A Appendix (not peer-reviewed)

A.1 Experimental setup

In total, three types of machines are used in the evaluation of Snowcat: Machine A is a Google Cloud Platform (GCP) VM with 30 E2 vCPUs, mainly used for data collection; machine B is a GCP VM with 8 A100 40GB GPUs, used for model training and inference; machine C is a GCP VM with 128 vCPUs and 512GB memory, for downstream task evaluation.

To build the dataset, 150 machine A VMs ran for about 8 weeks, 2 weeks, and 2 weeks, respectively, on Linux kernel 5.12, 6.1, and 5.13. They ran SKI to execute schedules of random concurrent test inputs, during which the executed kernel instructions and memory accesses were profiled. In addition, 2 hours were spent on a machine A VM to build a kernel CFG using Angr.

8 VMs of machine B were used to train the model and tune hyperparameters. In total, these machines ran for 42 days to train 80 *PIC* models, under different hyperparameter settings. After hyperparameters were tuned, we ran VMs of machine B for 4 days to make inference on graphs in the evaluation dataset. We then evaluated the downstream task performance on VMs of machine B and C.

A.2 *PIC* model parameter tuning

Different hyperparameters are studied during the training of the *PIC* model: learning rate ∈ {1e-3, 1e-4, 5e-4, 1e-5}, batch size ∈ {16, 32, 64, 256}, GNN architecture ∈ {GAT [59], GATv2 [5], TransformerConv [51], GINE [24], PDN [47]}, GNN hidden dimension size ∈ {50, 200, 400, 800}, the number of GNN layers ∈ {2, 4, 6, 8, 10, 16, 24, 30, 36, 48}, the number of encoder layers ∈ {6, 8}, the encoder dimension size of the RoBERTa module ∈ {128, 256}. Adam [31] optimizer and StepLR [38] (step size = 4) are used for training these models.

We use the mean Average Precision (AP) [63] to compare the performance of different trained *PIC* models and consider the hyperparameters of the model that has the highest validation AP as best. Specifically, we use every trained model to predict concurrent test candidate graphs in the validation dataset. Next, we compute the AP over *URBs* on each graph and then average APs on all graphs. Table 8 reports the average AP each model achieves. The best model (*PIC*-5) has an average AP around 83%, which uses the following hyperparameters: batch size 16, learning rate 1e-4, 36 layers of TransformerConv (hidden dimension size 200) in the GNN module, 6 layers of encoders (embedding dimension size 128) in the sequence module.

A.3 *PIC* performance of predicting both *SCBs* and *URBs*

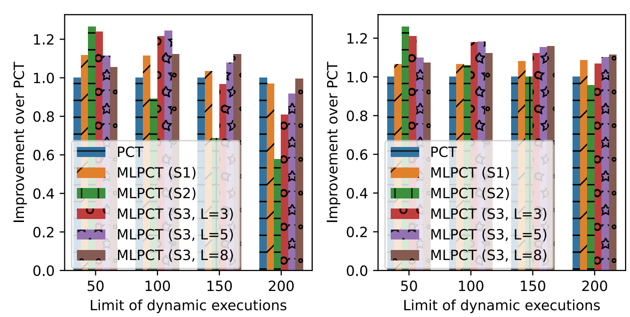
As shown in Table 6, **All pos** (§5.2) achieves relatively high performance when predicting all code blocks in the graph. The accuracy of *All pos* reflects the distribution of positive and negative blocks in a graph. That is, nearly 75% blocks in each graph will be executed under concurrent executions. Because of this distribution, the performance of *Fair coin* is less ideal and *Biased coin* is extremely poor. Compared with *All pos*, *PIC*-5 achieves even higher performance across all metrics (over 98%).

Predictor	F1	Precision	Recall	Accuracy	BA
PIC-5	99.42%	99.04%	99.85%	99.17%	98.71%
All pos	85.81%	75.18%	100.00%	75.18%	50.00%
Fair coin	60.03%	75.18%	50.00%	50.00%	50.00%
Biased coin	2.31%	75.18%	1.18%	25.42%	50.00%

Table 6. Results when using different predictors to predict **all blocks** in the graph. Average metrics across all graphs. BA stands for balanced accuracy.

