Runze Liu, Ji Qi, Atul C. Thakur, Adesh Rohan Mishra, Elliott Liu, Gerbrand Ceder, Santiago Miret, and Shyue Ping Ong. Materials Graph Library (MatGL), an open-source graph deep learning library for materials science and chemistry. *npj Computational Materials*, 11(1):253, August 2025. ISSN 2057-3960. doi: 10.1038/s41524-025-01742-y. URL https://doi.org/10.1038/s41524-025-01742-y.

[37] Filippo Bigi, Joseph W. Abbott, Philip Loche, Arslan Mazitov, Davide Tisi, Marcel F. Langer, Alexander Goscinski, Paolo Pegolo, Sanggyu Chong, Rohit Goswami, Pol Febrer, Sofiia Chorna, Matthias Kellner, Michele Ceriotti, and Guillaume Fraux. Metatensor and metatomic: Foundational libraries for interoperable atomistic machine learning. *J. Chem. Phys.*, 164(6): 064113, February 2026. ISSN 0021-9606. doi: 10.1063/5.0304911.

[38] Fairchem. URL `https://github.com/facebookresearch/fairchem/blob/28292f090743f36c1151002964140b04d4b2c6af/src/fairchem/core/common/data_parallel.py#L122`.

A AllScAIP graph collation profiling

We analyze AllScAIP's graph collation at the operator level. As shown in Figure 6, this stage involves a large number of small GPU kernels. Profiling shows that CPU-side dispatch/launch overhead exceeds the GPU execution time of individual kernels. Typically, a single kernel takes 1–10 μs to execute, while processing each graph incurs approximately 2.5 ms of CPU overhead. During graph collation, each graph within a batch executes a similar sequence of small operators. Although GPU execution time varies with graph size, the per-graph overhead remains. As a result, the execution time of this stage increases approximately linearly with the number of graphs in the local batch.

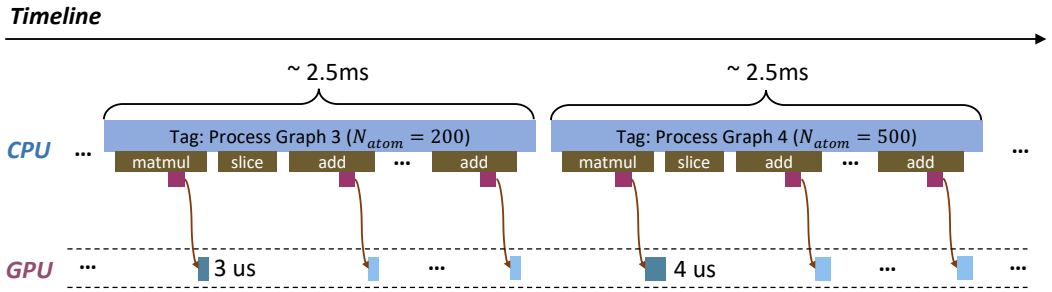

Figure 6: Operator-level profiling (partial) of the AllScAIP graph collation stage.

B Workload estimation

B.1 Domain-specific PBC corrections

The density estimate in Sec. 4.2 uses the full cell volume. In OC20 slabs, vacuum lowers this density and leads to underestimated edge counts. In ODAC25 MOFs, pores and cell anisotropy also affect the relation between average density and local coordination. Flux corrects the estimate using geometric features of each structure.

For a periodic structure with N atoms and cell volume V_{cell}, Flux first computes the uniform-density estimate

$$\hat{e}_{\text{uni}} = N\left(\frac{N}{V_{\text{cell}}} \cdot \frac{4}{3}\pi r_{\text{cut}}^3\right), \tag{12}$$

and applies a log-ratio correction. The feature vector $\psi(r_{\text{cut}})$ contains atom count, density, estimated coordination, cell shape, area-normalized density, and occupied thickness. We measure occupied thickness as the minimum periodic span of atoms along the longest cell-height direction, padded by r_{cut}.

