ATFormer: A Learned Performance Model with Transfer Learning Across Devices for Deep Learning Tensor Programs

Yang Bai, Wenqian Zhao, Shuo Yin, Zixiao Wang, Bei Yu

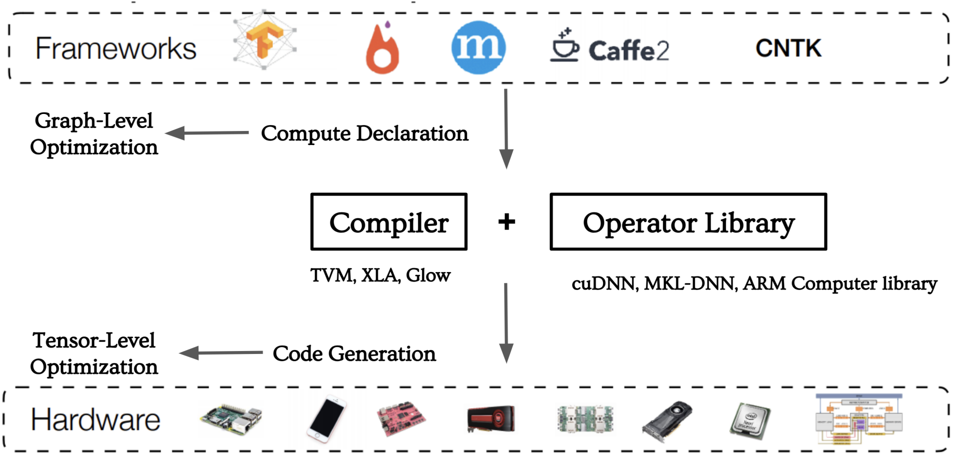
Department of Computer Science and Engineering
Chinese University of Hong Kong

EMNLP 2023

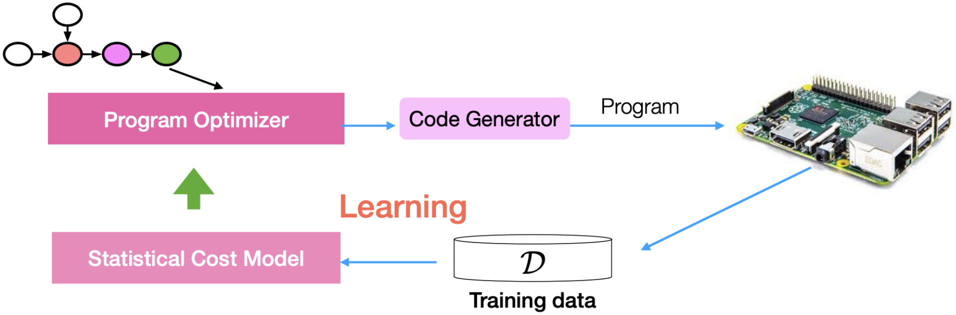
Outline

Background

Background: DL Compiler Stack



EMNLP 2023


Program Optimizer
Code Generator
Program
Learning
Statistical Cost Model
$\mathcal{D}$
Training data

Method

We describe a DNN model as a computation graph and then define some important terminologies.

Computation Graph

Computation Graph G is partitioned into a set of subgraphs S based on the graph-level optimizer.

Hierarchical Search Space

A tensor program, denoted by p, represents an implementation of the subgraph using low-level primitives that are dependent on the hardware platform. Each tensor program can be considered as a candidate in the search space. We define the hierarchical search space $\phi_{1,2}$, which decouples high-level structures ϕ_1 from low-level details ϕ_2, allowing for the efficient exploration of potential tensor candidates during the tuning process.

Method: Problem Formulation

Each search task is extracted from an independent subgraph S_i on a specific hardware platform $\mathbb{H}$. Thus, we define search task Q as follows:

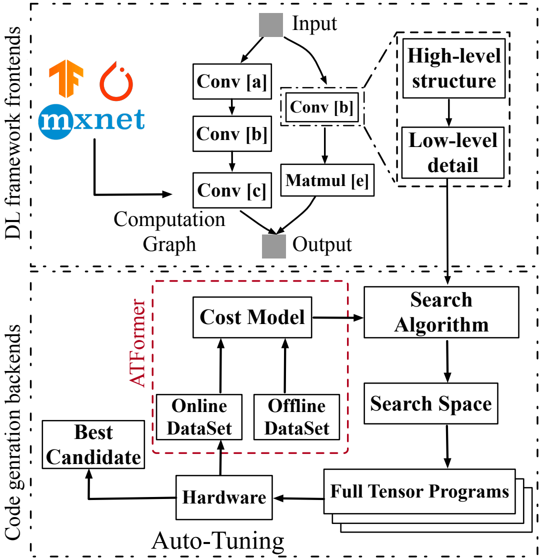
$$Q_{\mathbb{H}(S|G)} = \left\{ Q^1_{(S_1|G)}, Q^2_{(S_2|G)}, \ldots, Q^n_{(S_n|G)} \right\}, \tag{1}$$

where n is the number of subgraphs in G. Note that each subgraph S_i contains a computation-intensive operator σ and $\sigma \in S_i$. Here, we can transform a tuning problem into an optimization problem that explores the potential tensor programs in a hierarchical search space.

