TLP: A Deep Learning-Based Cost Model for Tensor Program Tuning

ASPLOS’23

Yi Zhai, Yu Zhang, Shuo Liu, Xiaomeng Chu, Jie Peng, Jianmin Ji, and Yanyong Zhang

University of Science and Technology of China

Introduction

- Tensor program tuning is a non-convex optimization problem that need to search.
- In searching, we need to know the performance of every tuned program.
  - [S1] We can measure online.
  - [S2] We can measure in advance and save in a table then look up.
  - [S3] We can predict online.

Introduction

- Tensor program tuning is a non-convex optimization problem that need to search.
- In searching, we need to know the performance of every tuned program.
  - [S1] We can measure online.
  - [S2] We can measure in advance and save in a table then look up.
  - [S3] We can predict online.
- S1 is too slow.
- S2 is impractical.
- So study on S3.

Introduction

- **Main problem:** program tuning is expensive on time consumption.
- **Goal:** reduce the tuning time and also keep tuning performance.

- Contribution1: A new cost model based on deep learning - TLP.
  - Extract features from the schedule primitives.
  - Take primitives as embeddings and the cost predicting task becomes a **natural language regression task**.

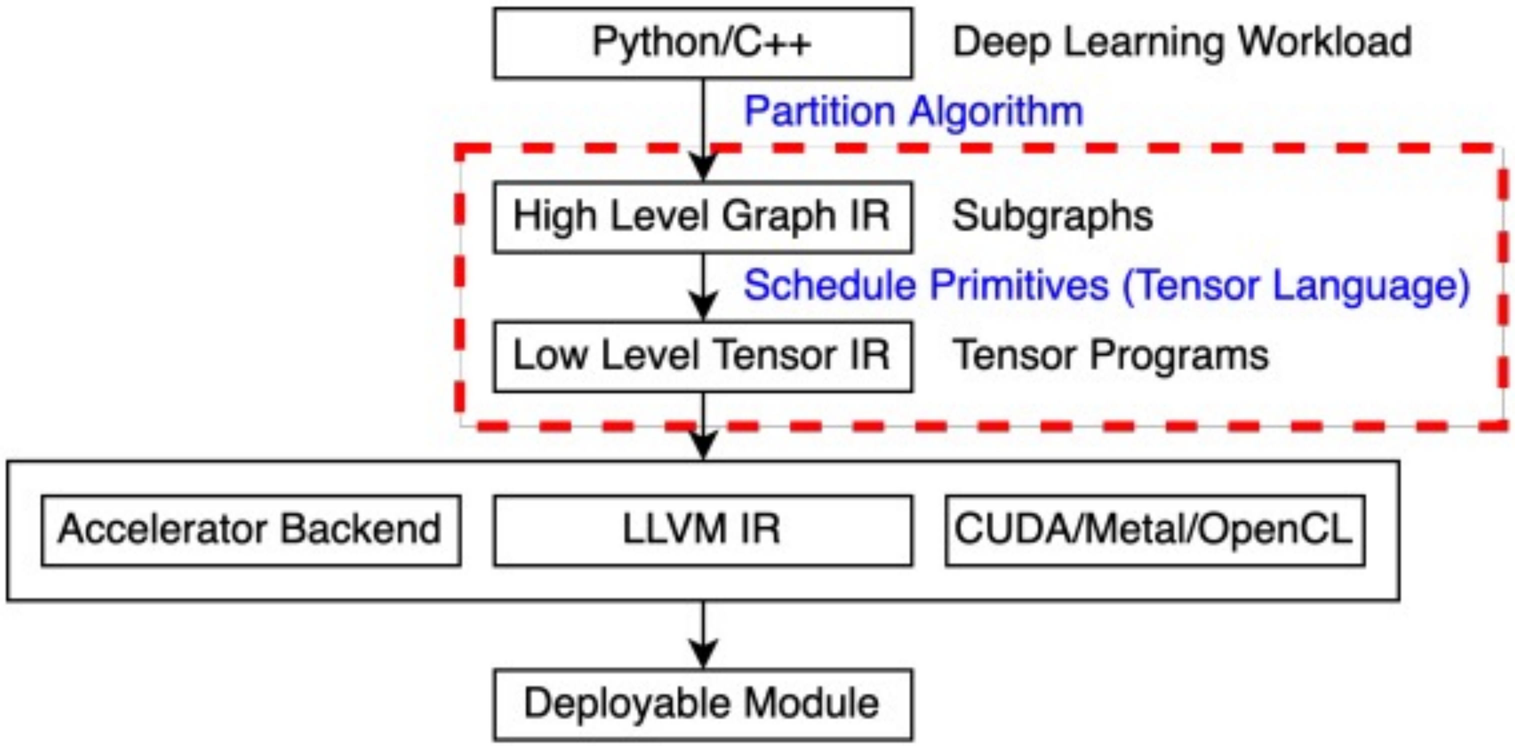
Introduction

- **Subordinate problem**: cross-hardware unavailability of cost model.
- **Goal**: cost model can perform well cross various hardware with few training data.

- Contribution2: A novel method to train cost model with cross-hardware availability based on multi-task learning – MTL-TLP.
  - Extend TLP model to a multi-task learning model aka. MTL-TLP.

