Trainable Transformer in Transformer

Abhishek Panigrahi* Sadhika Malladi* Mengzhou Xia Sanjeev Arora
Department of Computer Science, Princeton University * Equal Contribution
{ap34, smalladi, mengzhou, arora}@cs.princeton.edu

ICML 40 Years International Conference On Machine Learning

PLI PRINCETON Language + Intelligence

Abstract

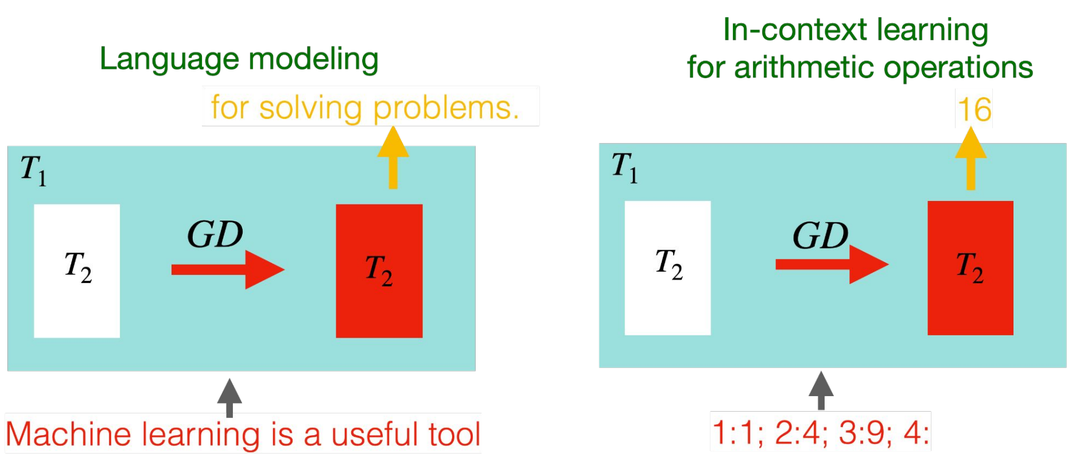
In-context Learning: Given a few task examples in context, a model learns on the fly to perform the task and complete a test query.

Hypothesis: LLMs simulate a few steps of GD on a small, simple model (e.g., logistic classifier) during inference [1, 2, 3]

Can a transformer simulate and train an internal transformer?

How many parameters to simulate+train a 125M transformer?

We construct a transformer with < 2B parameters that can simulate training a OPT-125M model during inference. We call this TinT.



- Extensible to many transformer layers: GLUs, RMSnorm, relative attention, etc.
- Possibility to encode intricate algorithms in large transformers.
- Insights into how to build transformers with explicit inductive biases.

Architecture Overview

Simulating GD with TinT: first few layers simulate the **forward propagation**, next few layers simulate **backpropagation** and **descent**, final layers simulate **evaluation** with the updated model.