We obtain reference edge counts $e > 0$ from a neighbor-list counter on a calibration subset and fit $\psi(r_{\text{cut}})^\top \omega$ to $\log(e/\hat{e}_{\text{uni}})$ by least squares. With fitted weights $\omega^\star$, the corrected estimate is

$$\hat{e} = \text{clip}\left(\hat{e}_{\text{uni}} \exp\left(\psi(r_{\text{cut}})^\top \omega^\star\right),\ 0,\ N(N-1)\right). \tag{13}$$

OC20 and ODAC25 use the same correction form with separately fitted weights.

Angle counts use the sampled-center estimator described below.

B.2 Higher-order feature estimation

Many MLIPs [31, 13, 11, 5] use higher-order geometric features, such as bond angles. Flux estimates the number of these features from local neighbor counts. For a structure with N atoms, let $r_{\text{cut}}^{(k)}$ be the cutoff for interaction order $k = 3, \ldots, K$ and $d_a^{(k)}$ the number of neighbors of center atom a within this cutoff. The number of k-body interactions is given by

$$c^{(k)} = \sum_{a=1}^{N} \binom{d_a^{(k)}}{k-1}. \tag{14}$$

For three-body interactions, this becomes

$$c^{(3)} = \sum_{a=1}^{N} \binom{d_a^{(3)}}{2} = \sum_{a=1}^{N} \frac{d_a^{(3)}(d_a^{(3)} - 1)}{2}. \tag{15}$$

For non-PBC structures in OMol25 and OPoly26, Flux computes neighbor degrees from the pairwise distance matrix when the structure is below the exact-counting threshold:

$$d_a^{(k)} = \sum_{b=1}^{N} \mathbf{1}\left[d_{ab}^2 \leq \left(r_{\text{cut}}^{(k)}\right)^2\right] - 1. \tag{16}$$

Here, d_{ab} is the interatomic distance, and subtracting one excludes the self-pair. These degrees give exact higher-order counts through Eq. 14, without constructing neighbor lists. For larger structures, Flux avoids the quadratic cost of the distance matrix by estimating the mean degree $\bar{d}^{(k)}$ from an effective density and computing

$$\hat{c}^{(k)} \approx N\binom{\bar{d}^{(k)}}{k-1}. \tag{17}$$

Most OMol25 and OPoly26 samples use the distance-matrix method.

For bulk-like PBC structures in OMat24 and OMC25, Flux uses the density-based mean degree $\bar{d}$ at cutoff r_{cut} from Eq. 7. Under the uniform-density approximation, neighbor count scales with the volume of the cutoff sphere, so

$$\bar{d}^{(k)} = \bar{d}\left(\frac{r_{\text{cut}}^{(k)}}{r_{\text{cut}}}\right)^3. \tag{18}$$

The estimated count is

$$\hat{c}^{(k)} = N\binom{\bar{d}^{(k)}}{k-1}, \tag{19}$$

where, for estimated degrees $d \geq 0$ and integers $m \geq 1$, we evaluate

$$\binom{d}{m} = \frac{1}{m!}\prod_{q=0}^{m-1} \max(d - q, 0). \tag{20}$$

For OC20 slabs and ODAC25 MOFs, local coordination varies within a structure. Flux therefore estimates neighbor combinations at sampled centers rather than substituting a single mean degree. It samples a set of center atoms $\mathcal{C} \subseteq \{1, \ldots, N\}$, counts their neighbors within $r_{\text{cut}}^{(k)}$ under PBC, and scales the mean number of neighbor combinations by N:

$$\hat{c}^{(k)} = \frac{N}{|\mathcal{C}|}\sum_{a \in \mathcal{C}} \binom{d_a^{(k)}}{k-1}. \tag{21}$$

We use up to 32 center atoms for angle estimation. The performance model sums the estimates for structures i in each local batch B: $\hat{E}(B) = \sum_{i \in B} \hat{e}_i$ and $\hat{C}^{(k)}(B) = \sum_{i \in B} \hat{c}_i^{(k)}$.

