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

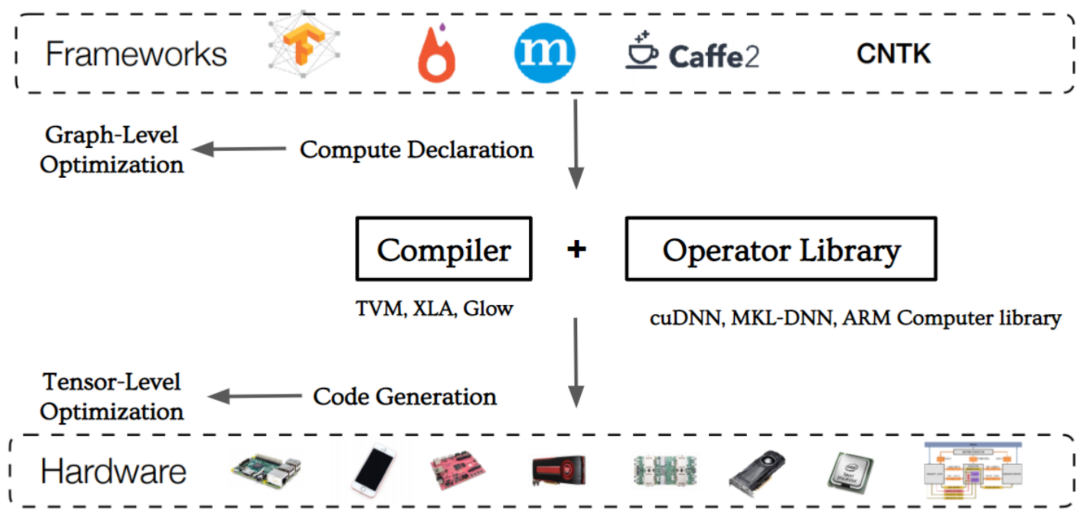
Yang Bai, Wenqian Zhao, Shuo Yin, Zixiao Wang, Bei Yu

Department of Computer Science and Engineering
The Chinese University of Hong Kong, Hong Kong SAR

EMNLP 2023

Motivation

- Pre-training a decent cost model offline requires a comprehensive dataset.
- Traditional learning makes the search very time-consuming.
- Existing tree-based models are insufficient for performance evaluation.
- Transferable knowledge is difficult to acquire across different platforms.



A big Challenge to map diverse models to diverse hardware

Problem Formulation

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

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

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

where n is the number of subgraphs in G. Note that each subgraph S_i contains a computation-intensive operator σ and $\sigma \in S_i$. Therefore, we use $Q^i_{(S_i|G)}$ to represent the i−th search task in G. Each subgraph S_i has its own search space, which is determined by the input and output shapes, data precisions, memory layout, and the hardware platform. The search space is usually large enough to cover almost all kinds of tensor candidates.

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.

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 $\mathbb{H}$.

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

The cost model f predicts score y of the tensor program p. The accuracy of the cost model f is crucial in finding ideal optimization configuration.