A.4 Per-CTI Coverage Improvement

Figure 7 shows the coverage (§5.3) improvement when testing individual CTIs. It shows the detailed relative improvement over PCT by each approach (the *MLPCT* variants), for various budget caps on dynamic executions. As the budget goes up, the relative improvement shrinks (since there is less room for MLPCT to explore uncovered blocks or races). The comparison shows that *MLPCT* provides better testing results when the user wants to set a low budget on the dynamic execution (e.g, 50) but achieves higher coverage than PCT. In terms of the *MLPCT* strategies, S1 and S2 perform better.



(a) *Potential-data-race coverage* **(b)** *Schedule-dependent code coverage*

Figure 7. Coverage improvement of *MLPCT* over PCT, evaluated with different limits of dynamic executions.

A.5 Adapting *PIC* models to Newer Kernels

Several *PIC* models are retrained for Linux kernel 6.1, using different sizes of new training data. On each training dataset, two *PIC* models are separately trained by fine-tuning *PIC*-5 and training from scratch as a new fresh model. We measure the time taken for collecting different numbers of training examples on machine A, which has a similar specification as machines used for other kernel concurrency testing tools [19, 25], and the time for training/fine-tuning *PIC* models for 5 epochs on machine B, a VM with 8 A100 GPUs. The data collection and model training time are added up

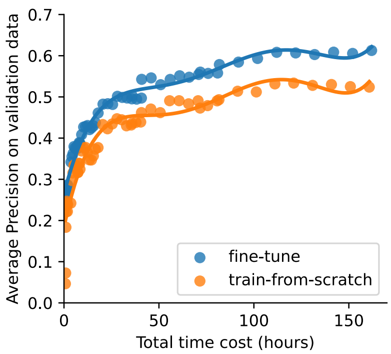


Figure 8. Validation performance over total time cost. The regression lines are drawn using polynomial regression with the order of 5 to fit data points.

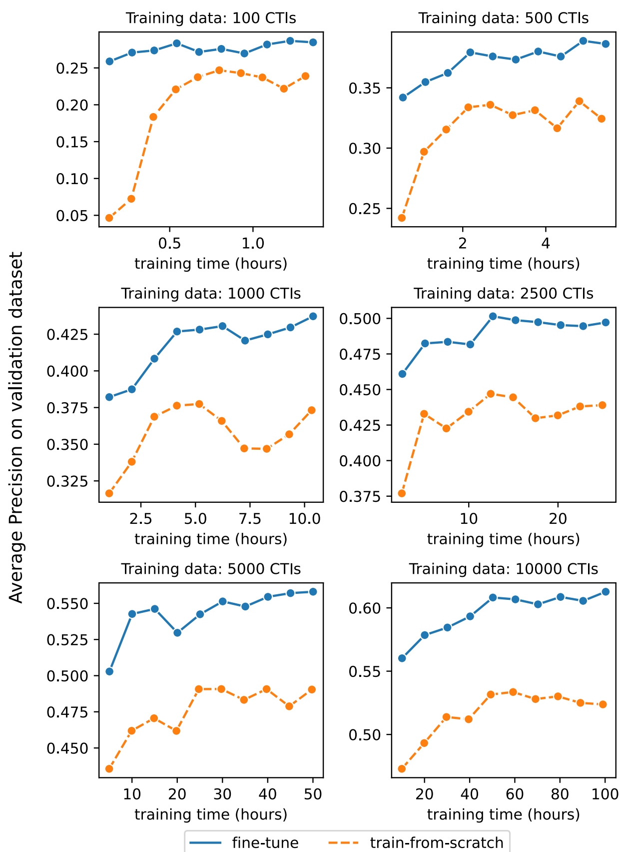


Figure 9. Validation performance over different sizes of training data and training time. Markers in each line represent the validation performance at different epochs.

as the total time cost. Figure 8 presents different validation performance (mean AP) given the total time cost and the re-training method. Besides the superiority of fine-tuning over re-training from scratch, it shows there might be other sweet spots (e.g., the first 25 hours) for fine-tuning to newer kernels that further improve the amortization speed of SNOWCAT.