B.3 Additional estimator accuracy

Figure 7 compares estimated and reference edge and angle counts on OC20 slabs, OMC25 molecular crystals, and ODAC25 MOFs. All six comparisons have $R^2 > 0.95$. Edge estimates are most accurate on OMC25 ($R^2 = 0.997$), while OC20 shows greater variation at high edge counts.

Angle estimates reach $R^2 = 0.993$ on OC20 and 0.984 on ODAC25, compared with 0.954 on OMC25. The sampled-center estimator uses local degrees for OC20 and ODAC25, whereas OMC25 uses a density-based mean degree. Because angle count grows quadratically with degree (Eq. 15), averaging degrees loses information about coordination variation.

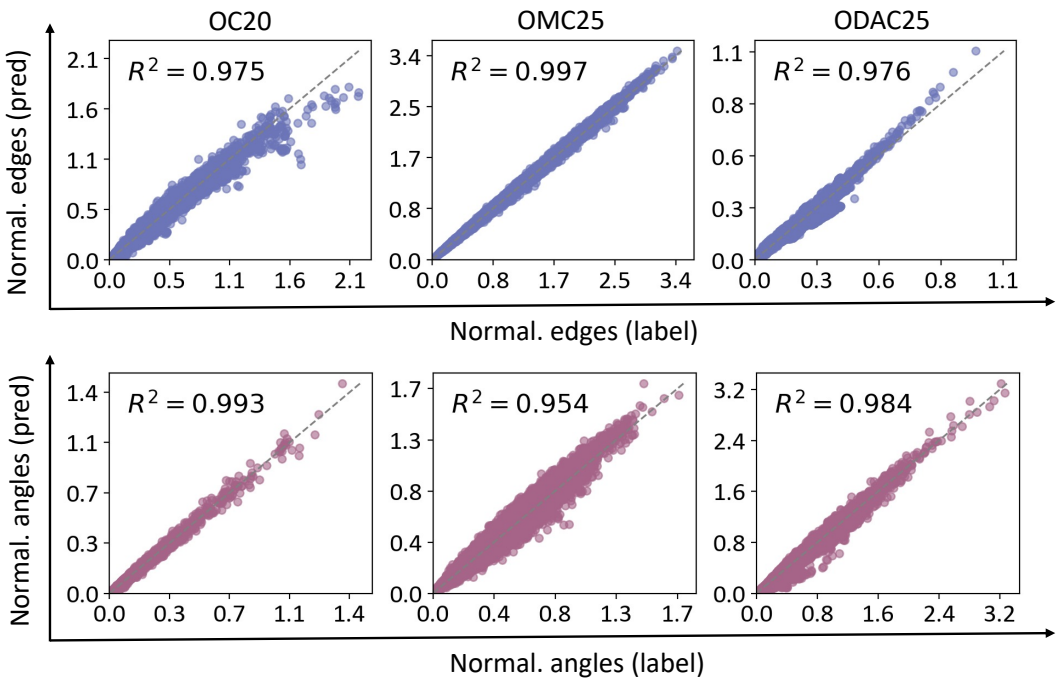

Figure 7: Edge-count (top) and angle-count (bottom) estimation accuracy on OC20, OMC25, and ODAC25. Dashed lines indicate exact agreement.

B.4 Extreme data heterogeneity

To test estimation accuracy across a wider range of structure sizes and geometries, we mix 5,000 OMol25 molecules, 5,000 ODAC25 MOFs, and 100 ODAC25-derived supercells with approximately 10,000 atoms each. The supercells extend the size range beyond that of the sampled MOFs. Flux processes each structure independently, using the non-PBC counter for molecules and the corrected PBC estimator for MOFs. Table 2 reports accuracy on each subset. The mean of the three subset-level R^2 values is 0.991, and the pooled R^2 is 0.9996.

Table 2: Edge-count accuracy under extreme heterogeneity, with a 6.0 Å cutoff. The supercells extend the size range beyond that of the sampled MOFs.