Two necessities

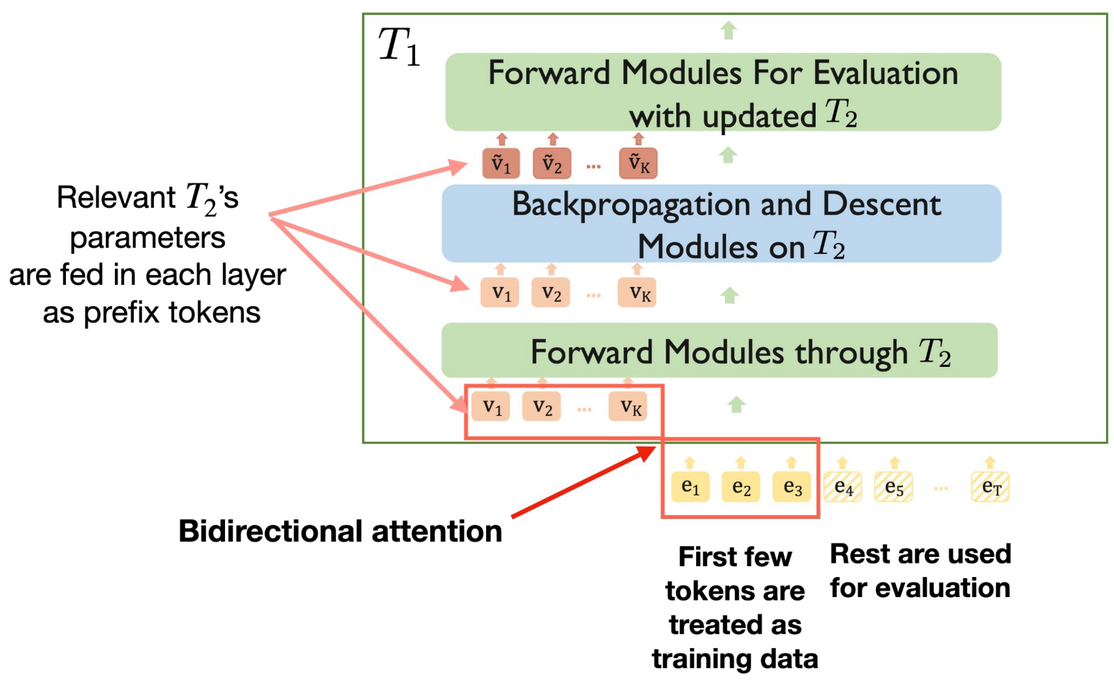
A. Read and write access to T_2

B. T_2's read access to in-context data

Strategy

A. Feed in T_2's parameters as input prefix tokens to each layer. Similar to Prefix tuning of LLMs [4]

B. Bidirectional attention on in-context and prefix tokens. Similar to Prefix LLMs [5, 6].



Internal Details

- 4 major operations: linear, activation, self-attention, layer-norm.
- Create 3 **simulator modules** for each operation (12 total) to simulate forward, backward, and descent operations.
- Stack the simulator modules according to the auxiliary model.

Example: Simulating forward operation on linear layer

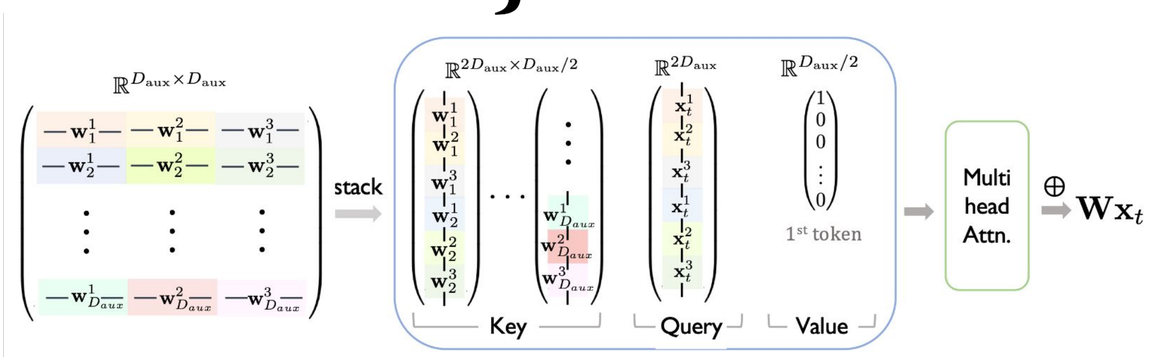
$$\mathbf{x}_1, \cdots, \mathbf{x}_T \to \mathbf{y}_1, \cdots, \mathbf{y}_T \qquad \mathbf{y}_t = \mathbf{W}\mathbf{x}_t$$

$$\begin{pmatrix} -\mathbf{w}_1- \\ -\mathbf{w}_2- \\ \vdots \\ -\mathbf{w}_{D_{aux}}- \end{pmatrix} \begin{pmatrix} | \\ \mathbf{x}_t \\ | \end{pmatrix} = \begin{pmatrix} \langle \mathbf{w}_1, \mathbf{x}_t \rangle \\ \langle \mathbf{w}_2, \mathbf{x}_t \rangle \\ \vdots \\ \langle \mathbf{w}_{D_{aux}}, \mathbf{x}_t \rangle \end{pmatrix}$$

