Breaking the Data Barrier in Learning Symbolic Computation: A Case Study on Variable Ordering Suggestion for Cylindrical Algebraic Decomposition

Rui-Juan Jing, Yuegang Zhao and **Changbo Chen**

Chongqing Institute of Green and Intelligent Technology,
Chinese Academy of Sciences, Chongqing, China
chenchangbo@cigit.ac.cn

SCML 2026, RISC, Hagenberg, Austria
July 7, 2026

Outline

Outline

Cylindrical Algebraic Decomposition and Variable Ordering

Example

Let $f := y^2(y^2 - b^2) - x^2(x^2 - a^2)$, where $a = 4/5$, $b = 1$.

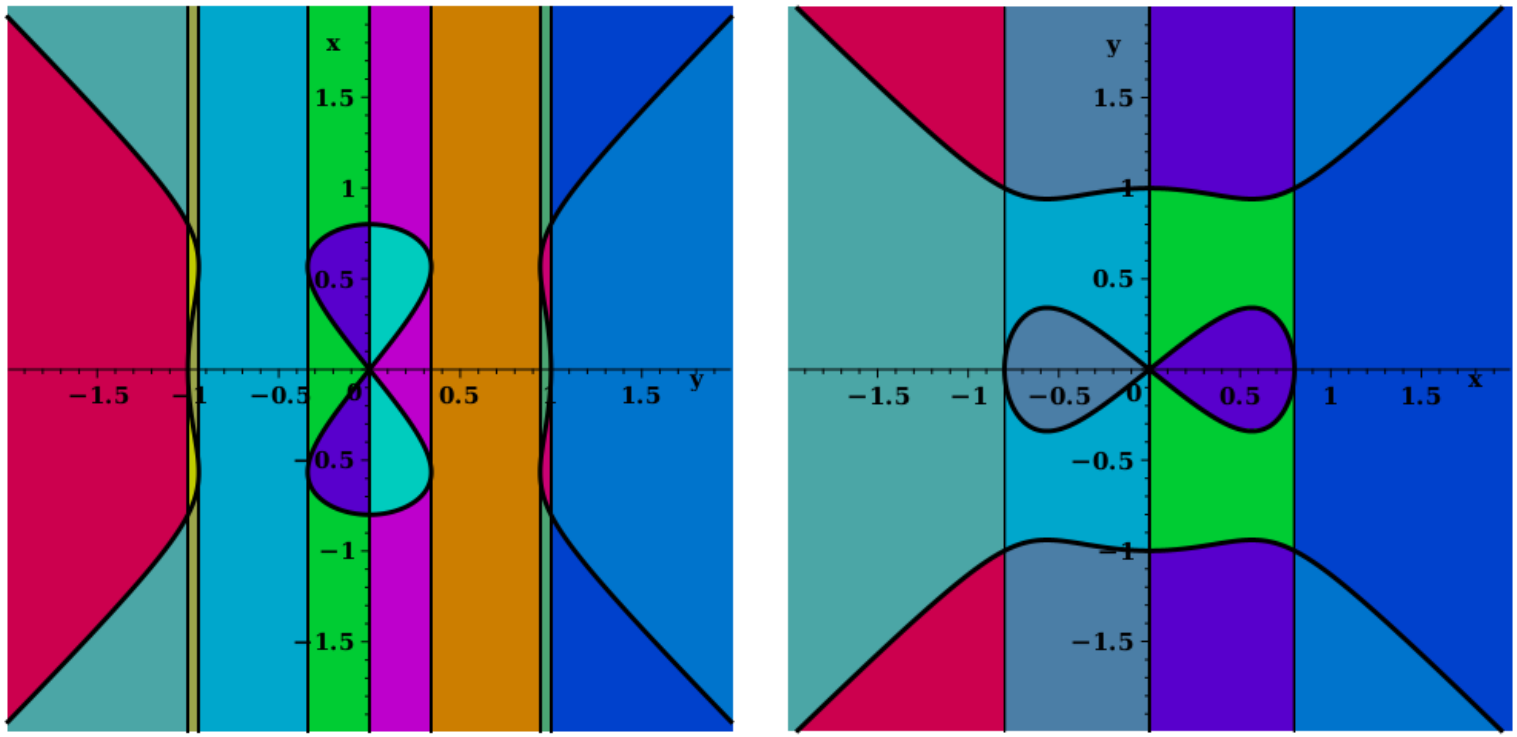


Figure: CADs of the Devil's curve under the ordering $x > y$ (89 cells) and $y > x$ (49 cells).

Existing Work on CAD

Theory/Algorithm

- G.E. Collins' original projection-lifting framework.
  - Improved projection operator (Hong, 1990; Lazard, 1994; McCallum, 1998; Brown, 2001; McCallum et al., 2019).
  - Improved lifting (Strzebonski, 2006; Iwane et al., 2013).
  - Input aware (Collins and Hong, 1991; Bradford et al., 2016).
- More geometric ways: regular chain (Chen et al., 2009) and Gröbner bases (Chen, 2025).
- SMT Oriented (Jovanovic and de Moura, 2012; Brown and Kovsta, 2015; Li et al., 2023b; Nalbach et al., 2024).

Software

- QEPCAD (Collins and Hong, 1991; Brown, 2003)
- Mathematica (Strzebonski, 2000),
- Redlog (Seidl and Sturm, 2003), Maple (Chen et al., 2009)
- Z3 and many other SMT solvers

Variable Ordering Matters!

Theoretically matters (Brown and Davenport, 2007)

Depending on orderings, doubly exponential or constant output size can occur for certain classes of problems.

Experimentally matters (Chen et al., 2024)

Dataset	A_{best}	A_{svob}	A_{svoc}	A_{gmods}	A_{tone}
DQ-3	6.17	42.36	52.22	35.39	**33.09**
DQS-3	9.42	161.74	203.6	145.27	**136.25**
DQ-4	0.76	2.94	3.4	2.87	**2.76**
DQS-4	16.52	**61.23**	63.61	62.78	63.17

Work on Variable Ordering Selection

Heuristic approaches

- Relying only on input information
  - For general system: svob (Brown, 2004), svoc (Chen et al., 2011), gmods (del Río Almajan and England, 2022), tone (Pickering et al., 2024a).
  - For variable sparse (chordal) system (Li et al., 2023a).
- Relying on projection information (Dolzmann et al., 2004; del Río Almajan and England, 2022; Chen et al., 2020).

ML approaches

- Choose the best heuristic approach (Huang et al., 2014).
- Classification models (England and Florescu, 2019; Zhu and Chen, 2020; Chen et al., 2020).
- Reinforcement learning (Jia et al., 2023; Jing et al., 2024)

Transformer

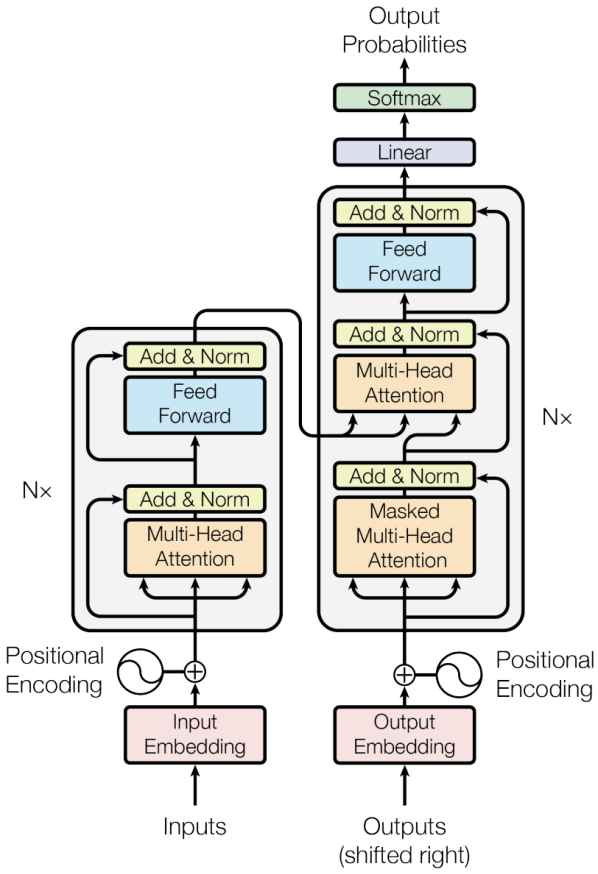


Figure 1: The Transformer - model architecture.

Use Transformer in a direct way

- Prepare a dataset of (*sys*, *order*).
- Tokenize both input and output.
- Train with teacher-forcing mode.
- Predict with auto-regressive mode.

Potential problem

- Dataset (size?, sequence length?)
- Model size? GPU global memory?

Related Work on Applying Transformer on Symbolic Computation

End-to-end learning for complex tasks

- Symbolic integration (Lample and Charton, 2020); Lyapunov function (Alfarano et al., 2024); Gröbner bases (Kera et al., 2024; Malhou et al., 2026).
- Status: Simple input and output. Dataset size matters.

Learning primitive/basic operations

- Integer addition/multiplication (Cho et al., 2025).
- Status: Length generalization is challenging. Prompt matters.

Learning strategies to replace existing heuristics

- Explainable AI for CAD ordering (Pickering et al., 2024a); Monte Carlo Tree Search for SMT (Lu et al., 2024).
- Status: One magnitude speedup is still missing.

Outline

Motivation

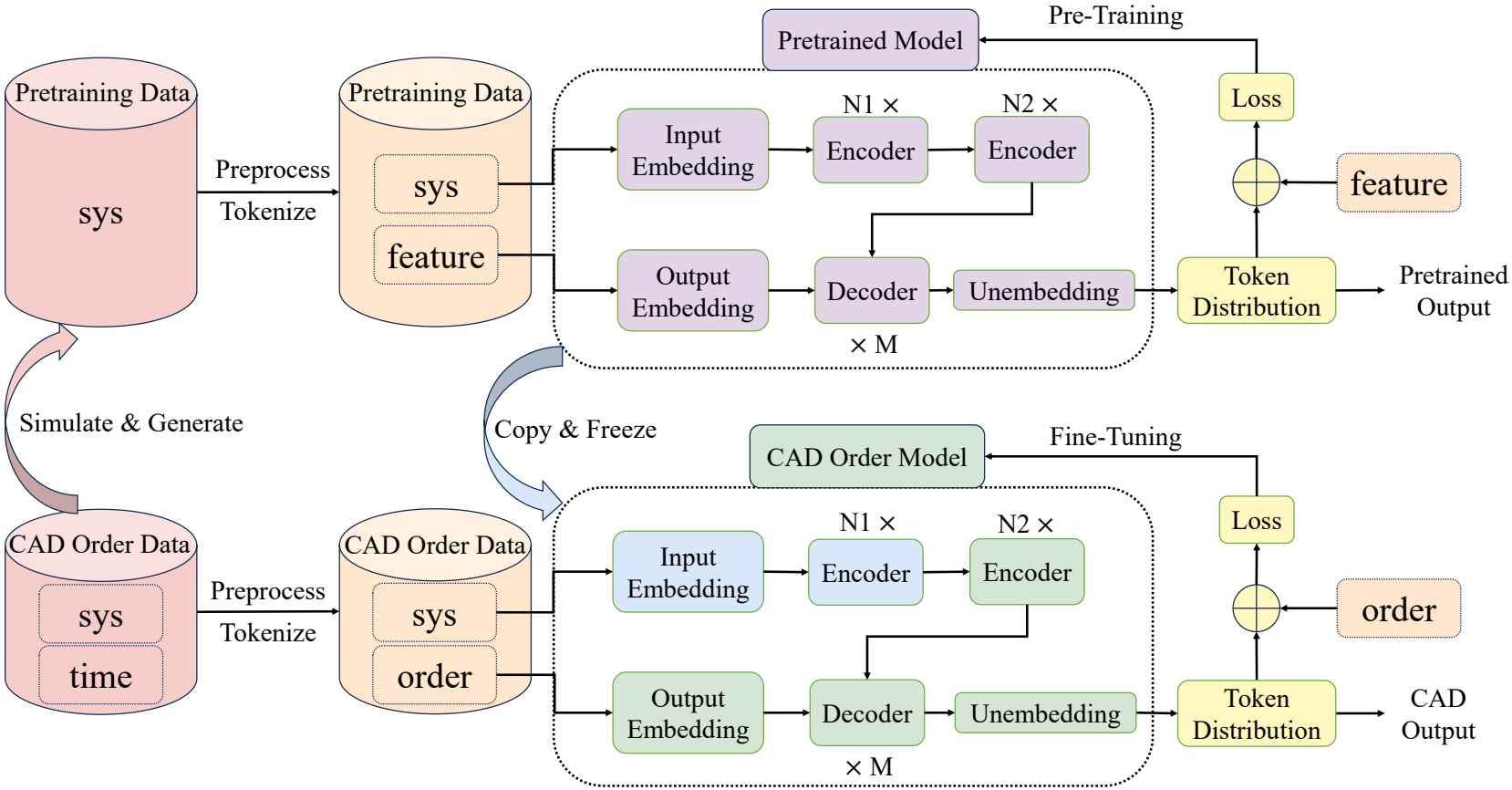
The successful experience on other tasks

- A huge amount of labelled data (forward or backward generation).
- Both input and output are relatively simple expressions.

