Local Mixtures of Experts: Essentially Free Test-Time Training via Model Merging

ETH zürich

Ryo Bertolissi*, Jonas Hübotter*, Ido Hakimi, Andreas Krause

Learning & Adaptive Systems

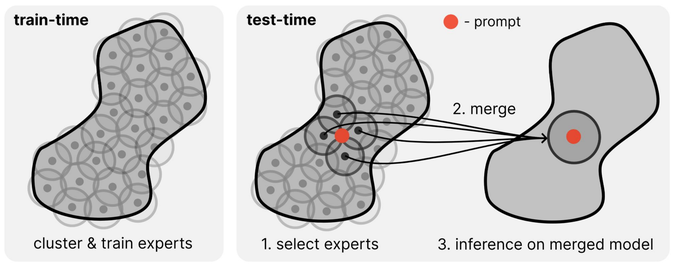
Background

- **Goal:** Use a specialized model, tailored to each prompt.
- **Problem:** Test-time training is expensive.

Can we realize the gains of TTT without almost any of its inference overhead?

Contributions

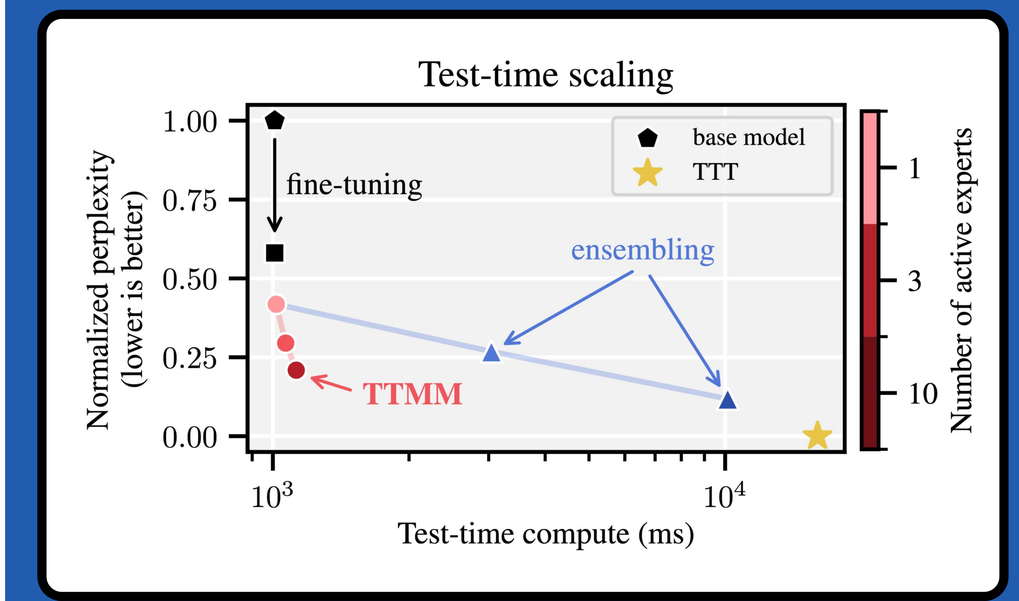
- We propose **TTMM**, which combines MoEs with model merging to approx. TTT.
- At **train-time**, TTMM clusters the training data and trains separate experts.
- At **test-time**, TTMM selects best experts via a *sparse cross-attention router* and merges experts in a *single local expert*.



- We show that TTMM significantly improves language modeling without almost any increase in inference cost.

Test-time training improves LLMs, but is expensive.

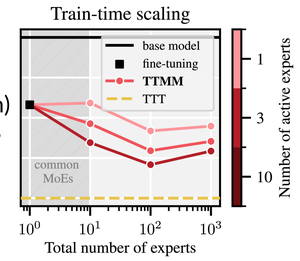
Test-Time Model Merging (TTMM) approximates test-time training w/o increased inference cost by merging experts at test-time.



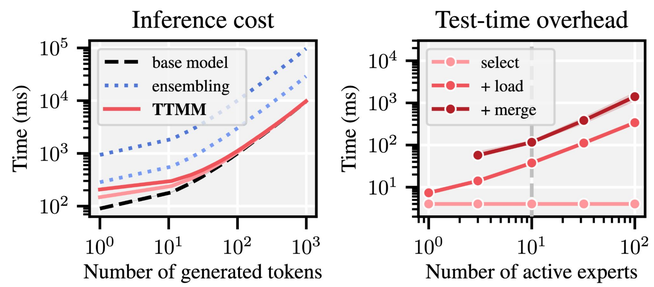
Details

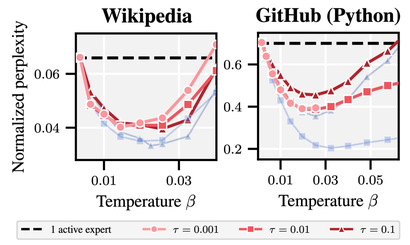
TTMM approximates TTT in 3 steps:
1. Fewer amortized experts
2. Summarizing experts via their centroids
3. Merging multiple experts

Details:
Corpora: Wikipedia, Github (Python)
Models: Llama-3.2-1B, Qwen2.5-1.5B
#Experts: 10, 100 **(best)**, 1000



Inference cost:
TTMM with 10 active experts has *constant* overhead of 115ms, a **125x speedup** over the 15s overhead of TTT. (with Llama-3.2-1B)





Is merging helpful?
Varying the temp of the cross-attention, we see the tradeoff for *locality*:
- Too narrow: single expert
- Too diverse: no benefit

Algorithm 2 TTMM at test-time

Require: Prompt $x^\star$ with embedding $\phi^\star$. Language model $f(x;\theta)$ with pre-trained weights θ; expert models $\{(\theta_k, \phi_k)\}_{k=1}^K$; temperature β and sparsity τ (tuned on holdout data).

1: Compute cluster-specific merging coefficients:
$$w_k \leftarrow \text{sparse-softmax}_\tau\left(\tfrac{1}{\beta}\phi_k^\top \phi^\star\right)$$
2: Merge into a single expert model:
$$\theta^\star \leftarrow \textstyle\sum_{k=1}^K w_k \theta_k$$
3: Use $f(x^\star;\theta^\star)$ to generate the next token.