Given code generation function $\eth$, high-level structure generation parameters ϕ_1, low-level detail sampling parameters ϕ_2, computation-intensive operator σ and operator setting k (*e.g.*, kernel size), our goal is to use $\phi_{1,2}$ to build a hierarchical search space and generate tensor program p to achieve the optimal prediction score y^* on a specific hardware platform $\mathbb{H}$.

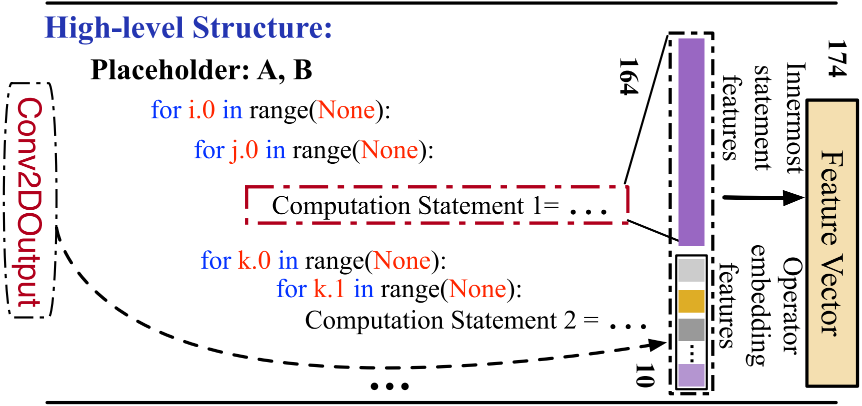
$$\begin{aligned} \phi^*_{1,2} &= \arg\max_{\phi} y, \\ y &= f_{\mathbb{H}}(\eth(\phi_1, \phi_2 | \sigma, k)). \end{aligned} \tag{2}$$

Method: Search-based Framework



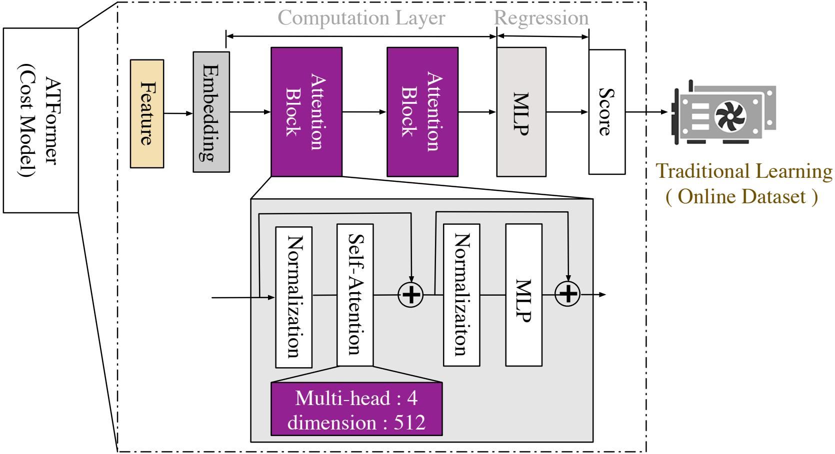
The overview of a search-based framework with computation graph, cost model, and search space.

Performance Model: Hierarchical Features



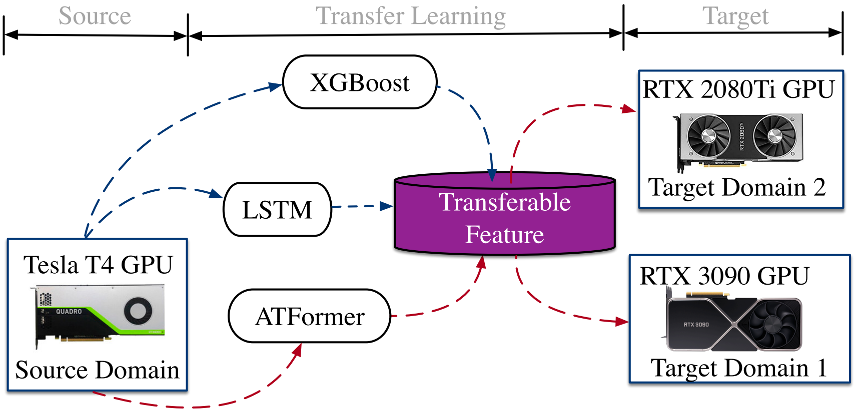
Hierarchical features of Conv2D with a full tensor program representation in the search space.

EMNLP 2023



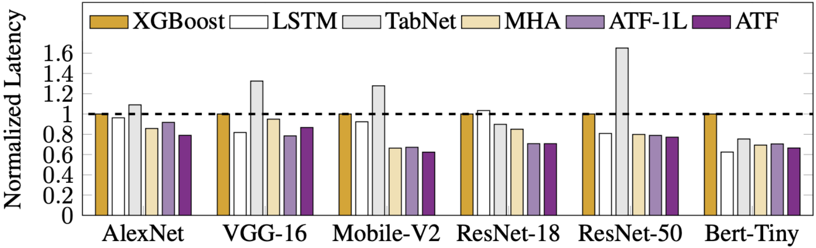
The performance model's architecture includes two attention blocks that extract coarse and fine-grained features of the tensor program, as well as a lightweight MLP layer for directly predicting the score.