The challenge for this task

- Forward generation is expensive (140 CPU days for 20K examples).
- No obvious backward way for generation.
- Forward generation involves constructing CAD, thus a complex task!
- Labelled data is scarce by nature.

The Pre-training & Fine-tuning Framework



Feature Operators

Candidate features (Brown, 2004; Florescu and England, 2019).

- For a term $T \in \mathcal{T}_f$ and a variable $v \in X$, let
  - $\mathsf{d}(T, v) = \deg_v(T)$
  - $\mathsf{a}(T, v) = \begin{cases} 1 & \text{if } v \text{ appears in } T \\ 0 & \text{else} \end{cases}$
  - $\mathsf{e}(T, v) = \begin{cases} \deg(T) & \text{if } v \text{ appears in } T \\ 0 & \text{else} \end{cases}$
- For $U, V \in \{\max, \mathrm{sum}, \mathrm{avg}\}$ and $o \in \{\mathsf{d}, \mathsf{a}, \mathsf{e}\}$, let

$$U\_V\_o(F, v) \;=\; \underset{f \in F}{U} \; \underset{T \in \mathcal{T}_f}{V} \; o(T, v)\,.$$

We select 11 of them:

$$R = \{U\_V\_o \mid U, V \in \{\max, \mathrm{sum}\}\,,\; o \in \{\mathsf{d}, \mathsf{a}, \mathsf{e}\}\} \setminus \{\mathrm{max\_max\_a}\}\,.$$

and use the order suggested in (Pickering et al., 2024b):

$$\begin{aligned} &\mathrm{sum\_max\_d}, \mathrm{sum\_max\_e}, \mathrm{sum\_sum\_d}, \mathrm{sum\_sum\_e}, \mathrm{sum\_sum\_a}, \\ &\mathrm{max\_max\_d}, \mathrm{max\_sum\_e}, \mathrm{max\_max\_e}, \mathrm{sum\_max\_a}, \mathrm{max\_sum\_d}, \mathrm{max\_sum\_a}. \end{aligned} \tag{1}$$

Let $\mathrm{fm}(F)$ be $\#R \times n$ matrix with entries $\mathrm{fm}(F)_{ij} = R_i(F, x_j)$.

Pre-training Features and Tasks

- Let $F \subset \mathbb{Q}[x_1, \ldots, x_n]$, $v \in X$, $q = \prod_{f \in F} f$.
- Let $g = \mathrm{gsfp}(q)$ be the greatest square free part of q.
- Let $r_v = \mathrm{res}_v(g, \frac{\partial g}{\partial v})$ and $sr_v = \mathrm{gsfp}(r_v)$.
- Let $\mathrm{pf}(F, v)$ be the projection of F w.r.t. v.

We define

- $\texttt{feature\_f} = \mathrm{fm}(F)$,
- $\texttt{feature\_p} = \mathrm{fm}(\bigcup_{1 \le i \le n} \mathrm{pf}(F, x_i))$,
- $\texttt{feature\_s} = \mathrm{fm}(\bigcup_{1 \le i \le n} \mathrm{pf}(\{g\}, x_i))$,
- $\texttt{feature\_e} = \{[(\deg_v(T) \mid v \in X) \mid T \in \mathcal{T}_f \setminus \mathbb{Q}] : f \in F\}$,
- $\texttt{feature\_m} = [(\deg_v(T) \mid v \in X) \mid T \in \mathcal{T}_q \setminus \mathbb{Q}]$,
- $\texttt{feature\_r} =$
  $[(\deg(r_v) \mid v \in X), (\deg(sr_v) \mid v \in X), (\#\mathcal{T}_{r_v} \mid v \in X), (\#\mathcal{T}_{sr_v} \mid v \in X)]$.

For $i \in \{f, p, s, e, m, r\}$, denote by `task_i` the pre-training task for predicting the feature *feature_i*.

Tokenization Scheme

Category	Tokens	Category	Tokens
special_tokens	$\langle s \rangle$ $\langle /s \rangle$ $\langle$pad$\rangle$	separators	$\langle$sep$\rangle$; ,
variables $x_1 \ldots x_n$	x1 ... xn	digits_of_exponents	0–9
arithmetic_operators	+ - * $\wedge$	digits_of_coefficients	c0–c9
relational_operators	= > $\geq$ < $\leq$ $\neq$	digits_of_feature_vectors	0–9

- System:
  $\{-6x_1^3x_2 - 4x_1x_2x_3^2 + 2x_2^2x_3 + 1 = 0, -5x_3^4 + x_3^3 - 7 = 0\}$.
- Tokens: $\{\langle s \rangle$, -, c6, *, x1, $\wedge$, 3, *, x2, $\wedge$, 1, *, x3, $\wedge$, 0, -, c4, *, x1, $\wedge$, 1, *, x2, $\wedge$, 1, *, x3, $\wedge$, 2, +, c2, *, x1, $\wedge$, 0, *, x2, $\wedge$, 2, *, x3, $\wedge$, 1, +, c1,=, c0, ,, -, c5, *, x1, $\wedge$, 0, *, x2, $\wedge$, 0, *, x3, $\wedge$, 4, +, x1, $\wedge$, 0, *, x2, $\wedge$, 0, *, x3, $\wedge$, 3, -, c7,=, c0, $\langle /s \rangle\}$.

Outline

More Heuristic Approaches

Table: Heuristic methods for CAD.

Heuristic	Features
`svob`	(max_max_d, max_max_e, sum_sum_a)
`svoc`	(max_max_d, max_l, sum_max_d)
`gmods`	(sum_max_d)
`tone`	(sum_max_d, avg_avg_d, sum_sum_d)
`slm`	(sum_max_d, max_l, max_max_d)
`SmSsAa`	(sum_max_d, sum_sum_d, avg_avg_d)
`SmAaMl`	(sum_max_d, avg_avg_d, max_l)
`isf`	`feature_f`
`psf`	`feature_p`
`ipf`	(`feature_f`, `feature_p`)
`pgf`	pseudo graph feature vector of F

Performance of Heuristic Approaches

Heuristics	Abs_Acc (%)				Rel_Acc (%)				Time_Ratio			
	SMT	DQ-3	DQ-4	DQ-4b	SMT	DQ-3	DQ-4	DQ-4b	SMT	DQ-3	DQ-4	DQ-4b
svob	55.45	37.86	20.41	18.06	67.32	41.07	27.69	24.38	1.25	6.86	3.85	4.50
svoc	53.88	37.45	18.26	16.64	65.36	40.61	25.16	22.64	1.31	8.46	4.46	4.41
gmods	**56.72**	44.57	15.78	15.30	**68.89**	48.13	22.16	20.98	**1.18**	5.73	3.76	3.68
tone	54.17	46.96	18.92	17.54	66.44	50.66	26.05	23.82	**1.18**	5.36	3.61	3.87
slm	55.64	46.20	18.35	17.45	67.91	49.81	25.27	23.60	**1.18**	5.65	3.84	3.69
SmSsAa	54.96	46.90	18.81	17.48	67.12	50.51	25.93	23.76	1.19	**5.28**	3.62	3.84
SmAaMl	54.27	47.75	20.31	18.66	66.63	51.40	27.89	25.30	**1.18**	5.37	**3.57**	3.86
isf	55.25	**48.53**	20.71	19.20	67.81	52.32	28.15	25.75	1.19	**5.28**	3.60	3.79
psf	40.63	29.94	**22.15**	**21.04**	50.74	32.55	**29.78**	**28.04**	1.55	7.65	4.23	**3.41**
ipf	55.05	48.51	21.51	19.86	67.71	**52.35**	29.24	26.63	1.20	**5.28**	3.60	3.78
pgf	53.58	47.75	19.44	18.13	66.05	51.38	26.48	24.38	1.33	5.51	3.68	3.75

The best ones

- SMT: gmods
- DQ3: isf, ipf
- DQ4: psf, SmAaMl
- DQ4b: psf

Outline

Performance on Pre-training Tasks

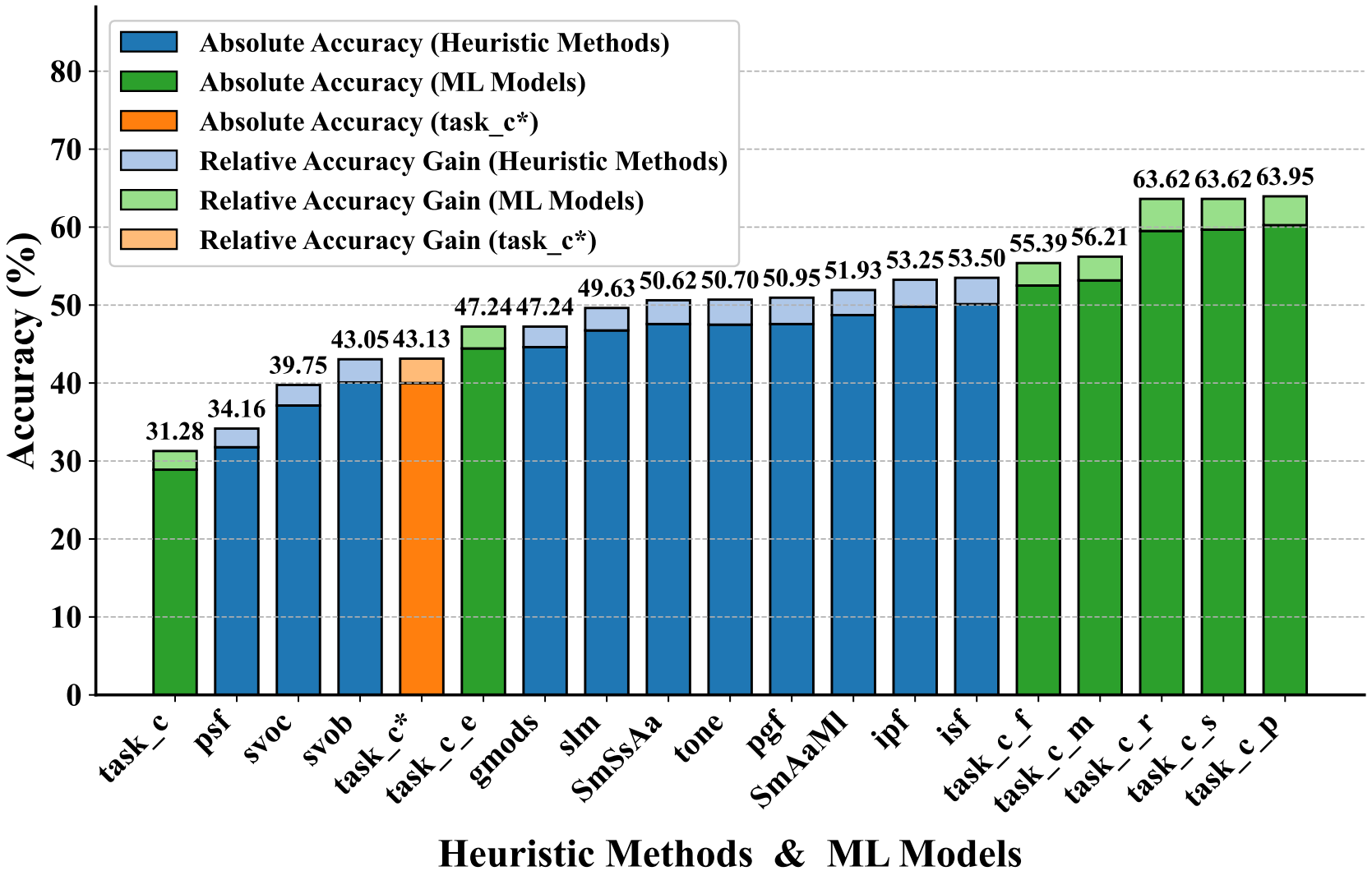
Table: Accuracies of pre-training tasks.