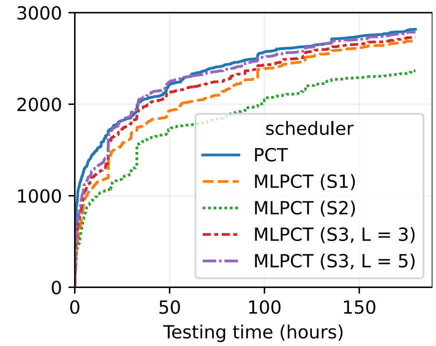


Figure 10. *Data-race-coverage* history comparison between PCT and *MLPCT*. Testing Linux kernel 6.1 using *PIC*-6.fs.sml.

Name		Source
Node:		
C_i	*Sequentially-covered block* (*SCB*)	Dynamic profiling
U_i	*Uncovered reachable block* (*URB*)	Static analysis
Edge:		
S_i	*SCB* control-flow	Dynamic profiling
P_i	Intra-thread data flow	Dynamic profiling
D_i	*URB* control-flow	Static analysis
A_i	Inter-thread possible dataflow	Dynamic profiling & Static analysis
I_i	Scheduling hint	Schedule generator (e.g., [17, 19, 25])

Table 7. Graph nodes and edges. All nodes and edges are determined through dynamic execution of sequential tests and static analysis.

Figure 9 provides a breakdown analysis on the size of training data and the training time. First, it shows again that the *fine-tuned* models are constantly better than *train-from-scratch* ones, regardless of the size of the training data and time. Second, it is observed that 5 epochs of training can generally reach the highest validation performance. This finding can help users of SNOWCAT set the time budget when training *PIC* models. Last, a larger amount of training data is shown to be beneficial to the model performance.

A.6 Cost/Benefit of a Candidate Evaluator

Note that in this problem formulation, although desirable, a perfect candidate evaluator is not required. In fact, false positives and false negatives need not be particularly low, as long as the end-to-end cost is favorable. Assuming that the original exhaustive approach will consider $N_{exhaustive}$ candidates (4 in the example of Figure 3), and that the imperfect evaluator will consider $N_{imperfect}$ candidates (7 in the above example), this new approach is favorable if $C_{Build} + N_{imperfect}C_{Evaluate} + kC_{Execute} \ll N_{exhaustive}C_{Execute}$, where C_* is the cost of various phases, e.g., building the ML training data and model, evaluating the model on a candidate, and executing a dynamic test, and k is the number of tests the evaluator considers potentially fruitful. Note that the false-positive rate determines how close k is to 1, and the false-negative rate determines how much greater $N_{imperfect}$ is than $N_{exhaustive}$.