Subset	Structures	Atoms	R^2
OMol25 molecules	5,000	2–348	1.0000
ODAC25 MOFs	5,000	16–821	0.9753
ODAC25 supercells	100	9,160–10,728	0.9977

In the corresponding eSEN-6M timing study on A100 80 GB GPUs, predicted and measured training times have $R^2 = 0.997$, with a P99 absolute relative error of 4.33%. CPU analysis sustains 7486.2 structures/s, compared with a training throughput of 201.5 structures/s, leaving substantial processing capacity for online scheduling in this setting.

C Runtime and memory models

C.1 Workload patterns across models

We compare workload patterns across MLIPs by fitting training-time and memory models using graph count, atom count, edge count, or the full feature set (Figure 8). We profile the models on the mixed dataset.

Edge count alone provides accurate predictions for MACE and ORBv3, while atom count alone nearly matches the full predictor for PET. For MatRIS, which includes higher-order features, combining structural counts yields more accurate predictions.

For UMA, edge count predicts memory usage well but is less accurate for training time ($R^2 = 0.716$). UMA's mixture of linear experts (MoLE) combines expert parameters for each structure [4], introducing additional graph-dependent computation. Including graph and atom counts alongside edge count improves the training-time R^2 to 0.987.

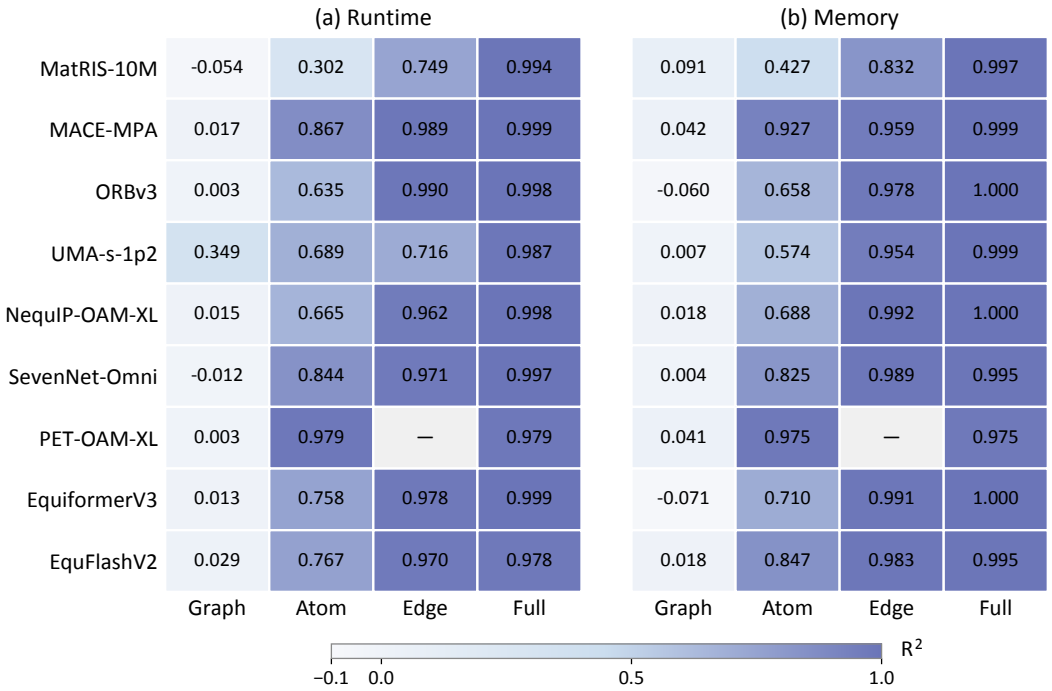

Figure 8: Workload-model accuracy on the mixed dataset. Panels show training-time and memory R^2 for graph-only, atom-only, edge-only, and full predictors. A dash denotes a predictor not evaluated for that model.

C.2 Graph-size and memory-model bounds

Flux performs scheduling before full graph construction, when atom counts are directly available but edge and higher-order interaction counts must be estimated. To assess how these estimates affect memory prediction, we derive upper and lower bounds on these counts based on cell geometry and cutoff radius, and use them to bound memory usage through the memory model in Eq. 11.