Dataset	Category	task_e	task_f	task_m	task_p	task_r	task_s
DQ-3	valid	100.00%	98.00%	93.49%	25.29%	20.28%	9.30%
	test-valid	99.99%	98.08%	93.65%	24.75%	19.93%	9.22%
	test	100.00%	98.11%	93.78%	24.87%	19.83%	9.02%
DQ-4	valid	99.99%	99.83%	94.03%	65.88%	49.80%	41.01%
	test-valid	99.98%	99.85%	93.91%	65.49%	49.44%	41.24%
	test	100.00%	99.79%	94.25%	65.14%	49.36%	40.76%
DQ-4b	valid	99.98%	99.90%	93.08%	56.62%	41.57%	35.19%
	test-valid	99.98%	99.90%	92.75%	56.02%	41.31%	35.46%
	test	100.00%	99.99%	93.32%	56.09%	41.03%	35.10%

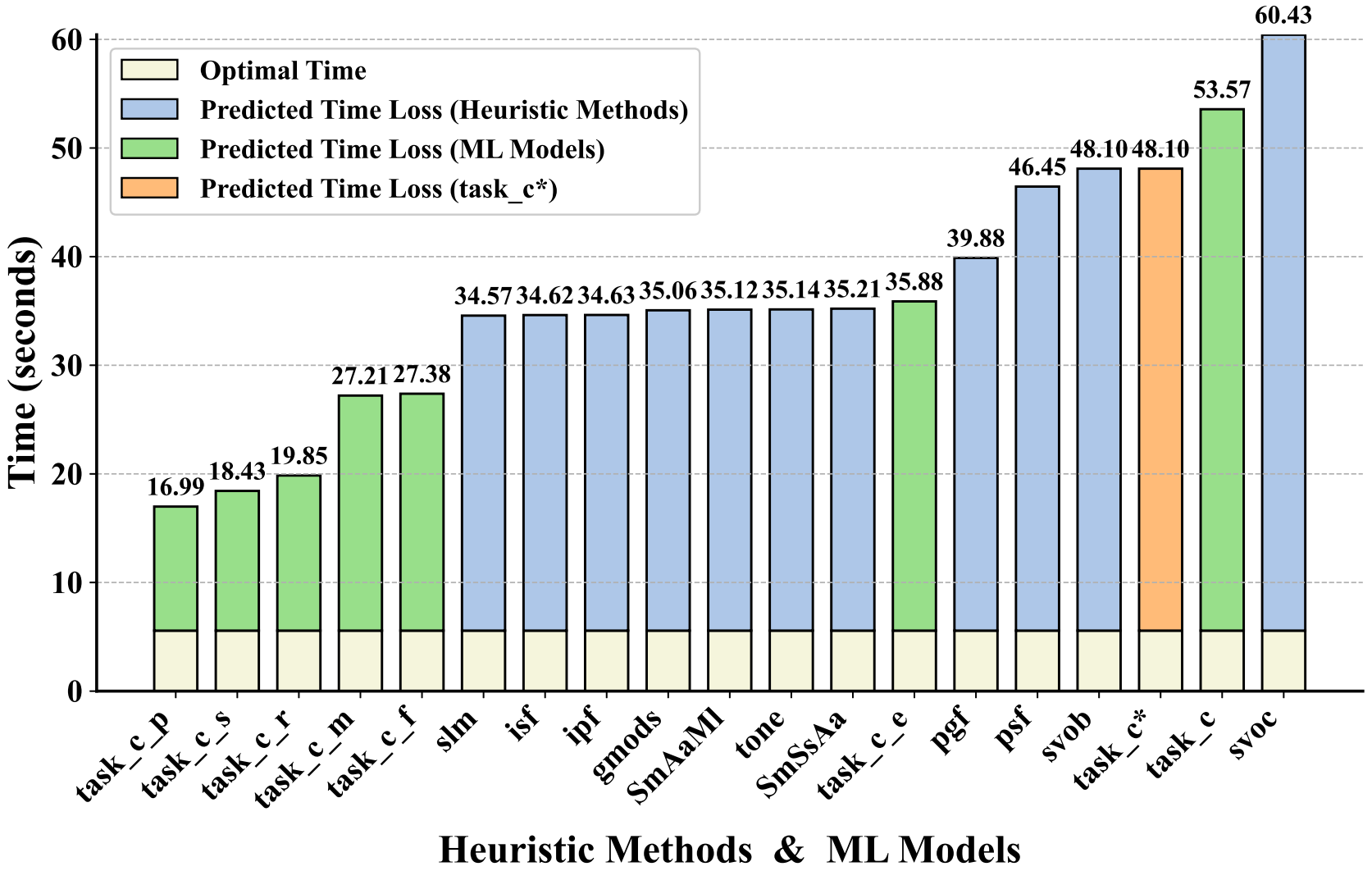
Table: Mean digit length for each pre-training task on testing set.

Dataset	task_p	task_r	task_s
DQ-3	1.83	2.12	2.75
DQ-4	1.34	1.60	2.20
DQ-4b	1.48	1.79	2.46

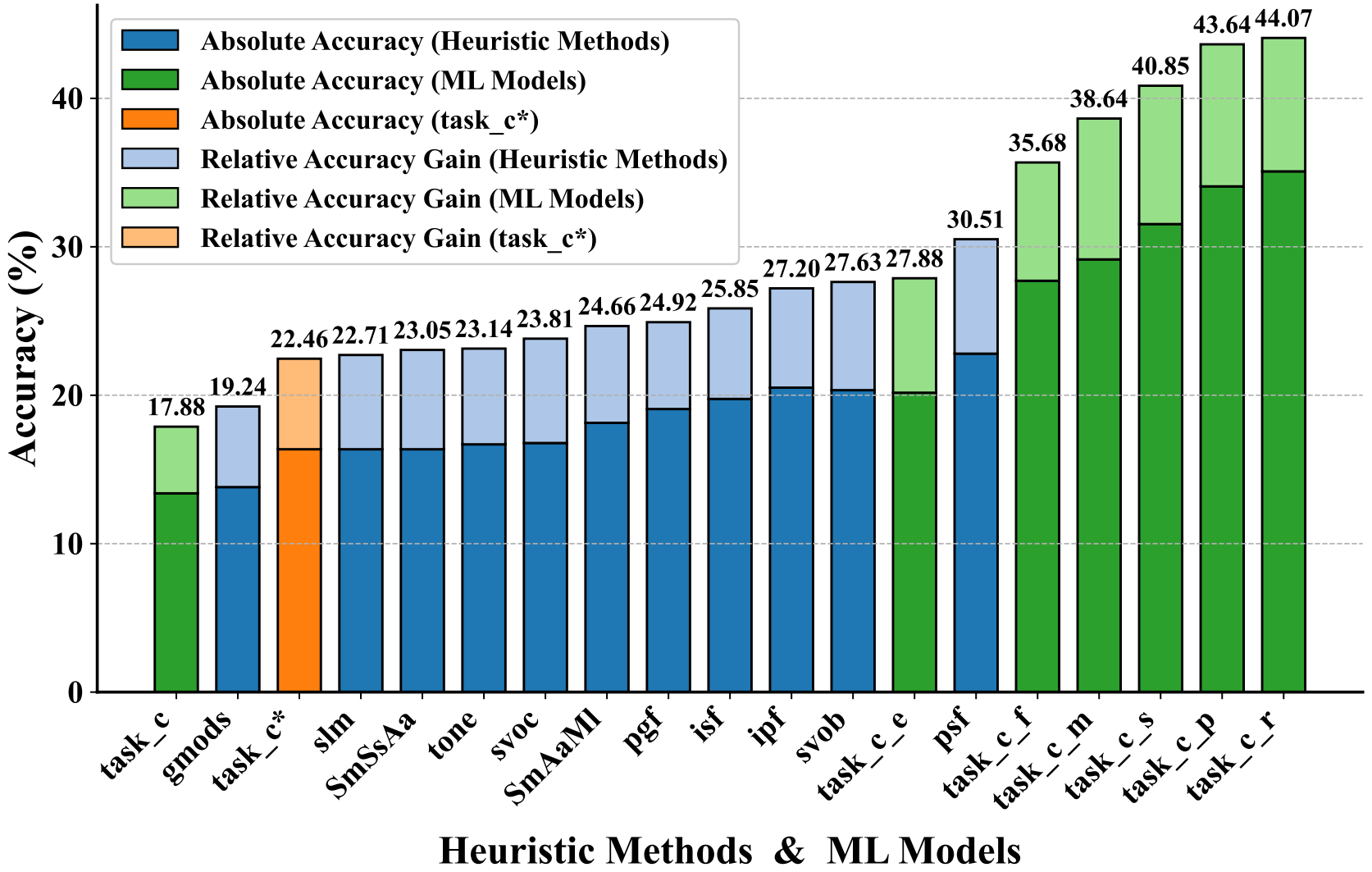
Random DQ-3 (I)


Absolute Accuracy (Heuristic Methods)
Absolute Accuracy (ML Models)
Absolute Accuracy (task_c*)
Relative Accuracy Gain (Heuristic Methods)
Relative Accuracy Gain (ML Models)
Relative Accuracy Gain (task_c*)
Accuracy (%)
0
10
20
30
40
50
60
70
80
31.28
34.16
39.75
43.05
43.13
47.24
47.24
49.63
50.62
50.70
50.95
51.93
53.25
53.50
55.39
56.21
63.62
63.62
63.95
task_c
psf
svoc
svob
task_c*
task_c_e
gmods
slm
SmSsAa
tone
pgf
SmAaMl
ipf
isf
task_c_f
task_c_m
task_c_r
task_c_s
task_c_p
Heuristic Methods & ML Models