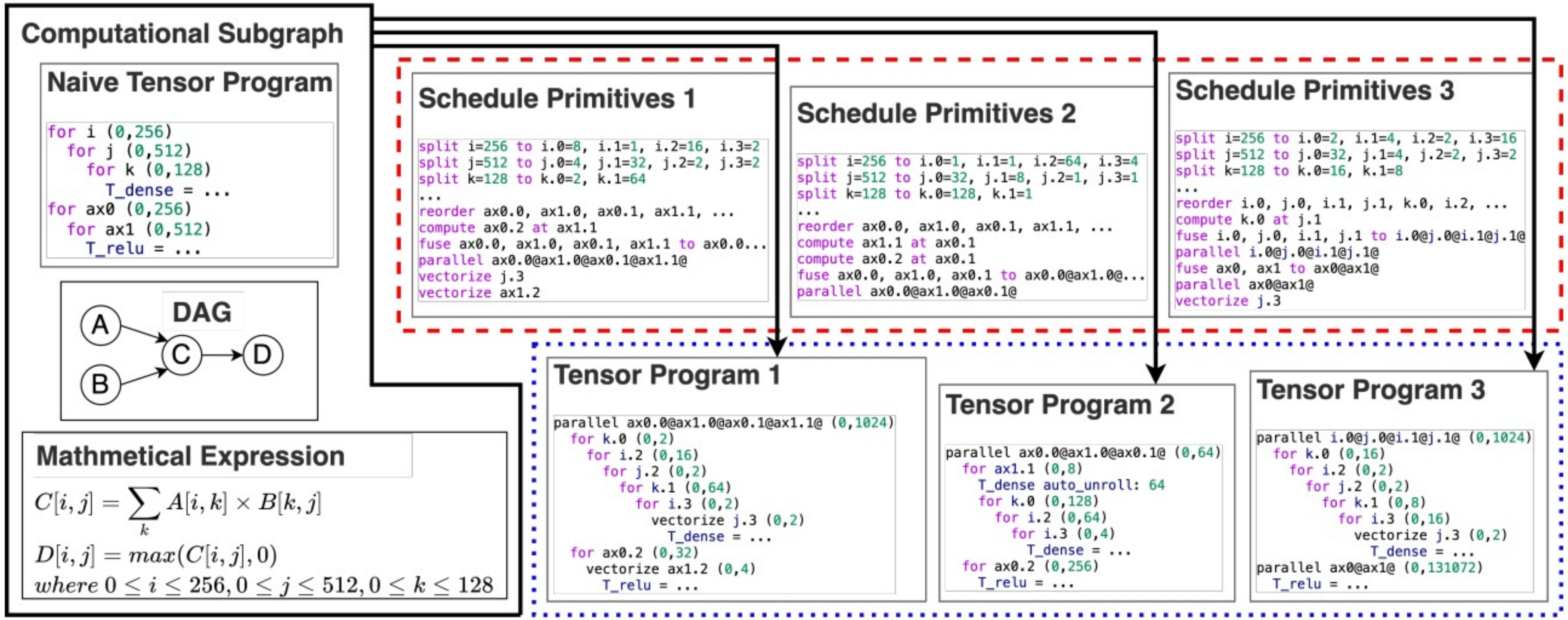
Background

- Deep learning program tuning



Background

- Deep learning program tuning



Background

- Cost models

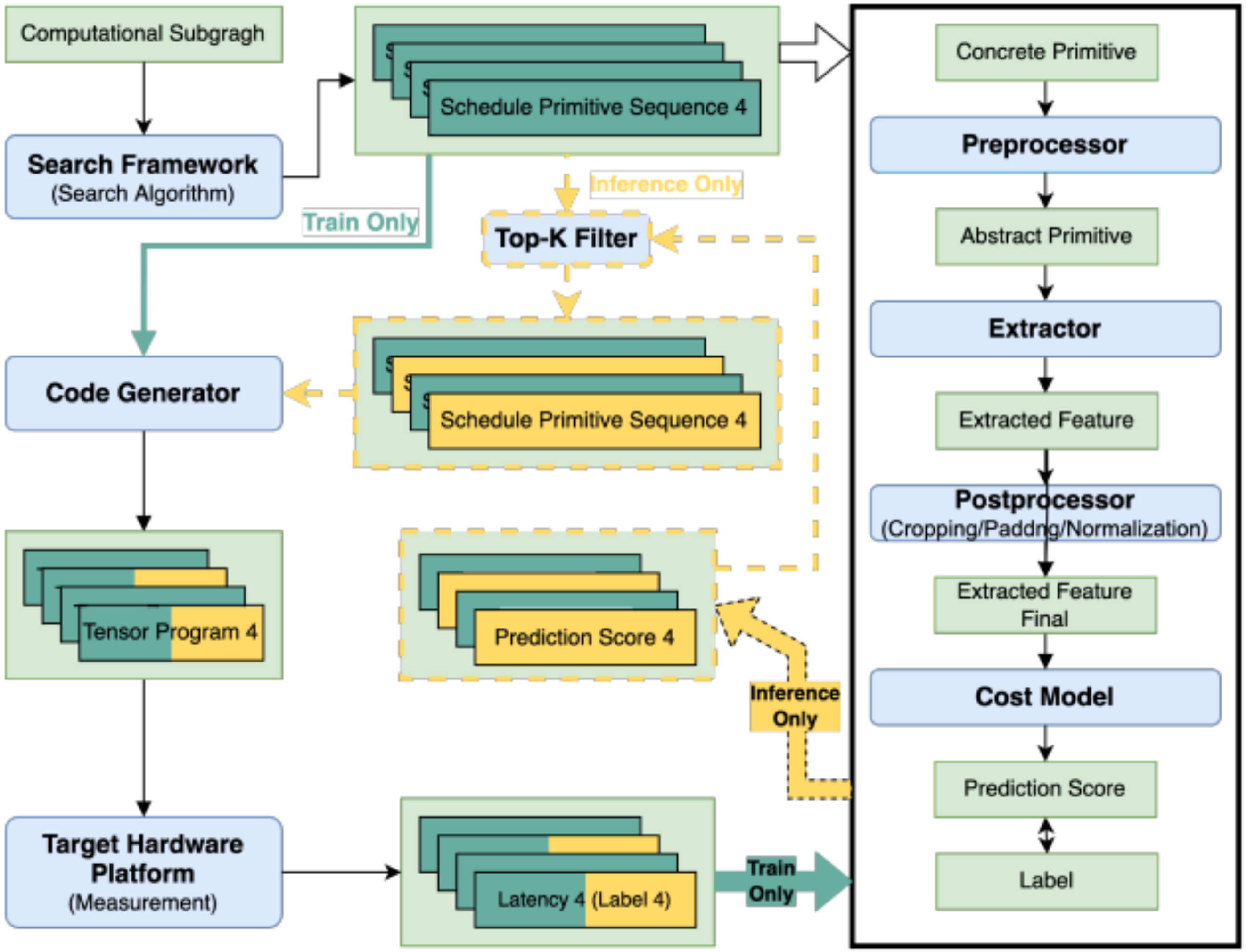
Cost Model	Representative Works
Empirical Formula Cost Model	Halide16 [25]
Online Learning Cost Model	AutoTVM [11], Ansor [38], Chameleon [2], FlexTensor [40], Halide20 [1, 3]
Offline Learning Cost Model	TenSet, TIRAMISU cost model, The work of Benoit Steiner et al. [30]