Consider a periodic structure with N atoms and a nonsingular cell. For $\alpha = 1, 2, 3$, let L_α be the perpendicular distance between the cell faces spanned by the other two lattice vectors, and let $L_{\min} = \min_\alpha L_\alpha$. Define

$$P = \left(\left\lfloor \frac{2r_{\text{cut}}}{L_{\min}} \right\rfloor + 1 \right)^3, \qquad U = N^2 P - N. \tag{22}$$

Here, P bounds the number of periodic images per ordered atom pair. Counting directed edges to distinct periodic images and excluding zero-shift self-edges, the exact edge count e and any estimate $\hat{e} \geq 0$ satisfy

$$0 \leq e \leq U, \qquad |e - \hat{e}| \leq \max\{\hat{e}, |U - \hat{e}|\}. \tag{23}$$

Proof. Project the displacement of atom pair (a, b) onto the unit normal of each face pair, oriented so that the αth lattice vector has projection L_α. Denote this displacement by $\delta_{ab,\alpha}$. The other two lattice vectors have zero projection onto this normal, so a periodic shift producing an edge must

satisfy $|\delta_{ab,\alpha} + m_\alpha L_\alpha| \le r_{\mathrm{cut}}$, where $m_\alpha \in \mathbb{Z}$. The allowed integers lie in an interval of length $2r_{\mathrm{cut}}/L_\alpha$, containing at most $\lfloor 2r_{\mathrm{cut}}/L_\alpha \rfloor + 1$ integers. Since $L_\alpha \ge L_{\min}$, the product of these three bounds is at most P. Summing over N^2 ordered atom pairs and excluding the N zero-shift self-edges gives $e \le U$. The error bound follows by maximizing $|e - \hat{e}|$ at the endpoints of $[0, U]$. When $r_{\mathrm{cut}} < L_{\min}/2$, we have $P = 1$ and $U = N(N-1)$.

Memory-model bound. Each atom has at most $D = NP - 1$ neighbors. For $r_{\mathrm{cut}}^{(k)} \le r_{\mathrm{cut}}$, choosing $k-1$ neighbors around each center bounds the unordered k-body count in Eq. 14:

$$c^{(k)} \le U^{(k)} := N \binom{D}{k-1}. \tag{24}$$

For a batch B, let U_i and $U_i^{(k)}$ denote these bounds for structure i. The exact batch counts are $N(B) = \sum_{i \in B} n_i$, $E(B) = \sum_{i \in B} e_i$, and $C^{(k)}(B) = \sum_{i \in B} c_i^{(k)}$, following Sec. 4.1. Evaluating the memory model with these counts gives

$$\widetilde{M}(B) = M_{\mathrm{static}} + \beta_N N(B) + \beta_E E(B) + \sum_{k=3}^{K} \beta_k C^{(k)}(B). \tag{25}$$

Define $\beta^+ = \max\{\beta, 0\}$. Since the counts are nonnegative, $\beta_E e_i \le \beta_E^+ U_i$ and $\beta_k c_i^{(k)} \le \beta_k^+ U_i^{(k)}$. Substitution gives an upper bound on the fitted memory model:

$$\widetilde{M}(B) \le U_{\mathrm{fit}}(B) := M_{\mathrm{static}} + \beta_N N(B) + \beta_E^+ \sum_{i \in B} U_i + \sum_{k=3}^{K} \beta_k^+ \sum_{i \in B} U_i^{(k)}. \tag{26}$$

Memory check after graph construction. After the assignment in Algorithm 1, the data-loading pipeline constructs the graphs and reevaluates Eq. 11 with the actual counts before transferring the batch to the GPU. If the estimate exceeds $M_{\max}$, samples are moved to the carry-over buffer until $\widetilde{M}(B_r) \le M_{\max}$.