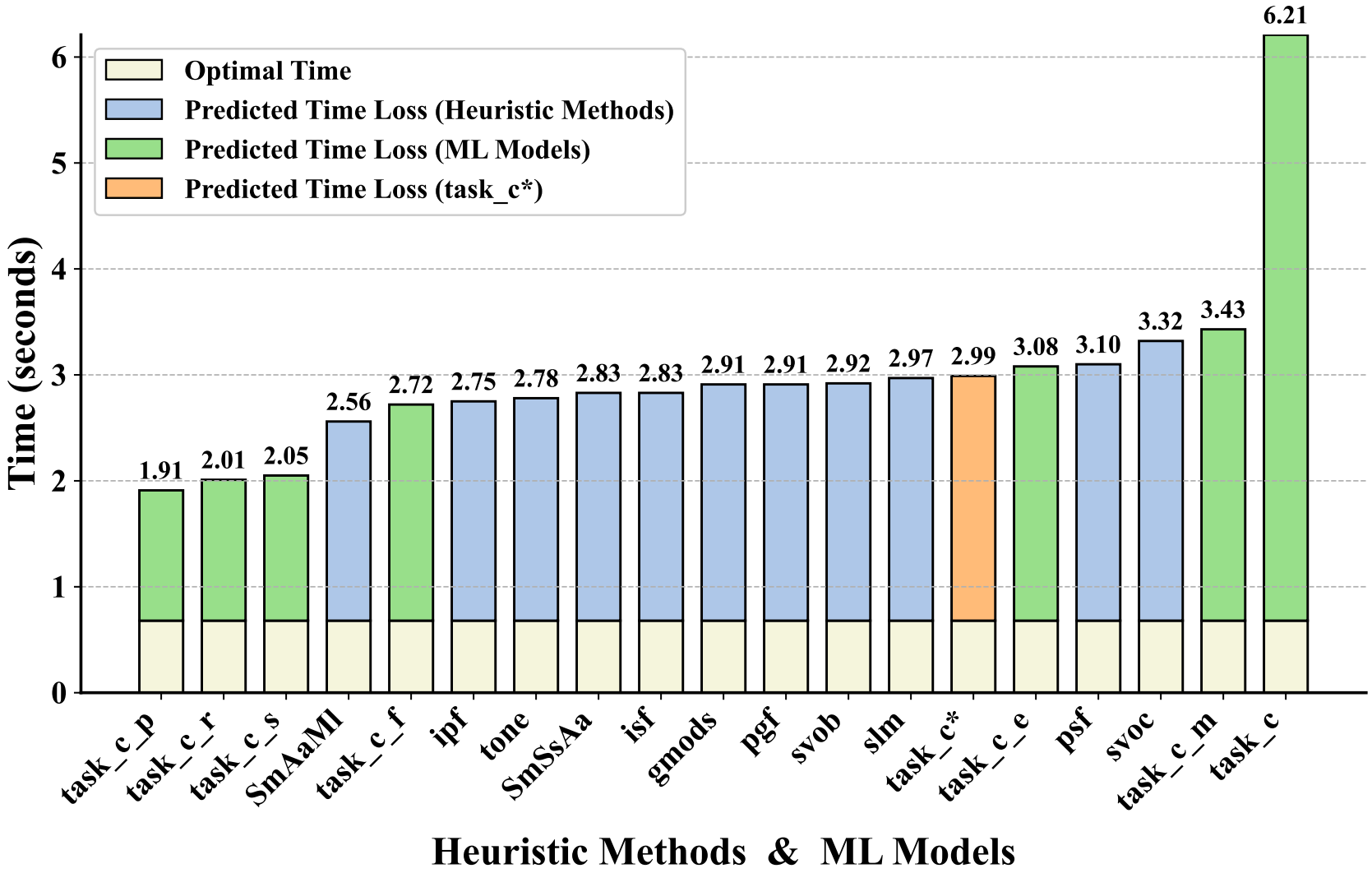
Random DQ-3 (II)



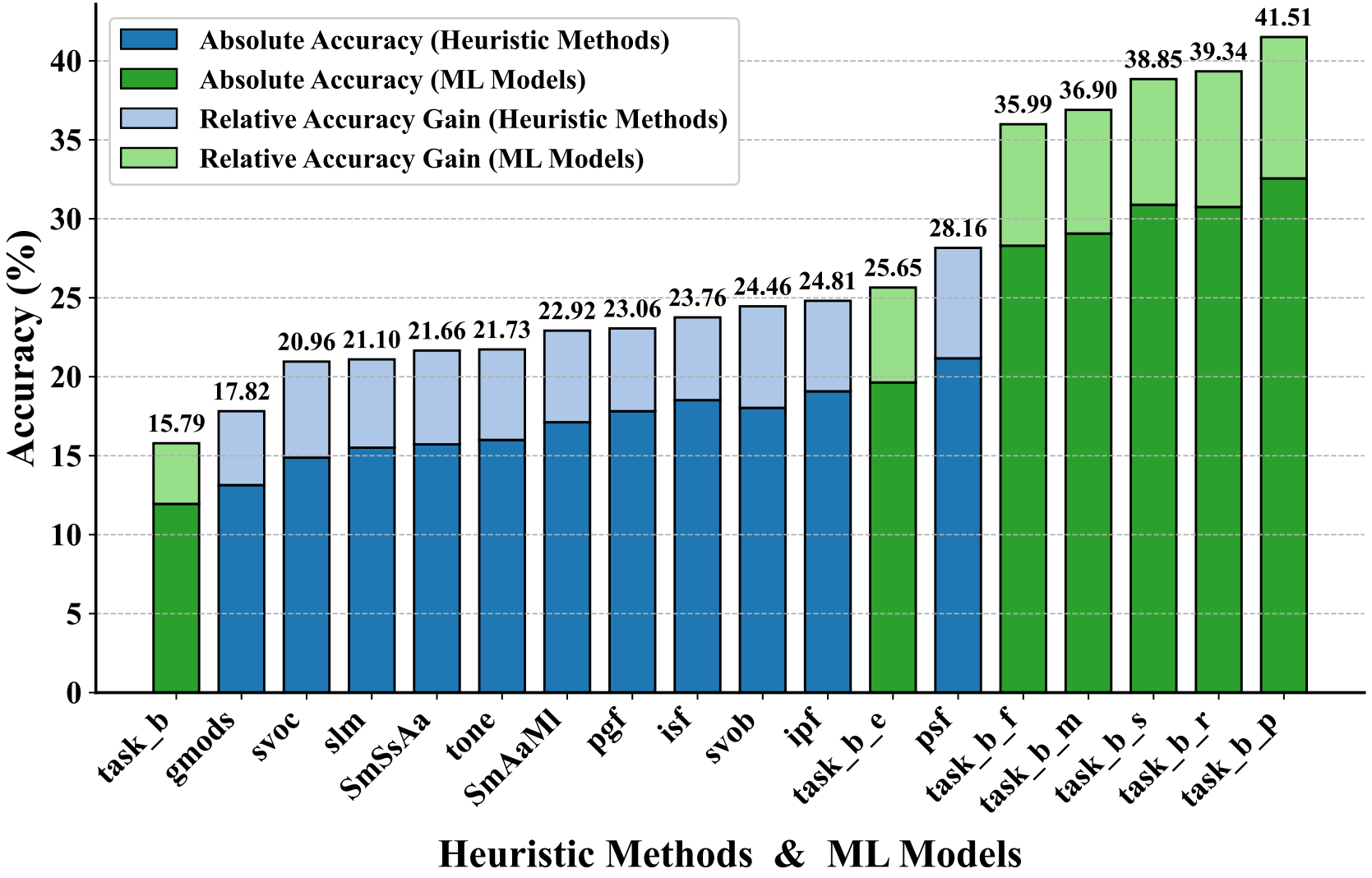
Random DQ-4 (I)


Absolute Accuracy (Heuristic Methods)
Absolute Accuracy (ML Models)
Absolute Accuracy (task_c*)
Relative Accuracy Gain (Heuristic Methods)
Relative Accuracy Gain (ML Models)
Relative Accuracy Gain (task_c*)
Accuracy (%)
0
10
20
30
40
17.88
19.24
22.46
22.71
23.05
23.14
23.81
24.66
24.92
25.85
27.20
27.63
27.88
30.51
35.68
38.64
40.85
43.64
44.07
task_c
gmods
task_c*
slm
SmSsAa
tone
svoc
SmAaMl
pgf
isf
ipf
svob
task_c_e
psf
task_c_f
task_c_m
task_c_s
task_c_p
task_c_r
Heuristic Methods & ML Models

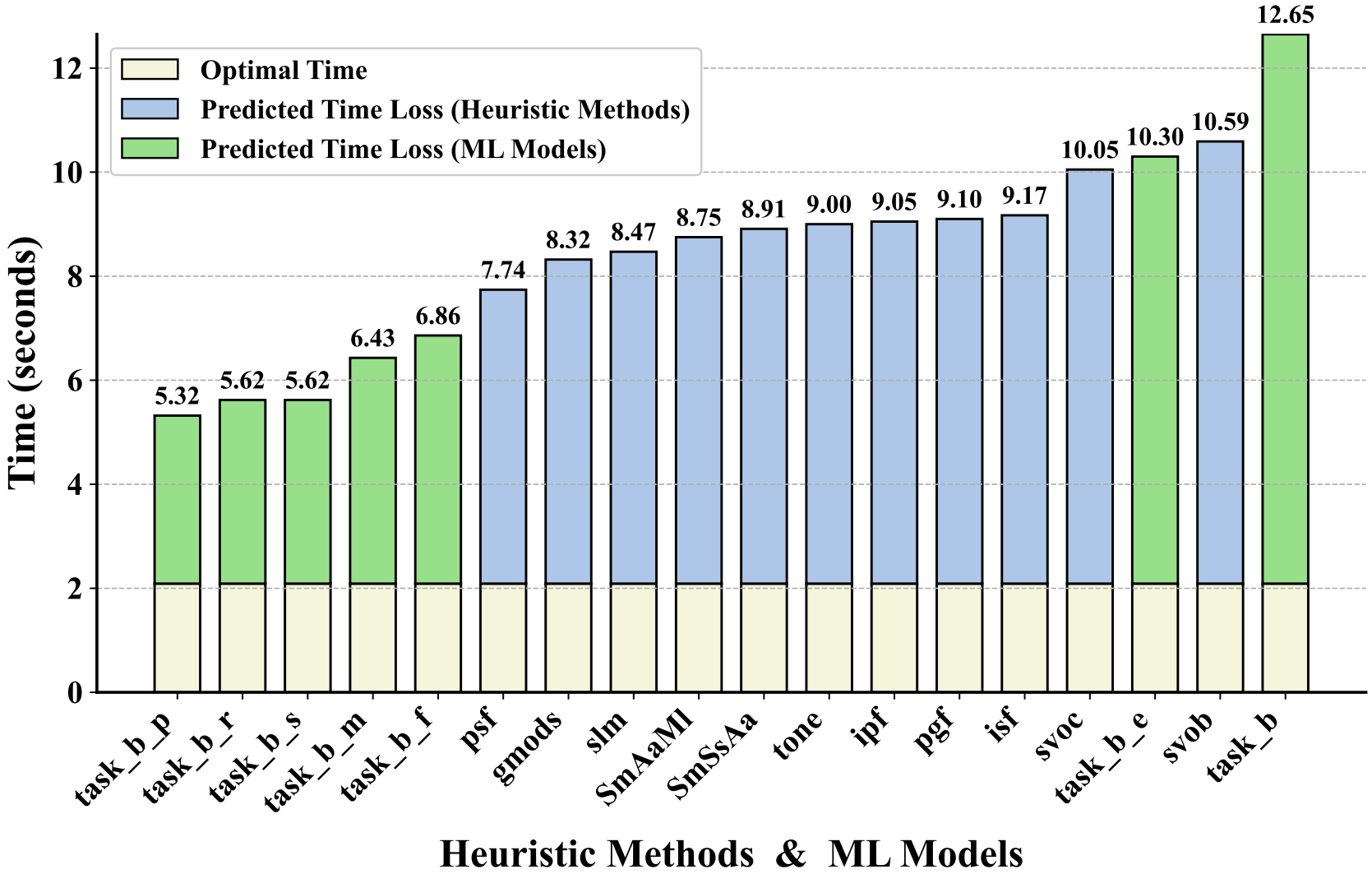
Random DQ-4 (II)