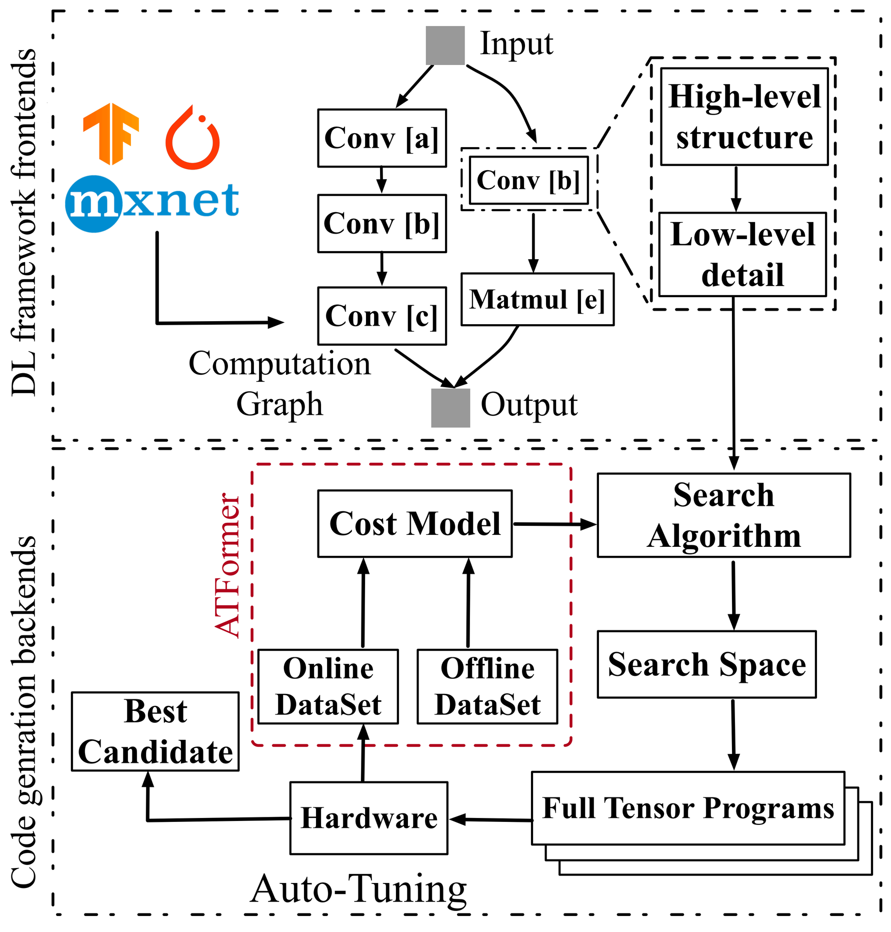


Figure 1. The overview of a search-based compilation framework with computation graph, cost model, search space, online and offline dataset.

Hierarchical Features

- Coarse-grained operator embedding features: 10 dimension.
- Fine-grained statement features: 164 dimension.

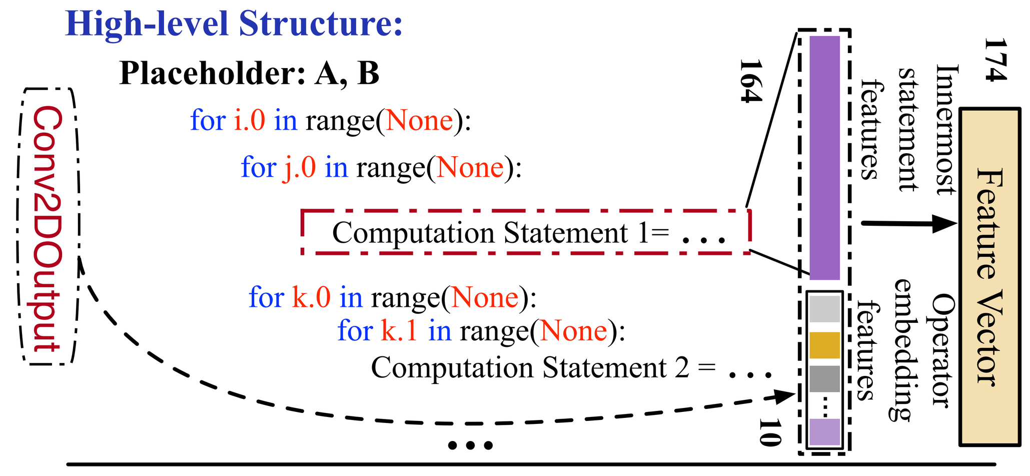


Figure 2. Hierarchical features of Conv2D with a full tensor program representation in the search space.

Model Architecture

- Kernel embedding layer: extract a compact feature representation.
- Computation layer: captures essential information from the innermost non-loop computation statements.
- Regression layer: make the final prediction.

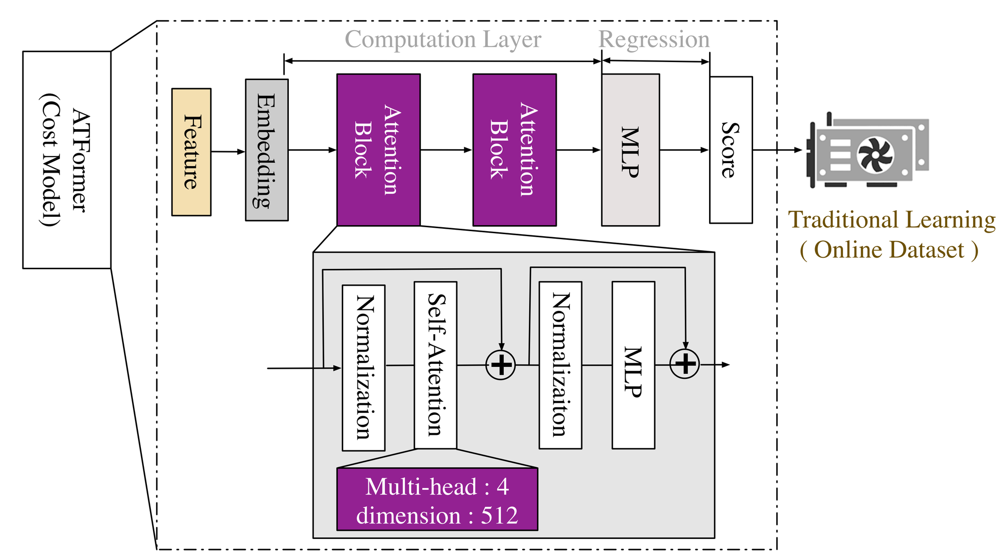


Figure 3. The performance model's architecture includes two attention blocks that extract coarse and fine-grained features of the tensor program, as well as a lightweight MLP layer for directly predicting the score.