Hard to modeling.

Extra overhead in tuning.

Extract features from tensor program.

System overview



TLP

- Extract features from primitive schedules.
  - Avoid generating tensor programs and reduce tuning time.
  - Features from primitive schedules are enough for predicting.

(PrimitiveSequence)	S	$::=$	p^*
(Primitive)	p	$::=$	τ (id | num)*
(PrimitiveType)	$\mathbf{T} \ni \tau$	$::=$	split | reorder | fuse | ...
(NameParam)	id		
(Number)	num		

(a) Abstract Schedule Primitive Sequence

(Features) $f = F(p) \quad ::= \quad F_1(\tau)\ (F_2(\text{id}) | F_3(\text{num}))^*$

F : Primitive $\longrightarrow$ Features

F_1 : PrimitiveType $\longrightarrow$ OnehotVector

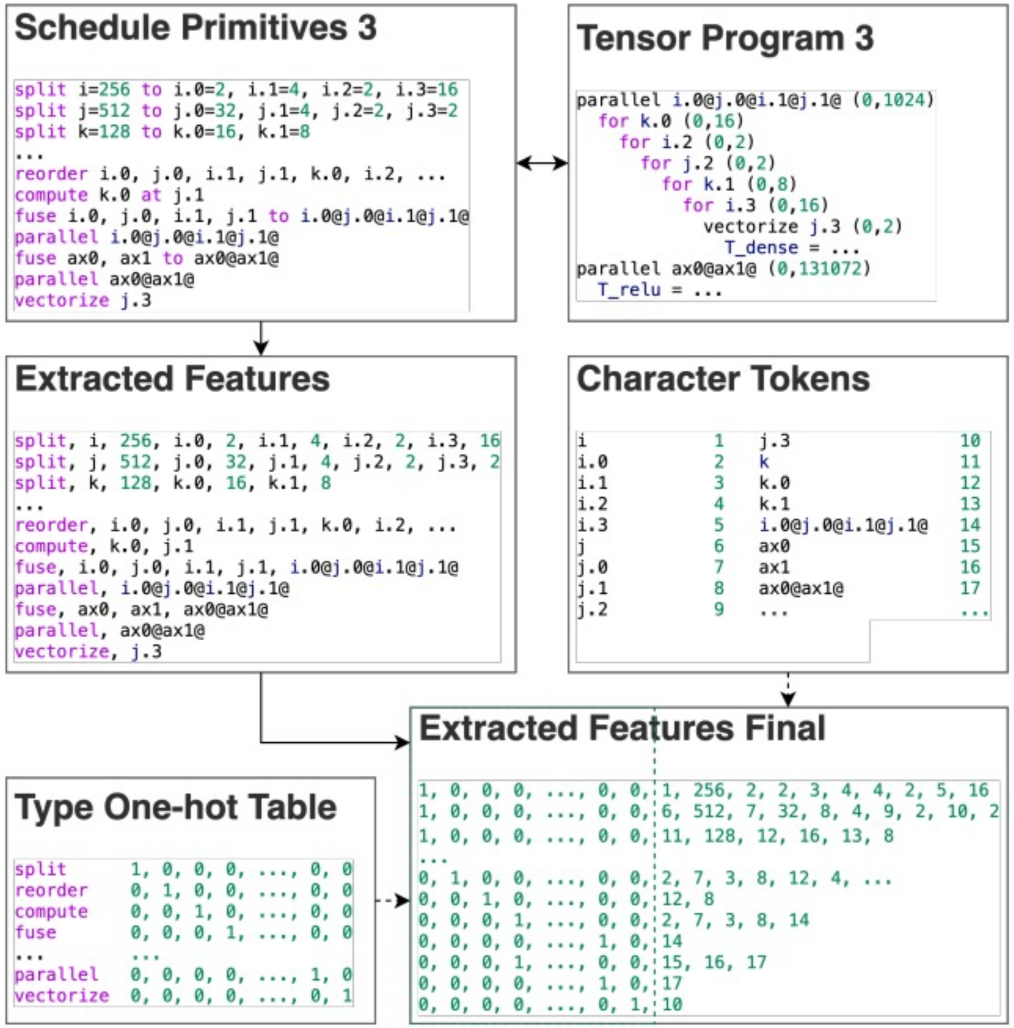
F_2 : NameParam $\longrightarrow$ Token

F_3 : Number $\longrightarrow$ Number

(b) TLP Extractor

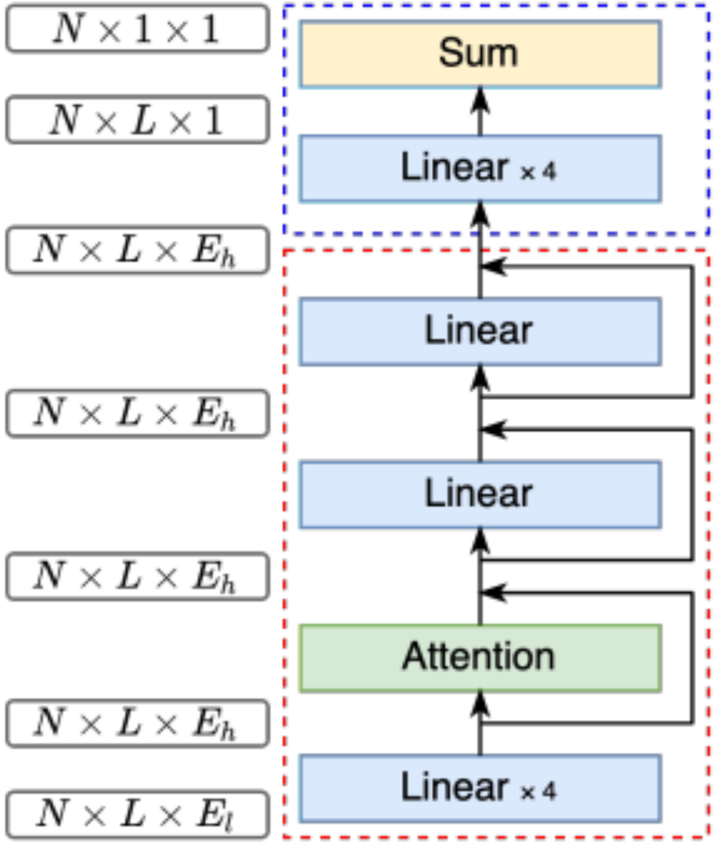
TLP

- Extraction processing



TLP

- Cost model



Train on TenSet dataset.
Using MSE loss or rank loss.

TLP

- Performance

$$top - k = \frac{\sum_m \sum_s min\_latency_{m,s} \times weight_{m,s}}{\sum_m \sum_s \min\left(latency_{m,s,i}\right) \times weight_{m,s}}, 1 \le i \le k$$

s: subgraph index
m: model index
i: predicting rank
weight_m,s: the number of times the subgraph s appears in model m