C.3 Memory usage on heterogeneous batches

To assess batch memory stability on heterogeneous data, we measure GPU memory usage under Flux for eSEN-6M and MatRIS-4M on mixed workloads containing ODAC25 MOFs and OC20 slabs. Table 3 reports the mean and the P95, P99, and P99.9 percentiles, capturing both typical usage and the high-memory tail.

Table 3: GPU memory usage (GB) on heterogeneous workloads. P95, P99, and P99.9 denote empirical percentiles.

Model	Mean	P95	P99	P99.9
eSEN-6M	70.94	72.97	72.97	73.18
MatRIS-4M	65.18	67.77	68.39	68.39

For eSEN-6M and MatRIS-4M, P99.9 memory usage is 3.16% and 4.92% above the mean, respectively. Memory usage increases by 0.21 GB from P95 to P99.9 for eSEN-6M and by 0.62 GB for MatRIS-4M. The upper percentiles remain close to the mean for both models, indicating stable batch memory usage across the evaluated heterogeneous workloads.

D Scheduler overhead

Flux schedules the next batch on the CPU while the current GPU step executes. For the candidate window $\mathcal{W}_t$ and R data-parallel ranks, sorting candidates and scanning feasible ranks during assignment take

$$T_{\mathrm{sort+assign}} = O(|\mathcal{W}_t| \log |\mathcal{W}_t| + R|\mathcal{W}_t|)\,. \tag{27}$$

The scheduler stores candidate features, predicted costs, and buffer entries in $O(|\mathcal{W}_t| + R)$ space.

Larger windows give the scheduler more assignment choices at higher CPU cost. With $W_{\text{win}} = 2048$, the total scheduling time $T_{\text{sched}}(\mathcal{W}_t)$ remained below the GPU step time T_{step} in our experiments, allowing scheduling to overlap with training.

E Datasets and model configurations

We evaluated Flux on atomistic datasets used in prior studies [4, 32, 17, 16]. These datasets span periodic inorganic materials, molecules, molecular crystals, polymers, surfaces, and adsorption-related systems. Table 4 summarizes the datasets and sampling ratios used in our experiments.

The selected models reflect the dominant workload patterns in current MLIP training. eSEN relies heavily on edge-level message passing, MatRIS places greater emphasis on higher-order geometric features, and AllScAIP represents an attention-based architecture with non-negligible overhead from neighborhood construction and data collation. Their configurations are listed in Table 5.

Table 4: Datasets used in this work.

Dataset	System type	Structures	Avg. Size	Sample ratio	Notes	License
OC20 [12]	Catalyst slab systems	117.9M	77	1	Slab/adsorbate structures	CC BY 4.0
OMat24 [1]	Inorganic materials	100.8M	19	4	Bulk/PBC systems	CC BY 4.0
OMol25 [3]	Molecules	101.7M	52	4	Non-PBC systems	CC BY 4.0
OMC25 [14]	Molecular crystals	24.9M	130	2	Molecular-crystal structures with PBC	CC BY 4.0
ODAC25 [23]	MOFs	36.9M	178	1	Metal Organic Frameworks with PBC	CC BY 4.0
OPoly26 [15]	Polymers	5.9M	185	4	Non-PBC systems	CC BY 4.0

Table 5: Hyperparameters of the three models.

Hyperparameter	eSEN (6M)	MatRIS (4M)	AllScAIP-sm (34M)
Conservation	yes	yes	yes
Number of layers	4	4	6
Maximum degree $L_{\max}$	2	—	—
Maximum order $M_{\max}$	2	—	—
Atom feature dimension	128	128	512
Edge feature dimension	128	128	512
Three-body feature dimension	—	128	—
Hidden size	128	128	1024
Attention heads	—	—	8
Radial basis function	Gaussian	Bessel	Gaussian
Number of radial basis functions	10	7	10
Pairwise cutoff (Å)	6.0	6.0	6.0
Three-body cutoff (Å)	—	4.0	—
KNN soft	False	False	True
Maximum neighbors	300	inf	30
License	MIT	BSD-3-Clause	MIT

E.1 Convergence experiment

For the convergence comparison in Figure 5, we train MatRIS with 1.4M parameters for 30 epochs on MatPES-r2SCAN [28]. The model has four layers, with atom, edge, and three-body feature dimensions of 128, 32, and 32, respectively. The pairwise and three-body cutoffs are 6 and 4 Å.