Performance Model: Transfer Learning Across Devices



Transfer learning among different platforms.

Evaluation



End-to-end performance comparison of cost models across DNNs and normalized by the XGBoost.

Evaluation: Transfer Learning

	cost model (ms/s)	**XGBoost**		LightGBM		LSTM		TabNet		**MHA**		**ATFormer-1L**		**ATFormer**		**ATFormer-M**	
		latency	time	latency	time	latency	time	latency	time	latency	time	latency	time	latency	time	latency	time
	ResNet-18-2080Ti	1.47	573	1.58	770	1.29	604	1.52	748	1.32	687	1.25	706	**1.04**	787	1.23	762
RTX 2080Ti Transfer	TenSet-50	0.86	535	0.98	527	1.02	614	1.13	583	1.01	595	1.00	602	0.97	600	1.00	611
	TenSet-100	0.96	533	0.98	526	1.07	615	0.82	596	0.87	602	1.00	602	0.85	594	0.84	611
	TenSet-200	0.99	536	0.86	525	1.07	611	0.88	582	0.83	602	0.82	612	0.82	604	0.82	632
	TenSet-300	0.89	538	0.85	526	1.02	622	0.83	583	0.85	600	0.81	609	0.89	612	0.87	607
	TenSet-500	0.96	530	0.81	529	1.03	622	0.82	574	0.83	593	0.87	598	0.84	612	0.79	615
	ResNet-18-3090	1.07	589	1.11	676	1.24	762	1.64	741	1.11	658	**0.97**	661	**1.02**	677	3.01	665
RTX 3090 Transfer	TenSet-50	0.70	537	0.74	524	0.88	593	0.75	581	0.75	610	0.77	605	0.78	599	0.79	604
	TenSet-100	0.71	540	0.73	526	0.83	599	0.67	620	0.65	607	0.68	601	0.66	606	0.69	614
	TenSet-200	0.78	534	0.68	526	0.87	582	0.70	589	0.65	612	0.73	599	0.64	596	0.66	611
	TenSet-300	0.70	536	0.68	531	0.83	616	0.66	585	0.64	617	0.67	595	0.71	607	0.66	613
	TenSet-500	0.72	535	0.67	540	0.85	618	0.69	587	0.67	591	0.68	581	0.67	607	0.63	609

Table: Transferable adaptation evaluation between different GPU platforms on ResNet-18.

cost model performance (ms / s)		XGBoost		LSTM		MHA		ATFormer-1L		ATFormer		ATFormer-M	
		latency	time	latency	time	latency	time	latency	time	latency	time	latency	time
RTX 2080Ti	Traditional Learning	1.26	1026	1.02	1487	1.03	1172	1.20	1269	1.02	1382	1.71	1124
	Transfer Learning	1.23	281	1.05	348	0.99	261	1.15	264	0.99	271	0.93	266
RTX 3090	Traditional Learning	0.96	1004	1.03	1235	0.79	1125	0.87	1141	0.74	2054	0.94	2018
	Transfer Learning	0.98	287	1.02	270	0.77	261	0.83	269	0.76	267	0.65	264

Table: Pre-trained models on TenSet-500 via transfer learning with converged latency.

Evaluation: Transfer Learning

Methods	ResNet-18						MobileNet-V2						Bert-Tiny					
	(a)	(b)	(c)	(d)	(e)	(f)	(a)	(b)	(c)	(d)	(e)	(f)	(a)	(b)	(c)	(d)	(e)	(f)
mask?			✓	✓					✓	✓					✓	✓		
pre-trained?				✓	✓					✓	✓					✓	✓	
RMSE Loss?	✓						✓						✓					
Rank Loss?		✓	✓	✓	✓	✓		✓	✓	✓	✓	✓		✓	✓	✓	✓	✓
AutoTVM?						✓						✓						✓
total latency (ms)	1.42	1.04	1.23	**0.81**	0.83	1.92	0.53	0.51	0.76	**0.39**	0.40	1.29	4.18	3.41	3.97	**2.32**	2.46	5.07
search time (s)	781	787	762	**620**	611	3274	962	1000	958	**617**	604	2996	1127	1141	1150	**818**	816	3826

Table: Total latency and tuning time of different methods, using ResNet-18, MobileNet-V2 and Bert-Tiny networks for end-to-end evaluation. The relative gains obtain for batch size = 1 with 300 measurement trials.

Conclusion

Conclusion

This paper introduces ATFormer, a novel and effective design for optimizing tensor programs.

- ATFormer employs hierarchical features with varying levels of granularity to model the end-to-end compilation.
- Self-attention blocks are utilized to explore global dependencies of a complete tensor program for high-quality evaluation.
- Through transfer learning, ATFormer achieves faster-converged latency and superior transferability across different hardware platforms, outperforming previous state-of-the-art benchmarks.

THANK YOU!