Transfer Learning

- Source domain: collected from T4 dataset with offline.
- Target domain: collected from 3090/2080 Ti with online.
- Cost model: XGBoost, LSTM, ATFormer.

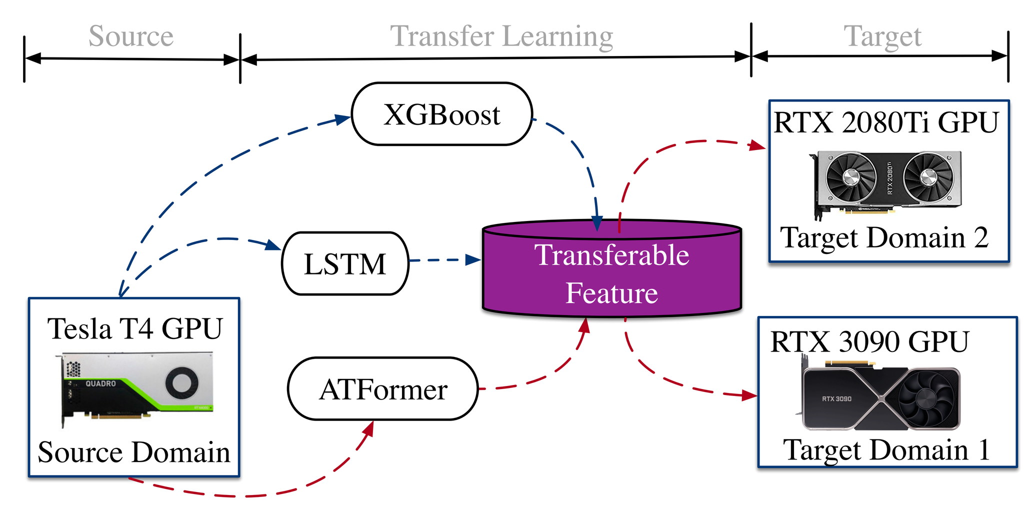


Figure 4. Transfer learning among different platforms.

Self-attention Mechanism

- All innermost non-loop statements in a full tensor program.
- Attention to capture the relationship.
- Provide accuracy and speedup the compilation time.

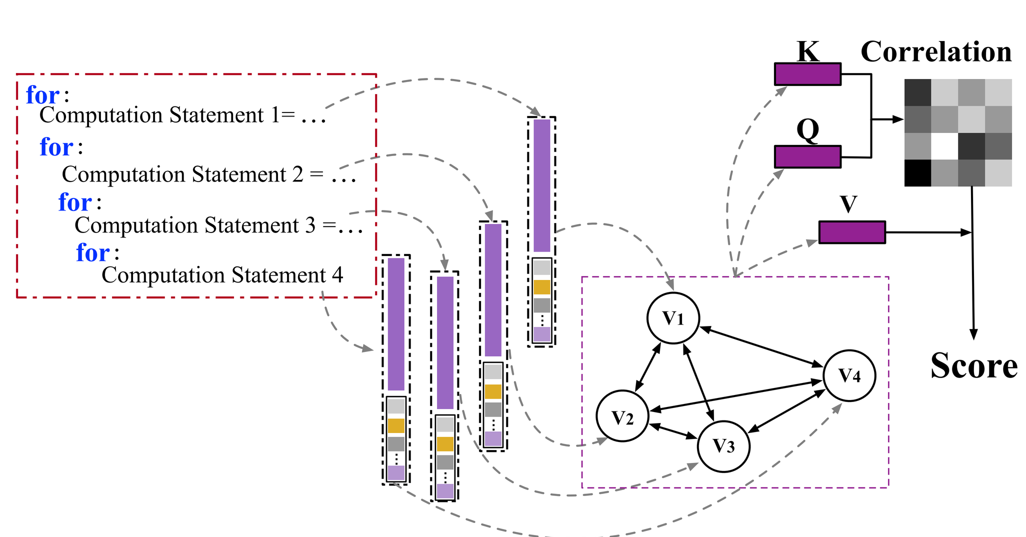


Figure 5. Self-attention between statement features.