Learning rate	Batch size	GCN arch	# GCN layer	GCN hidden dimension	Bert embedding dimmesion	Bert layers	Shortcut edge K	Validation AP
1.0E-05	16	Transformer	48	200	128	6	None	67.65%
1.0E-04	16	Transformer	48	200	128	6	None	74.48%
1.0E-04	16	Transformer	36	400	128	6	0	75.66%
1.0E-04	16	Transformer	36	400	128	6	2	81.77%
1.0E-04	16	Transformer	36	400	128	6	8	82.94%
1.0E-05	16	Transformer	36	400	128	6	8	73.92%
1.0E-04	64	Transformer	36	400	128	6	8	82.55%
1.0E-05	16	Transformer	36	800	128	6	8	82.18%
1.0E-04	16	Transformer	36	800	128	6	8	67.78%
1.0E-05	16	Transformer	36	200	128	6	None	67.25%
5.0E-05	16	Transformer	36	200	128	6	None	75.20%
1.0E-04	16	Transformer	36	200	128	6	None	74.33%
1.0E-04	16	Transformer	36	200	128	6	2	79.52%
1.0E-04	16	Transformer	36	200	128	6	4	80.15%
1.0E-04	16	Transformer	36	200	128	6	6	80.58%
1.0E-04	16	Transformer	36	200	128	6	8	81.79%
1.0E-04	16	Transformer	36	200	128	6	10	80.29%
1.0E-04	16	Transformer	36	200	128	6	16	80.51%
1.0E-04	16	Transformer	36	200	128	6	32	78.49%
1.0E-04	16	Transformer	36	200	128	6	4&8&16	82.06%
1.0E-04	16	Transformer	36	200	128	6	4&8&10	83.48%
1.0E-04	16	Transformer	36	200	128	6	0	76.41%
1.0E-04	16	Transformer	36	200	128	6	4	78.92%
1.0E-04	16	Transformer	36	200	128	6	8	79.03%
1.0E-04	16	Transformer	36	200	128	6	16	80.44%
1.0E-04	16	Transformer	36	200	128	6	None	75.23%
1.0E-05	16	GAT	36	200	128	6	None	38.28%
5.0E-05	16	GAT	36	200	128	6	None	65.15%
1.0E-04	16	GAT	36	200	128	6	None	1.68%
1.0E-05	16	GATv2	36	200	128	6	None	52.93%
5.0E-05	16	GATv2	36	200	128	6	None	1.36%
1.0E-04	16	GATv2	36	200	128	6	None	1.18%
1.0E-04	64	Transformer	36	200	128	6	None	74.31%
1.0E-04	256	Transformer	36	200	128	6	None	69.40%
1.0E-05	16	Transformer	30	200	128	6	None	67.68%
1.0E-04	16	Transformer	30	200	128	6	None	73.85%
1.0E-04	16	GINE	24	200	128	6	None	12.22%
1.0E-04	16	Transformer	24	200	128	6	None	72.31%
1.0E-05	16	Transformer	24	200	128	6	None	67.41%
1.0E-05	16	GAT	24	200	128	6	None	30.88%
1.0E-04	16	GAT	24	200	128	6	None	63.46%
1.0E-05	16	GATv2	24	200	128	6	None	48.70%
1.0E-04	16	GATv2	24	200	128	6	None	60.69%
1.0E-04	16	Transformer	16	200	128	6	None	68.84%
1.0E-05	16	Transformer	16	200	128	6	None	66.25%
1.0E-04	16	Transformer	10	200	128	6	None	65.44%
5.0E-05	16	Transformer	10	200	128	6	None	65.03%
1.0E-05	16	Transformer	10	200	128	6	None	61.16%
5.0E-05	16	GAT	10	200	128	6	None	59.15%
1.0E-04	16	GINE	8	200	128	6	None	28.72%
1.0E-04	16	GINE	8	200	128	6	None	31.56%
5.0E-05	16	Transformer	8	200	128	6	None	62.59%
1.0E-04	16	Transformer	8	200	128	6	None	62.53%
1.0E-05	16	Transformer	8	200	128	6	None	58.27%
1.0E-04	16	GATv2	8	200	128	6	None	61.12%
1.0E-05	16	GATv2	8	200	128	6	None	53.36%
5.0E-05	16	GAT	8	200	128	6	None	50.68%
1.0E-05	16	GAT	8	200	128	6	None	46.98%
1.0E-04	16	GAT	8	200	128	6	None	59.55%
1.0E-04	16	Transformer	6	200	128	6	None	59.61%
1.0E-04	16	GAT	6	200	128	6	None	56.84%
1.0E-04	16	PDN	4	50	128	6	None	22.82%
1.0E-04	16	Transformer	4	400	128	6	None	56.99%
1.0E-04	16	Transformer	4	200	128	6	None	56.69%
1.0E-04	16	GAT	4	200	128	6	None	54.34%
1.0E-04	16	GAT	3	400	128	6	None	50.38%
5.0E-04	64	GAT	2	2	128	6	None	32.09%
1.0E-04	16	GAT	2	400	128	6	None	24.09%
1.0E-04	16	GAT	2	200	256	8	None	34.88%
1.0E-04	16	GAT	2	200	128	6	None	32.34%
5.0E-04	16	GAT	2	200	128	6	None	31.23%
1.0E-05	16	GAT	2	200	128	6	None	19.84%
1.0E-03	16	GAT	2	200	128	6	None	1.75%
1.0E-03	32	GAT	2	200	128	6	None	31.28%
5.0E-04	32	GAT	2	200	128	6	None	30.63%
1.0E-04	32	GAT	2	200	128	6	None	29.17%
1.0E-05	32	GAT	2	200	128	6	None	14.70%
1.0E-04	64	GAT	2	200	128	6	None	26.63%
1.0E-05	64	GAT	2	200	128	6	None	8.76%
1.0E-03	64	GAT	2	200	128	6	None	1.48%

Table 8. Comparison of *PIC* models using different hyperparameters.