Training uses four data-parallel ranks with a global batch size of 128, corresponding to 32 structures per rank before redistribution. We use AdamW with an initial learning rate of 5×10^{-4}, weight decay of 10^{-3}, and gradient clipping at 1.0. The learning rate follows a per-step cosine schedule with a minimum of 2.5×10^{-6}. The loss is computed at the graph level, with energy, force, and stress weights of 5, 5, and 0.1.

F Additional end-to-end evaluation

F.1 Performance across architectures

We extend the evaluation to UMA, MACE, EquiformerV3, SevenNet-Omni, ORBv3, and EquFlashV2 [4, 27, 33, 17, 34]. Figure 9 reports speedups relative to each model's baseline. Flux improves throughput by 2.27–4.87× over the baseline and by 1.04–1.17× over FairChem. The largest relative gain over FairChem occurs on EquFlashV2.

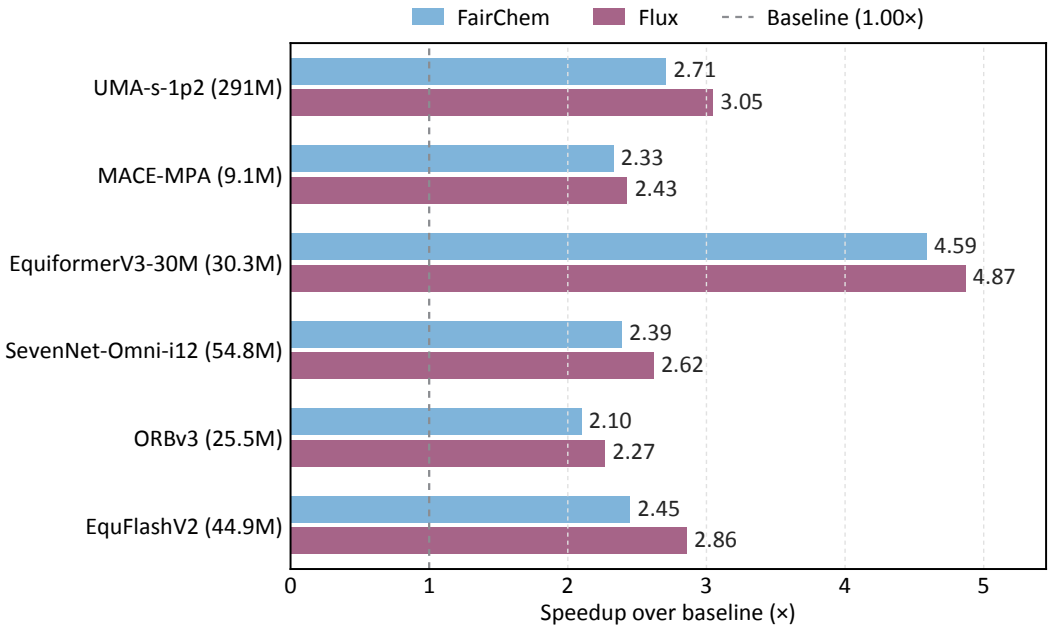

Figure 9: End-to-end speedup on 32 A100 80 GB GPUs. The dashed line marks the baseline (1.00×); labels give the measured speedups. Parameter counts identify the evaluated model configurations.

F.2 Comparison with edge-level binning

We compare Flux with edge-level binning, which balances batches using edge counts. Figure 10 shows the speedups across four eSEN and MatRIS configurations, normalized to their respective baselines.