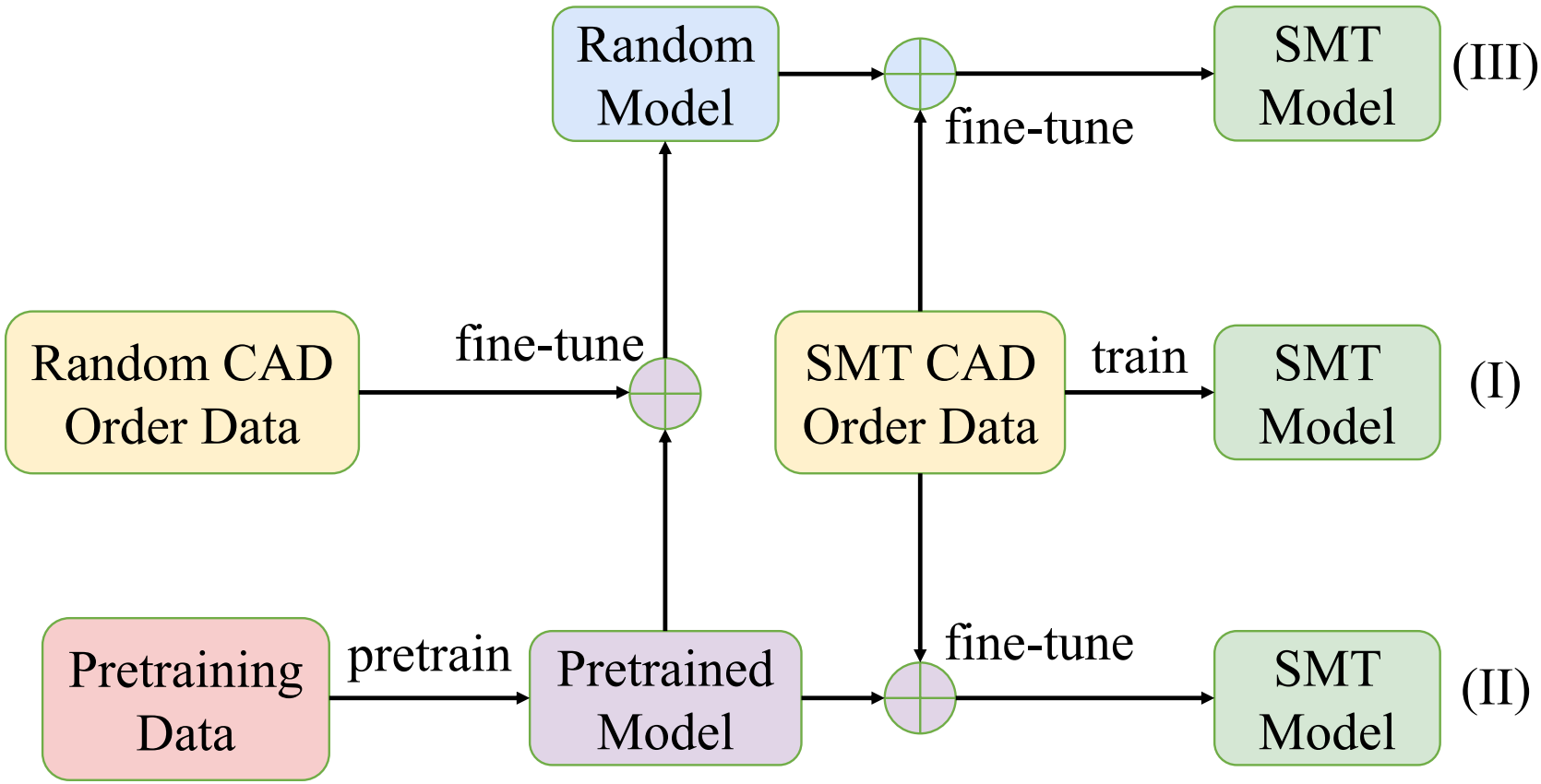
Random DQ-4b (I)



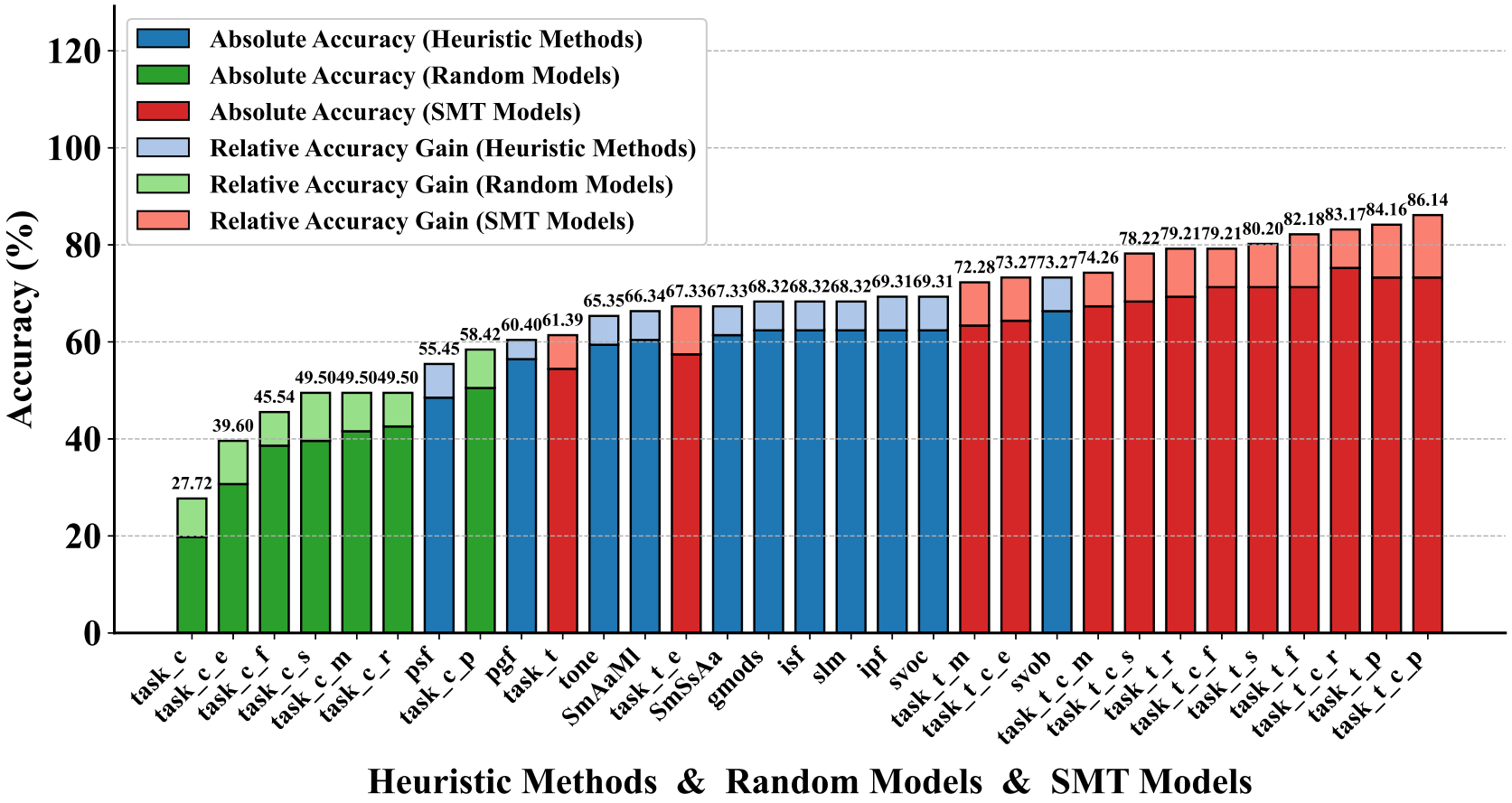
Random DQ-4b (II)