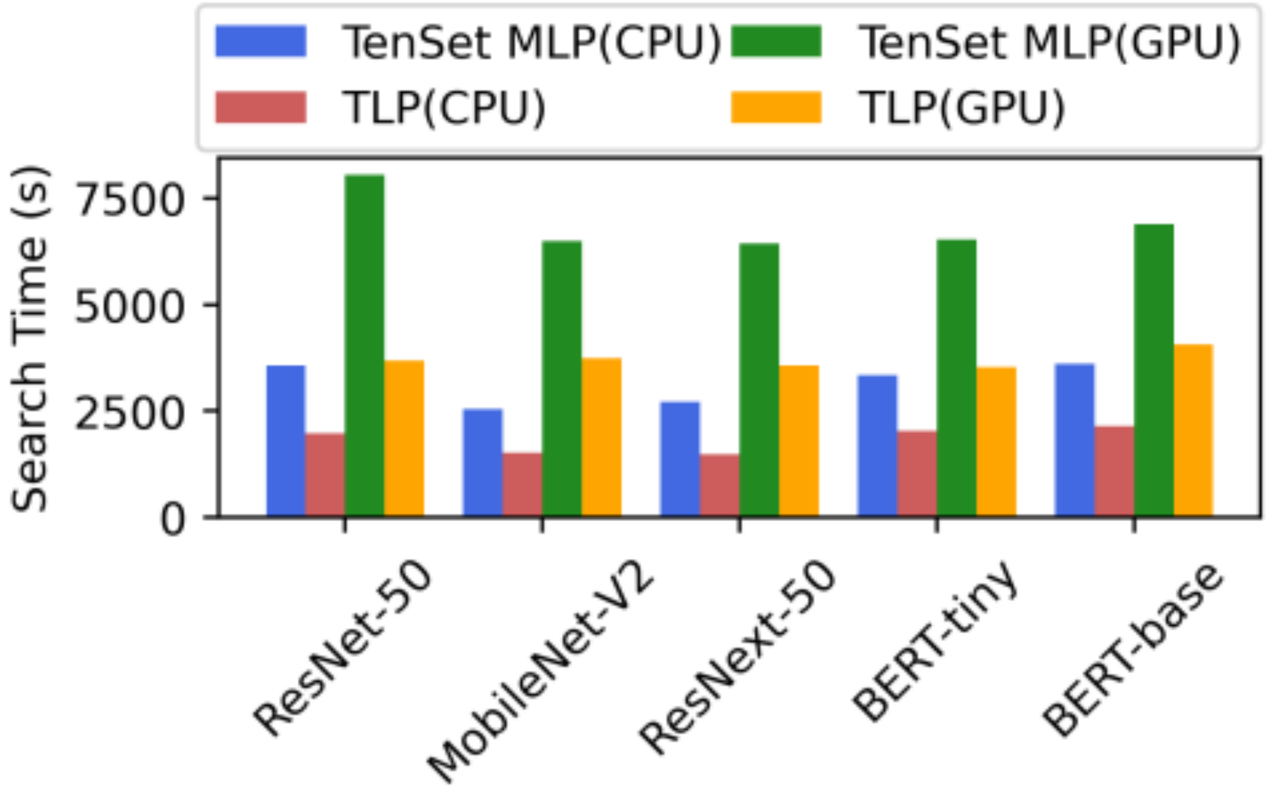
TLP

- Performance

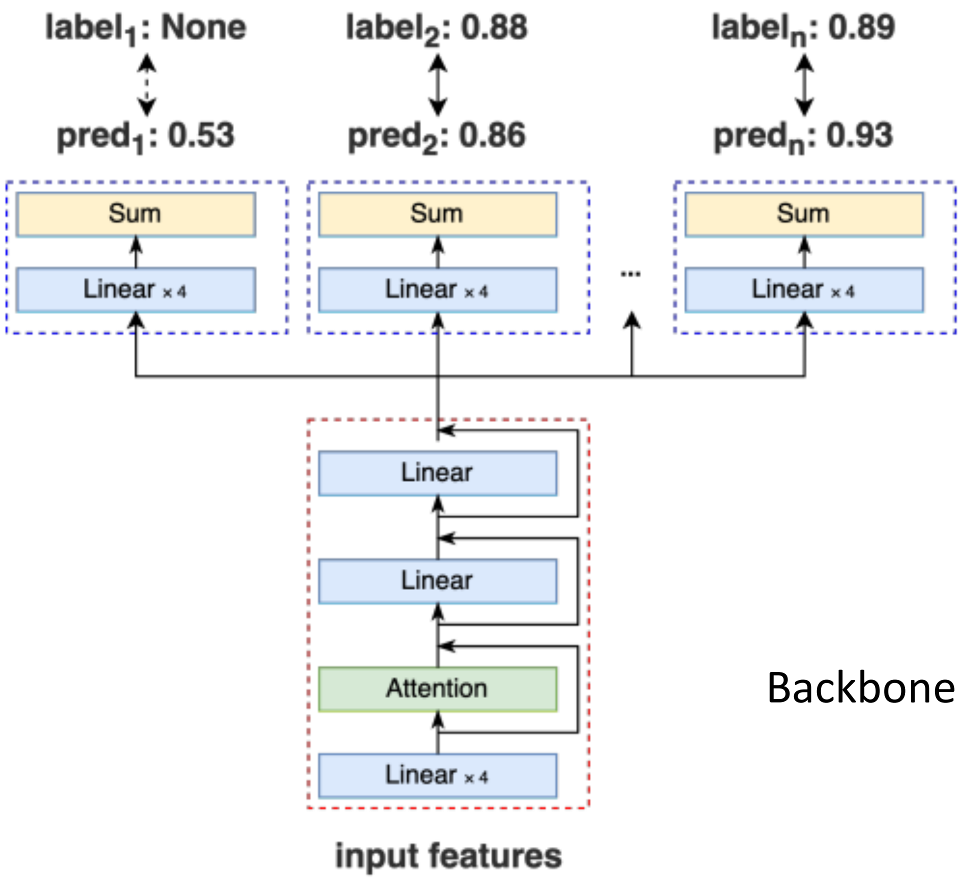
	TenSet		Ours	
	Top-1 Score	Top-5 Score	Top-1 Score	Top-5 Score
Intel Platinum 8272CL @ 2.60GHz (16 cores)	0.8748	0.9527	**0.9194**	**0.9710**
Intel E5-2673 v4 @ 2.30GHz (8 cores)	0.8332	0.8977	**0.8941**	**0.9633**
AMD EPYC 7452 @ 2.35GHz (4 cores)	0.8510	0.9175	**0.9055**	**0.9494**
ARM Graviton2 (16 cores)	0.7799	0.9049	**0.8207**	**0.9226**
Intel i7-10510U @ 1.80GHz (8 cores)	0.7776	0.8590	**0.8473**	**0.9427**
NVIDIA Tesla K80	**0.9083**	0.9629	0.9059	**0.9741**
NVIDIA Tesla T4	0.8757	**0.9528**	**0.8847**	0.9250

TLP

Tune 2000 times:



MTL-TLP



Each head is work for a specific platform.

Backbone is shared cross platforms.

MTL-TLP

		Top-1 Score	Top-5 Score
E5-2673	500K	0.6647	0.8848
E5-2673 Platinum-8272	500K ALL	0.8741	0.9385
E5-2673 Platinum-8272 EPYC-7452	500K ALL ALL	**0.8901**	**0.9520**
E5-2673 Platinum-8272 EPYC-7452 Graviton2	500K ALL ALL ALL	0.8753	0.9302

		Top-1 Score	Top-5 Score
i7-10510U Platinum-8272	500K ALL	**0.8413**	0.9202
i7-10510U E5-2673	500K ALL	0.8331	**0.9672**
i7-10510U EPYC-7452	500K ALL	0.8082	0.9122
i7-10510U Graviton2	500K ALL	0.7711	0.8909

		Top-1 Score	Top-5 Score
Tesla T4	500K	0.7971	0.8984
Tesla T4 Tesla K80	500K ALL	**0.8876**	**0.9373**

Inspiration

- AI achievements transfer to system.

- **An idea**: I think the method is relied on the programmer for character name is designed by programmer and has logical relationship between different codes. [Possible Solution] Maybe an extra preprocessing that formatting schedule primitives can work.

Thank You!

Mar 9, 2023

Presented by Mengyang Liu