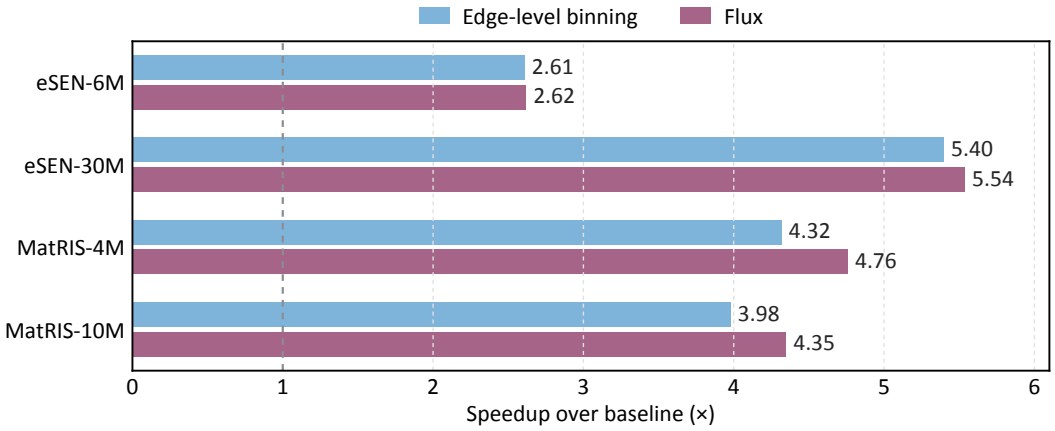

Figure 10: End-to-end speedup with edge-level binning and Flux.

Edge-level binning achieves comparable performance to Flux on eSEN, where edgewise convolutions account for much of the computation, making edge count an effective workload proxy. On MatRIS,

Flux achieves larger speedups by accounting for three-body interactions, whose computational costs are not fully captured by edge count.

G Programming interface

Flux can be integrated into existing PyTorch training scripts as a replacement for the standard batch sampler. In a typical distributed training setup, `DistributedSampler` partitions the shuffled dataset across ranks, and `BatchSampler` groups samples into local mini-batches:

```
from torch.utils.data import DataLoader
from torch.utils.data.distributed import DistributedSampler

sampler = DistributedSampler(
    dataset,
    num_replicas=world_size,
    rank=rank,
    shuffle=True,
    drop_last=True,
)

loader = DataLoader(
    dataset,
    batch_size=batch_size,
    sampler=sampler,
    collate_fn=collate_fn,
    num_workers=num_workers,
    pin_memory=True,
    drop_last=True,
)
```

With Flux, the standard batch sampler is replaced by `FluxDistributedBatchSampler`, which estimates sample workloads online and forms balanced local mini-batches across ranks:

```
from flux import FluxDistributedBatchSampler

sampler = FluxDistributedBatchSampler(
    dataset,
    num_replicas=world_size,
    rank=rank,
    shuffle=True,
    drop_last=True,
    global_batch_size=global_batch_size,
    estimator=flux_estimator,
    order=k, # workload order: 1 = atom counts, 2 = edge counts, >=3 = k-body term
    performance_model=flux_perf_model,
    lookahead_ratio=2.0,
    max_memory=max_memory_per_gpu,
)

loader = DataLoader(
    dataset,
    batch_sampler=sampler,
    collate_fn=collate_fn,
    num_workers=num_workers,
    pin_memory=True,
)
```

Flux operates within the data-loading pipeline, leaving the model, loss function, optimizer, and distributed training loop unchanged. Existing MLIP training frameworks [35–38] can therefore integrate Flux without modifying model computation or gradient synchronization.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims made in the abstract and introduction are detailed in the results.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss some limitations in Section 7.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: Appendix C.2 states the assumptions and derives the graph-size and fitted memory-model bounds.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide all details of our method and a schematic.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The datasets used in our experiments are publicly available. Code release is pending approval.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We list in Section 5.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Our experimental results are averaged by multiple runnings.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Yes, we provide computational details.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have reviewed this and the research conforms to this code.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: Our work is about system optimization.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: Our work is about system optimization.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Yes, we cite the relevant datasets and works we used.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Yes, we provide a new data scheduling strategy and include implementation details.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: We don't have this.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve this.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The paper do not involve usage of LLMs as any important, original, or non-standard components.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.