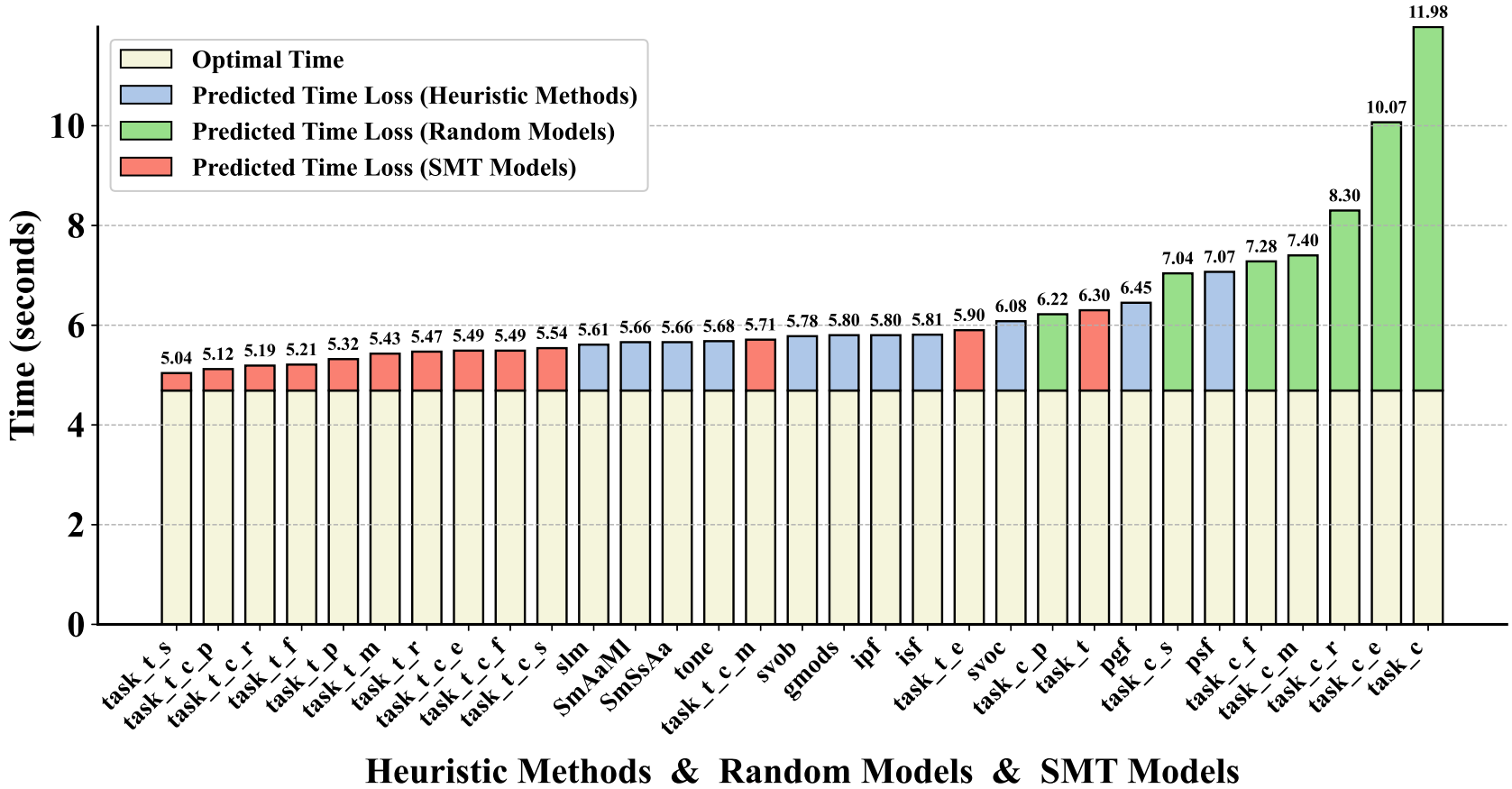
Fine-tuning on SMT Dataset



Accuracies on SMT Dataset



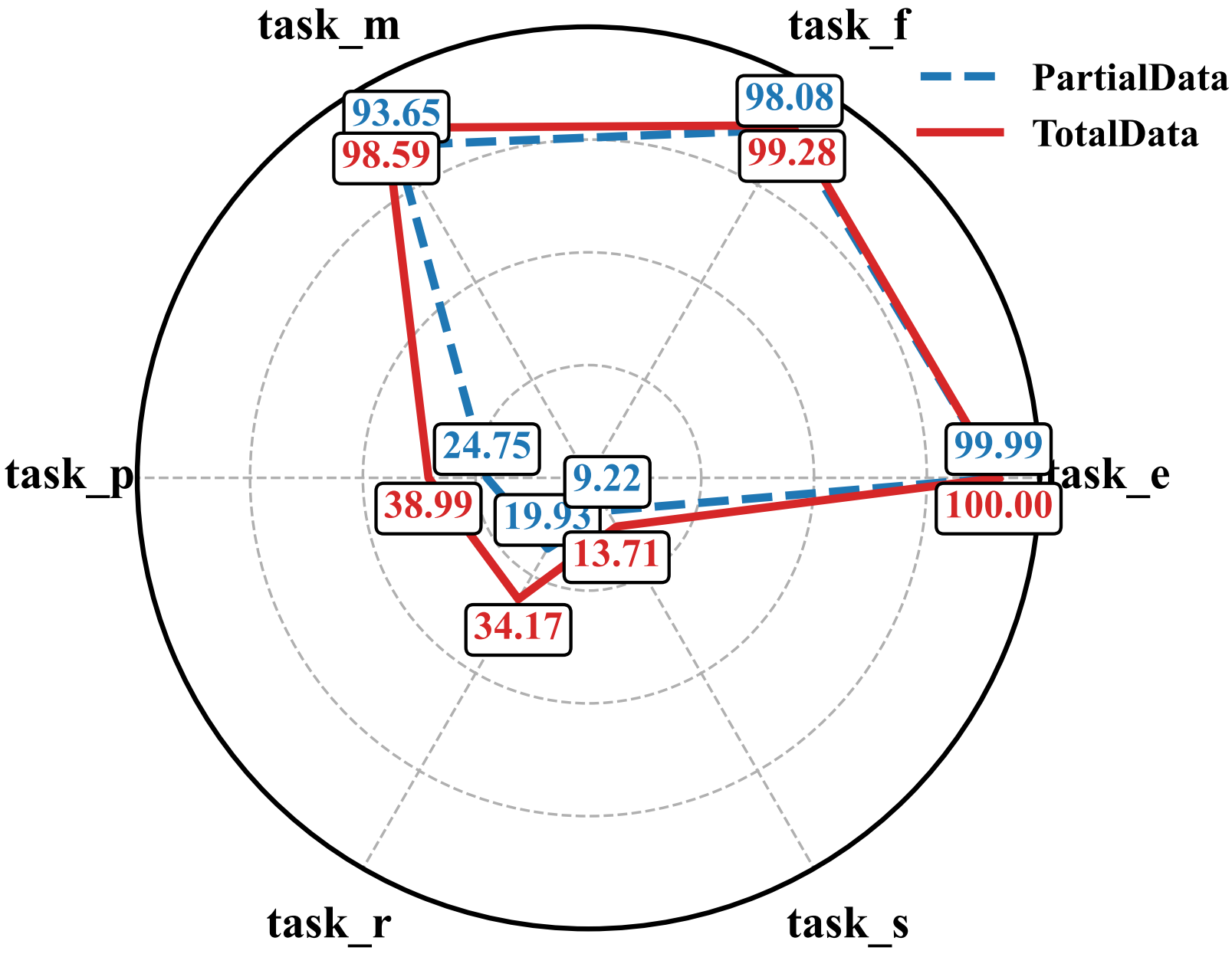
CAD Running Timing on SMT Dataset



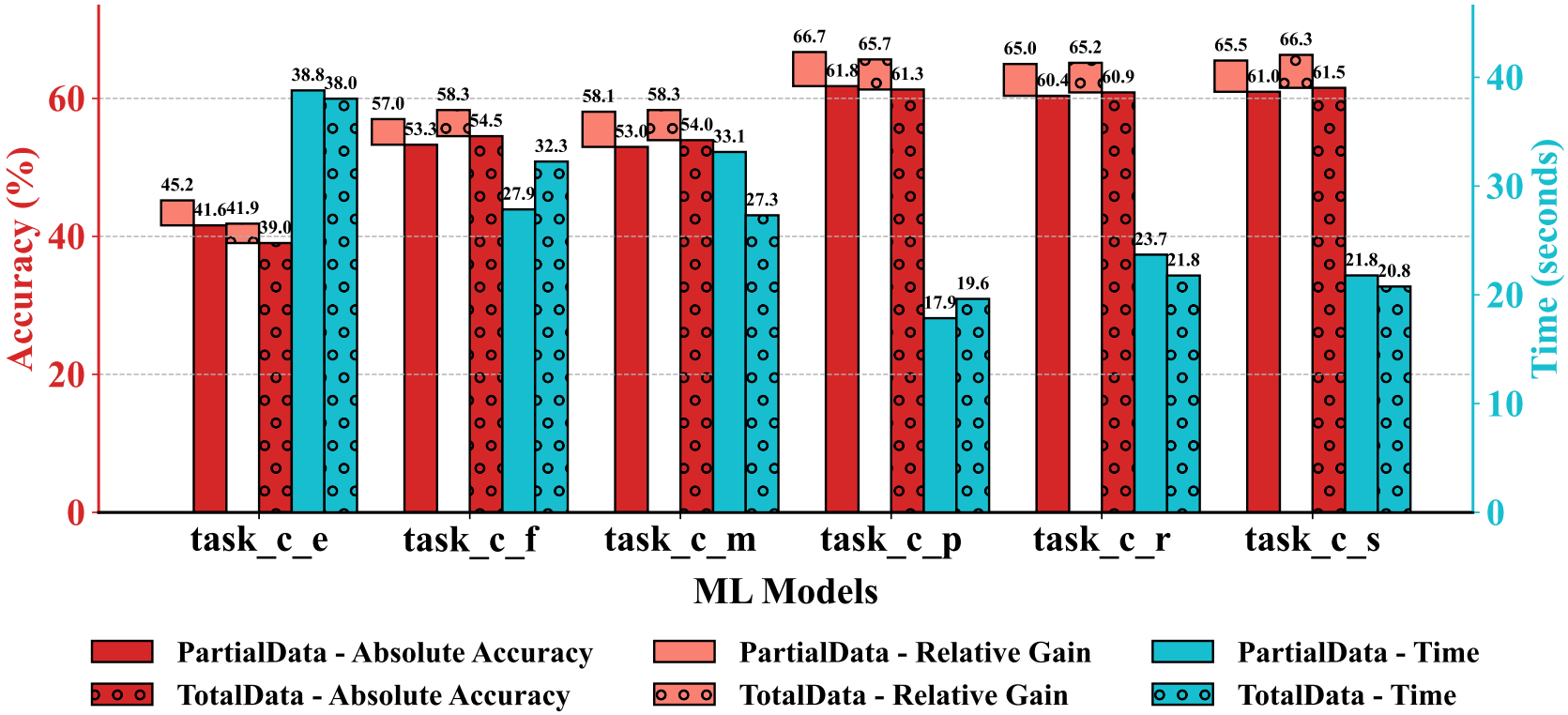
Can we do better?

- (**RQ$_1$**) If we increase the size of the pre-training dataset by another magnitude.
- (**RQ$_2$**) If we increase the capacity of Transformer models.
- (**RQ$_3$**) If we directly use computed features.
- (**RQ$_4$**) If we adopt a multi-stage pre-training scheme.
- (**RQ$_5$**) If we use a different tokenization scheme.

Increase Dataset Size (I)



Increase Dataset Size (II)

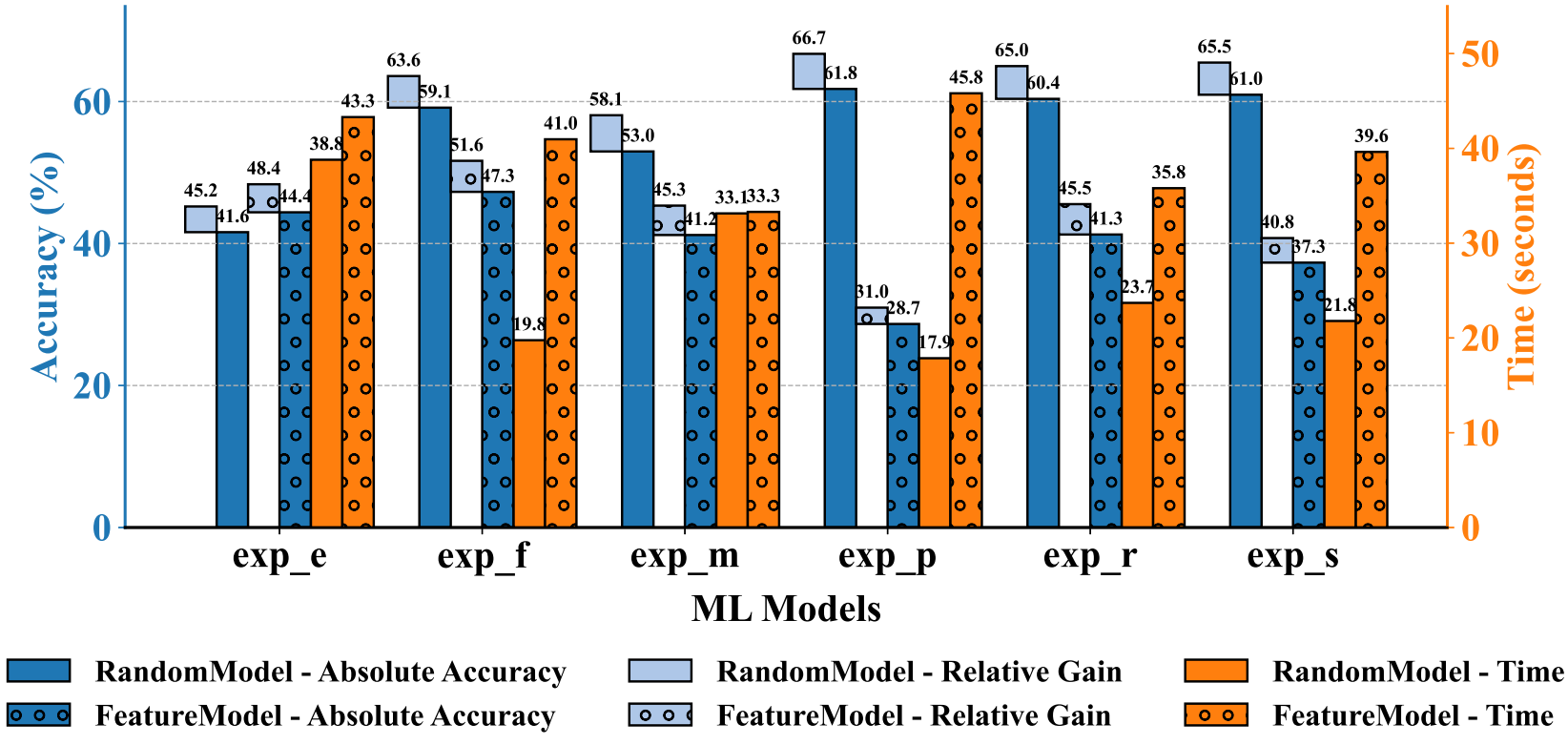


Increase the Capacity

Task_id	Embedding Dimension	Heads	Encoders/Decoders	Batch Size	Epochs
task_s	256	8	7/6	128	100
task_ss	512	16	6/12	32	100