Attention scores between $\mathbf{w}_i$ and $\mathbf{x}_t$

Issues with a blind implementation
- Sequence Length blowup
- Inefficient use of attention heads

→ Strategies
- Stacking multiple params to prefix tokens
- Computation split across attention heads

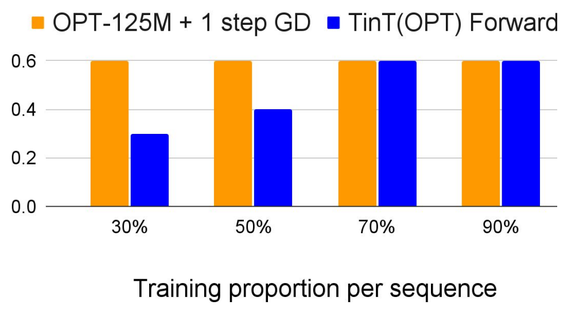


Other modifications for parameter efficiency:

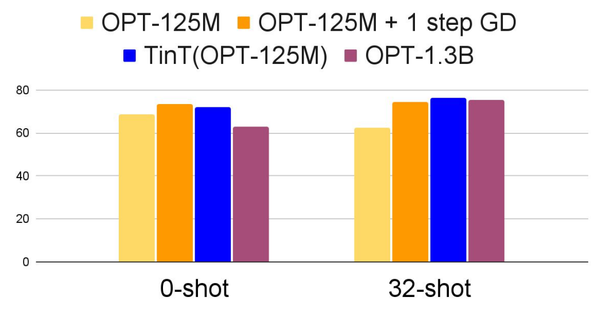
- Duality of self-attention for efficient descent through linear layer
- Zero-order gradient approximations
- Approximations via stop-gradient
- Parallelizing computation by Sub-MLP partition in MLPs
- Low-rank linear layers in TinT

Empirical Validations

Drop in Perplexity w.r.t. OPT-125M



Average Downstream Performance



Language Modeling

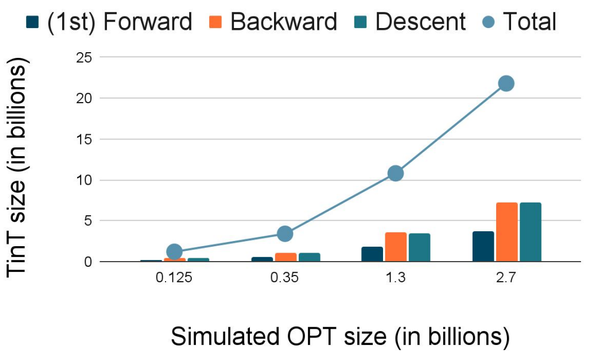
- 0.4-0.6 better perplexity over OPT-125M.
- TinT performs close to true one-step GD.

In-Context Learning

- Achieves >4% better accuracy over OPT-125M.
- TinT performs close to true one-step GD.
- Competitive with comparatively sized OPT.

TinT's size grows as internal model grows

- Backward and descent operations require ~2x more parameters than forward operations.



References

1. Ekin Akyurek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. What learning algorithm is in-context learning? investigations with linear models. arXiv preprint arXiv:2211.15661, 2022.
2. Angeliki Giannou, Shashank Rajput, Jy yong Sohn, Kangwook Lee, Jason D. Lee, and Dimitris Papailiopoulos. Looped transformers as programmable computers, 2023.
3. Johannes von Oswald, Eyvind Niklasson, Ettore Randazzo, Joao Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. Transformers learn in-context by gradient descent, 2022.
4. Xiang Lisa Li, Xiang Lisa Li. Prefix-Tuning: Optimizing Continuous Prompts for Generation. 2021
5. Peter J Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. Generating wikipedia by summarizing long sequences. arXiv preprint arXiv:1801.10198, 2018.
6. Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. The Journal of Machine Learning Research, 21(1):5485–5551, 2020.