Experimental Results

cost model (ms/s)		XGBoost latency	XGBoost time	LightGBM latency	LightGBM time	LSTM latency	LSTM time	TabNet latency	TabNet time	MHA latency	MHA time	ATFormer-1L latency	ATFormer-1L time	ATFormer latency	ATFormer time	ATFormer-M latency	ATFormer-M 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 1. Transferable adaptation evaluation between different GPU platforms on ResNet-18.

cost model performance (ms / s)		XGBoost latency	XGBoost time	LSTM latency	LSTM time	MHA latency	MHA time	ATFormer-1L latency	ATFormer-1L time	ATFormer latency	ATFormer time	Speed up latency	Speed up time
BERT_{base}	Traditional Learning	24.51	3028	32.89	3246	19.13	2890	18.77	2996	17.56	2874	1.39×	4.97×
	Transfer Learning	23.82	654	33.35	880	19.98	602	19.51	648	18.72	578		
BERT_{large}	Traditional Learning	51.63	5016	59.81	5540	53.21	5218	54.32	5312	46.54	5232	1.11×	5.10×
	Transfer Learning	52.49	1098	60.33	1302	55.88	1084	56.58	1192	47.76	1026		
GPT-2_{large}	Traditional Learning	489.12	6240	502.22	6531	467.22	6311	452.56	6380	445.52	6268	1.10×	5.69×
	Transfer Learning	491.24	1392	503.52	1594	468.29	1375	454.18	1272	447.31	1102		
GPT-3_{350M}	Traditional Learning	513.61	7789	542.23	8582	479.42	8082	468.59	7982	442.02	7891	1.16×	6.08×
	Transfer Learning	514.42	1857	543.59	2302	480.12	1890	470.52	1920	443.62	1296		

Table 2. The performance of large-scale Transformer models on TenSet-500 with transfer learning.

cost model performance (ms / s)		XGBoost latency	XGBoost time	LSTM latency	LSTM time	MHA latency	MHA time	ATFormer-1L latency	ATFormer-1L time	ATFormer latency	ATFormer time	ATFormer-M latency	ATFormer-M 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 3. Pre-trained models on TenSet-500 via transfer learning with converged latency on GPU platforms.

Methods	ResNet-18 (a)	(b)	(c)	(d)	(e)	(f)	MobileNet-V2 (a)	(b)	(c)	(d)	(e)	(f)	Bert-Tiny (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 4. 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.

architecture	n_head	hidden_dim	latency (ms)	search time (s)
MHA	2	512	3.71	652
	4	256	1.58	647
	4	512	1.24	641
	4	1024	1.29	652
	6	768	1.48	658
	8	512	1.19	658
ATFormer-1L	4	512	1.25	706
ATFormer	4	512	1.04	777
ATFormer-3L	4	512	1.23	788

Table 5. Different architecture design about ATFormer.

Methods	ResNet-18 (a)	(b)	(c)	(d)	(e)	(f)
Hierarchical features?	✓		✓		✓	
XGBoost?	✓	✓				
LSTM?			✓	✓		
ATFormer?					✓	✓
w/o Transfer total lantency (ms)	1.47	1.63	1.29	1.58	1.04	1.18
w/o Transfer search time (s)	573	618	604	648	787	796
w/ Transfer total latency (ms)	0.96	0.98	1.03	1.12	0.84	0.91
w/ Transfer search time (s)	530	599	622	689	612	632

Table 6. Hierarchical features and performance model architecture improvements for end-to-end evaluation.

cost model performance (ms / s)		XGBoost latency	XGBoost time	LSTM latency	LSTM time	MHA latency	MHA time	ATFormer-1L latency	ATFormer-1L time	ATFormer latency	ATFormer time
ResNet-18	Traditional Learning	5.28	634s	5.91	702	5.17	611	5.32	602	4.75	628
	Transfer Learning	5.21	314	5.88	432	5.16	326	5.19	384	4.74	254
ResNet-50	Traditional Learning	16.42	621	18.23	632	13.51	608	12.51	584	11.62	602
	Transfer Learning	20.01	342	21.99	461	18.11	338	17.91	362	17.02	318
VGG-16	Traditional Learning	29.52	845	31.54	967	28.55	799	28.71	796	25.49	812
	Transfer Learning	29.41	352	31.47	378	28.46	299	28.69	278	25.46	216
BERT-Tiny	Traditional Learning	13.88	862	15.22	1138	13.55	986	14.41	942	11.55	998
	Transfer Learning	13.76	339	15.47	438	13.91	345	14.39	377	11.58	320

Table 7. Pre-trained models with the converged latency on the Tensor Cores.

Conclusions

- A novel and effective design for optimizing tensor programs.
- Self-attention blocks are utilized to explore global dependencies.
- Further analysis and performance improvement on Tensor Cores.
- Transfer learning from GPUs to CPUs.