Pretraining_Task	**task_s**	**task_ss**
Accuracy	13.71%	18.57%
CAD_Task	**task_c_s**	**task_c_ss**
Absolute_Accuracy	61.53%	61.29%
Relative_Accuracy	66.31%	65.65%
Predicting_Time	20.77	22.02
Optimal_Time	6.28	6.28

Replace Pre-training by Feature



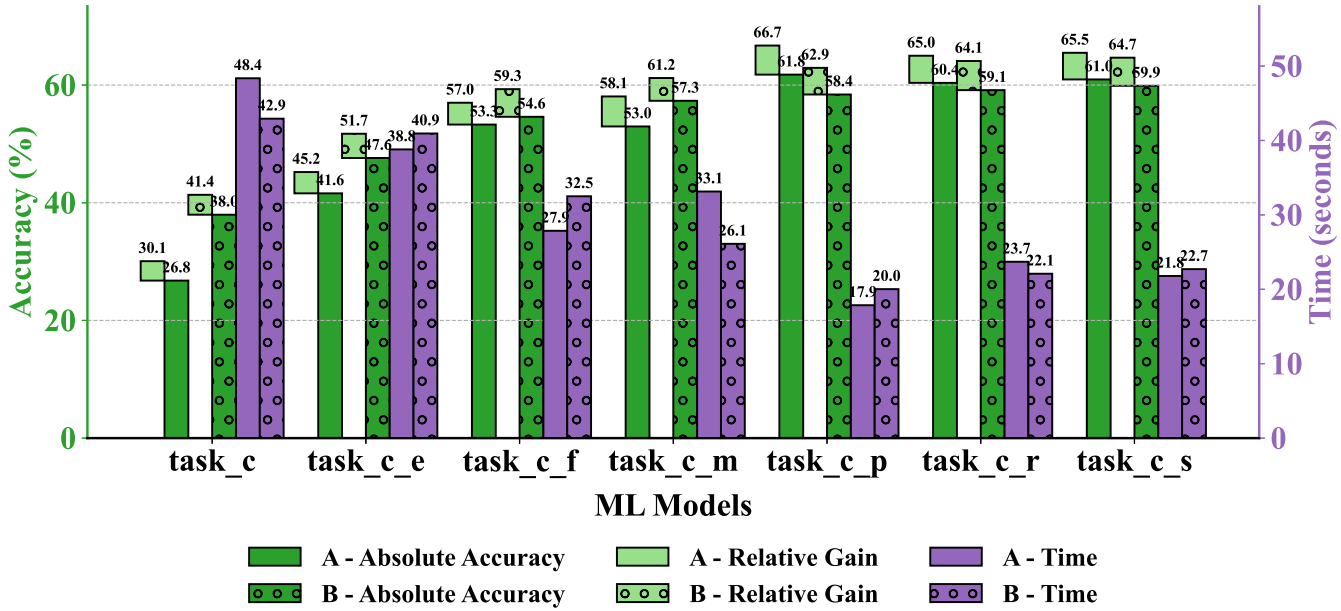
Multi-level Pre-training & Fine-Tuning

Table: Comparison of one-step and multi-step pre-training models.

Pre-training_Task	acc	Order_Task	abs_acc	rel_acc	time_ratio
`task_f`	98.08%	`task_c_f`	53.29%	57.00%	4.43
`task_p`	24.75%	`task_c_p`	61.78%	66.72%	2.84
`task_p_f`	22.48%	`task_c_p_f`	59.14%	63.59%	3.15
`task_fp`	24.77%	`task_c_fp`	58.73%	62.93%	3.87

- `task_c_p_f`: Pre-training with feature_f, fine-tuning with feature_p, fine-tuning again on task_c,
- `task_c_fp`: Pre-training with (feature_f, feature_p), fine-tuning on task_c.

Different Tokenization Scheme



- Method A: completes omitted exponents (0 and 1) in polynomial expressions and distinguishes digits in coefficients and exponents.
- Method B: only distinguishes digits in coefficients and exponents but does not complete missing exponents.

Outline

Cross-dimension Strategy

Suppose that we have a model for n variables, how to handle systems in m variables?

$m = n$ Directly apply the trained models.

$m < n$ Force the model to ignore the rest $n - m$ variables.

$m > n$ $t = \binom{m}{n}$ ways to choose n different variables from m ones.

- For $1 \leq i \leq t$, set the rest $m - n$ variables to 1 and get F_i.
- Call an n-model on (F_i, X_i) to get an ordering O_i.
- Set $\mathrm{score}(x_{i_j}, O_i) = n - j$, $j = 1, \ldots, n$.
- $\mathrm{score}(x_k) = \sum_{i=1}^{t} \mathrm{score}(x_k, O_i)$.
- Sorting x_i by $\mathrm{score}(x_k)$ in descending order.

Blocking ordering

The m variables already satisfy: $X_1 > X_2 > \cdots > X_r$.

$m \leq n$ First choose O and force O to respect the block ordering.

$m > n$ Evaluate F at $X \setminus X_i = 1$ to obtain BF_i.

- Get an ordering O_i for (BF_i, X_i).
- Set $O = O_1 > O_2 > \cdots O_r$.

Experimental Results

Adapted cross-dimension models vs separately trained models.

Dataset	Category	task_b	task_b_e	task_b_f	task_b_m	task_b_p	task_b_r	task_b_s
DQ-4b	abs_acc	11.95%	19.64%	28.30%	29.07%	**32.56%**	30.75%	30.89%
	rel_acc	15.79%	25.65%	35.99%	36.90%	**41.51%**	39.34%	38.85%
	time_ratio	6.05	4.92	3.28	3.07	**2.54**	2.69	2.69
	Category	task_c	task_c_e	task_c_f	task_c_m	task_c_p	task_c_r	task_c_s
	abs_acc	8.60%	15.44%	19.50%	21.80%	23.06%	23.13%	**24.32%**
	rel_acc	12.51%	21.80%	25.79%	28.65%	29.63%	29.84%	**31.94%**
	time_ratio	5.97	4.87	3.50	3.04	2.75	2.73	**2.65**
Dataset	Category	task_c	task_c_e	task_c_f	task_c_m	task_c_p	task_c_r	task_c_s
DQ-3	abs_acc	28.89%	44.44%	52.51%	53.17%	**60.25%**	59.51%	59.67%
	rel_acc	31.28%	47.24%	55.39%	56.21%	**63.95%**	63.62%	63.62%
	time_ratio	9.61	6.44	4.91	4.88	**3.05**	3.56	3.31
	Category	task_b	task_b_e	task_b_f	task_b_m	task_b_p	task_b_r	task_b_s
	abs_acc	20.08%	22.22%	30.04%	42.72%	43.29%	**47.74%**	33.83%
	rel_acc	21.56%	24.28%	32.10%	45.84%	46.83%	**50.86%**	36.79%
	time_ratio	12.52	10.64	8.44	6.29	**5.56**	6.33	7.62

- The best model is still the separately trained.
- 3-variable models perform well on 4-variable datasets.
- 4-variable models perform less satisfactorily on 3-variable datasets but the best one still outperforms the best heuristic.

A Testing Example with 9 Variables

The problem (Sturm and Weispfenning, 1997)

$\exists\{u, v, w\},\ x(f-w) = fu \wedge y(f-w) = fv \wedge (u-a)^2 + (v-b)^2 + (w-c)^2 = 1.$

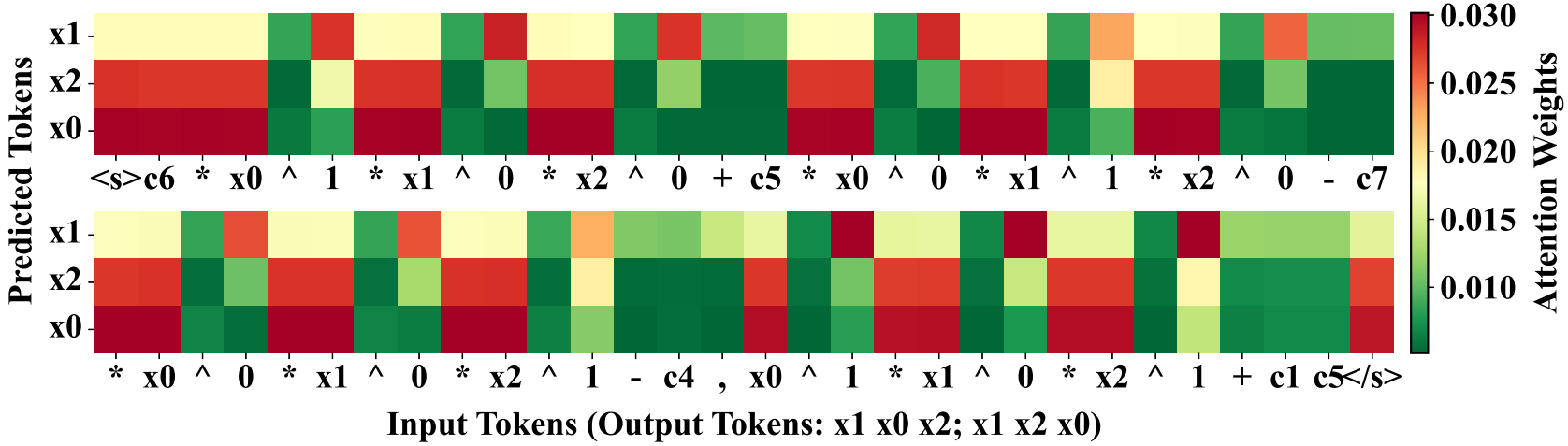
- The bound variables: u, v, w.
- The free variables: x, y, f, a, b, c.
- The command to use: QuantifierElimination (with the option output=rootof) inside the RegularChains library of Maple 2025.

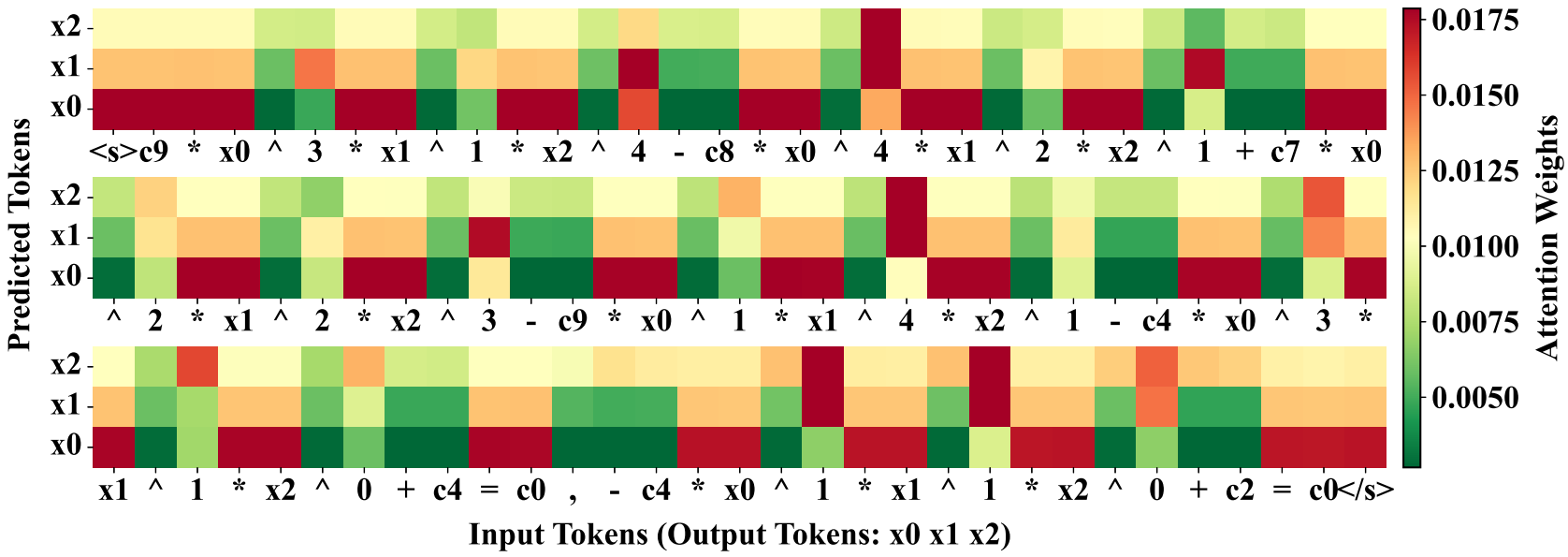
The result:

- Heuristic approaches: $u > v > w > x > y > f > a > b > c$, $u > v > w > x > y > a > b > c > f$, and $w > u > v > x > y > a > b > c > f$.
- None of them can help the solver (timelimit: 100 seconds).
- Model M_p: $v > w > u > b > c > a > f > y > x$ (8.779).
- Model M_r: $w > v > u > a > b > c > x > f > y$ (16.435).
- Model M_s: $v > u > w > c > f > b > a > y > x$ (10.633).

Outline

Cross-attention Weights Analysis





Linear Probing

The main idea

- Let $\mathbf{X} \in \mathbb{R}^{\ell \times d}$ be the output of the frozen encoder.
- Let $\mathbf{h} = \frac{1}{\ell} \sum_{i=1}^{\ell} \mathbf{X}[i, :]$.
- A linear probe: $\hat{\mathbf{y}} = \mathbf{W}\mathbf{h} + \mathbf{b}$ is added after the frozen encoder.
- Training only the linear probe by minimizing the loss $\mathcal{L} = \frac{1}{N} \sum_{i=1}^{N} \|\hat{\mathbf{y}}_i - \mathbf{y}_i\|^2$.

A mini-experiment

- The feature to be regressed: $\mathrm{sum\_max\_d}$.
- Choose $N = 5000$ examples for training.

Example	Metric	probe_p	probe_s
REdMn2rCv3_ex-1490	label output	[2, 1, 2] [1.9770, 1.8595, 1.8570]	[2, 1, 2] [1.6654, 1.6002, 1.7781]
REeMn2tMv3_ex-508	label output	[5, 5, 4] [4.5846, 4.4885, 4.3604]	[5, 5, 4] [4.8229, 4.5072, 4.6669]

Outline

Conclusion and Future Work

- The complex nature of CAD variable ordering selection implies data scarcity.
- Pre-training and fine-tuning can alleviate this problem.
- Deep pre-training tasks perform better than shallow ones.
- Designing the best pre-training task is still an open question.

Deep learning for computing (DL4C)

- Directly learning the output may be difficult.
- Introducing intermediate simple objects to learn.
- Leveraging pre-training and fine-tuning.

Deep computing for learning (DC4L)

- Re-design the algorithm to form blocks.
- The input and output of each block is simple and easy to be verified if possible.
- Leveraging deep learning to accelerate each block.

References I

Alfarano, A., Charton, F., and Hayat, A. (2024). Global Lyapunov functions: a long-standing open problem in mathematics, with symbolic transformers. *Advances in Neural Information Processing Systems*, 37:93643–93670.

Bradford, R. J., Davenport, J. H., England, M., McCallum, S., and Wilson, D. J. (2016). Truth table invariant cylindrical algebraic decomposition. *Journal of Symbolic Computation*, 76:1–35.

Brown, C. (2004). Tutorial: Cylindrical algebraic decomposition, at ISSAC.

Brown, C. W. (2001). Improved projection for cylindrical algebraic decomposition. *Journal of Symbolic Computation*, 32(5):447–465.

Brown, C. W. (2003). QEPCAD B: A program for computing with semi-algebraic sets using CADs. *ACM SIGSAM Bulletin*, 37(4):97–108.

References II

Brown, C. W. and Davenport, J. H. (2007). The complexity of quantifier elimination and cylindrical algebraic decomposition. In *Proceedings of the 2007 International Symposium on Symbolic and Algebraic Computation*, pages 54–60, New York, NY, USA. Association for Computing Machinery.

Brown, C. W. and Kovsta, M. (2015). Constructing a single cell in cylindrical algebraic decomposition. *Journal of Symbolic Computation*, 70:14–48.

Chen, C., Davenport, J. H., Lemaire, F., Moreno Maza, M., Xia, B., Xiao, R., and Xie, Y. (2011). Computing the real solutions of polynomial systems with the RegularChains library in maple. *ACM Communications in Computer Algebra*, 45(3/4):166–168.

References III

Chen, C., Jing, R., Qian, C., Yuan, Y., and Zhao, Y. (2024). A dataset for suggesting variable orderings for cylindrical algebraic decompositions. In *International Workshop on Computer Algebra in Scientific Computing*, volume 14938 of *Lecture Notes in Computer Science*, pages 100–119. Cham: Springer Nature Switzerland.

Chen, C., Moreno Maza, M., Xia, B., and Yang, L. (2009). Computing cylindrical algebraic decomposition via triangular decomposition. In *Proc. of ISSAC*, pages 95–102. ACM.

Chen, C., Zhu, Z., and Chi, H. (2020). Variable ordering selection for cylindrical algebraic decomposition with artificial neural networks. In *Proceedings of International Congress on Mathematical Software*, volume 12097 of *Lecture Notes in Computer Science*, pages 281–291. Springer.

Chen, R. (2025). A geometric approach to cylindrical algebraic decomposition. *Math. Comput.*, 95(360):2025–2059.

References IV

Cho, H., Cha, J., Bhojanapalli, S., and Yun, C. (2025). Arithmetic transformers can length-generalize in both operand length and count. In *Proceedings of the 13th International Conference on Learning Representations*.

Collins, G. E. and Hong, H. (1991). Partial cylindrical algebraic decomposition for quantifier elimination. *Journal of Symbolic Computation*, 12(3):299–328.

del Río Almajan and England, M. (2022). New heuristic to choose a cylindrical algebraic decomposition variable ordering motivated by complexity analysis. In *Proc. of CASC*, volume 13366 of *Lecture Notes in Computer Science*, pages 300–317. Springer.

Dolzmann, A., Seidl, A., and Sturm, T. (2004). Efficient projection orders for CAD. In *Symbolic and Algebraic Computation, International Symposium ISSAC 2004, Santander, Spain, July 4-7, 2004, Proceedings*, pages 111–118. ACM.

References V

England, M. and Florescu, D. (2019). Comparing machine learning models to choose the variable ordering for cylindrical algebraic decomposition. In *Intelligent Computer Mathematics - 12th International Conference, CICM 2019, Prague, Czech Republic, July 8-12, 2019, Proceedings*, volume 11617 of *Lecture Notes in Computer Science*, pages 93–108. Springer.

Florescu, D. and England, M. (2019). Algorithmically generating new algebraic features of polynomial systems for machine learning. In Abbott, J. and Griggio, A., editors, *Proceedings of SC-Square 2019*, CEUR Workshop Proceedings. CEUR-WS.org.

Hong, H. (1990). An improvement of the projection operator in cylindrical algebraic decomposition. In *Proc. of ISSAC*, pages 261–264.

References VI

Huang, Z., England, M., Wilson, D., Davenport, J. H., Paulson, L. C., and Bridge, J. (2014). Applying machine learning to the problem of choosing a heuristic to select the variable ordering for cylindrical algebraic decomposition. In *Intelligent Computer Mathematics*, pages 92–107, Cham. Springer International Publishing.

Iwane, H., Yanami, H., Anai, H., and Yokoyama, K. (2013). An effective implementation of symbolic–numeric cylindrical algebraic decomposition for quantifier elimination. *Theoretical Computer Science*, 479:43–69. Symbolic-Numerical Algorithms.

Jia, F., Dong, Y., Liu, M., Huang, P., Ma, F., and Zhang, J. (2023). Suggesting variable order for cylindrical algebraic decomposition via reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pages 76098–76119. Curran Associates, Inc.

References VII

Jing, R., Qian, C., and Chen, C. (2024). Variable ordering selection for cylindrical algebraic decomposition via reinforcement learning. *Journal of System Science and Mathematical Science Chinese Series*, 0.

Jovanovic, D. and de Moura, L. (2012). Solving non-linear arithmetic. In *Automated Reasoning*, pages 339–354, Berlin, Heidelberg. Springer Berlin Heidelberg.

Kera, H., Ishihara, Y., Kambe, Y., Vaccon, T., and Yokoyama, K. (2024). Learning to compute Gröbner bases. *Advances in Neural Information Processing Systems*, 37:33141–33187.

Lample, G. and Charton, F. (2020). Deep learning for symbolic mathematics. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

References VIII

Lazard, D. (1994). An improved projection for cylindrical algebraic decomposition. In *Algebraic geometry and its applications: collections of papers from Shreeram S. Abhyankar's 60th birthday conference*, pages 467–476. Springer.

Li, H., Xia, B., Zhang, H., and Zheng, T. (2023a). Choosing better variable orderings for cylindrical algebraic decomposition via exploiting chordal structure. *Journal of Symbolic Computation*, 116:324–344.

Li, H., Xia, B., and Zhao, T. (2023b). Local search for solving satisfiability of polynomial formulas. In *Computer Aided Verification - 35th International Conference, CAV 2023, Paris, France, July 17-22, 2023, Proceedings, Part II*, volume 13965 of *Lecture Notes in Computer Science*, pages 87–109. Springer.

References IX

Lu, Z., Siemer, S., Jha, P., Day, J., Manea, F., and Ganesh, V. (2024). Layered and staged Monte Carlo tree search for SMT strategy synthesis. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence*, IJCAI '24, pages 1907–1915.

Malhou, M., Perret, L., and Lauter, K. (2026). HATSolver: Learning Gröbner bases with hierarchical attention transformers. In *Proceedings of the 14th International Conference on Learning Representations*.

McCallum, S. (1998). An improved projection operation for cylindrical algebraic decomposition. In *Quantifier Elimination and Cylindrical Algebraic Decomposition*, pages 242–268. Springer.

McCallum, S., Parusiński, A., and Paunescu, L. (2019). Validity proof of lazard's method for CAD construction. *Journal of Symbolic Computation*, 92:52–69.

References X

Nalbach, J., Ábrahám, E., Specht, P., Brown, C. W., Davenport, J. H., and England, M. (2024). Levelwise construction of a single cylindrical algebraic cell. *Journal of Symbolic Computation*, 123:102288.

Pickering, L., del Río Almajan, T., England, M., and Cohen, K. (2024a). Explainable AI insights for symbolic computation: A case study on selecting the variable ordering for cylindrical algebraic decomposition. *Journal of Symbolic Computation*, 123:102276.

Pickering, L., del Río Almajano, T., England, M., and Cohen, K. (2024b). Explainable ai insights for symbolic computation: A case study on selecting the variable ordering for cylindrical algebraic decomposition. *Journal of Symbolic Computation*, 123:102276.

References XI

Seidl, A. and Sturm, T. (2003). A generic projection operator for partial cylindrical algebraic decomposition. In *Proceedings of the 2003 international symposium on Symbolic and algebraic computation*, pages 240–247.

Strzebonski, A. W. (2000). Solving systems of strict polynomial inequalities. *Journal of Symbolic Computation*, 29(3):471–480.

Strzebonski, A. W. (2006). Cylindrical algebraic decomposition using validated numerics. *Journal of Symbolic Computation*, 41(9):1021–1038.

Sturm, T. and Weispfenning, V. (1997). Computational geometry problems in REDLOG. In Wang, D., editor, *Automated Deduction in Geometry*, pages 58–86. Springer Berlin Heidelberg.

Zhu, Z. and Chen, C. (2020). Variable order selection for cylindrical algebraic decomposition based on machine learning. *Journal of System Science and Mathematical Sciences (Chinese)*, 40(8):1492–1506.