

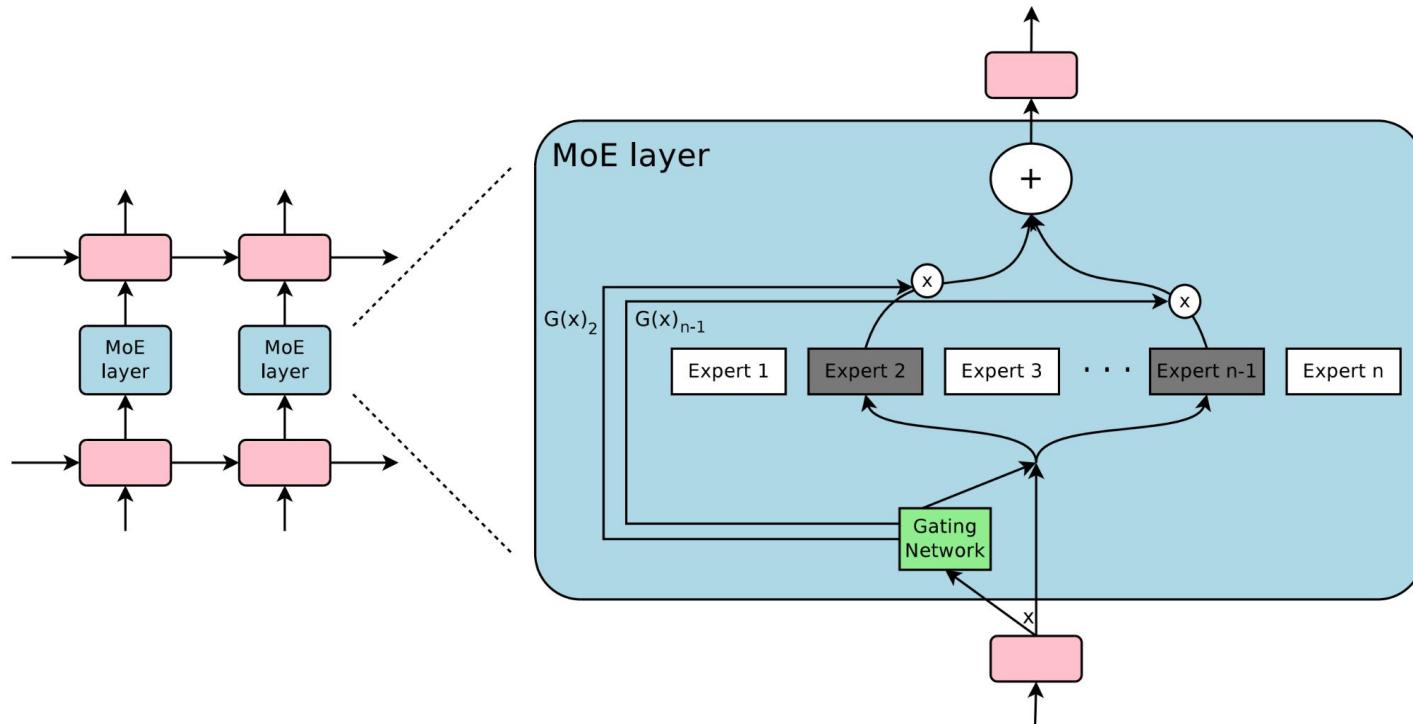


Sparse/Retrieval-based LM

Lihe Yang, Meng Wei

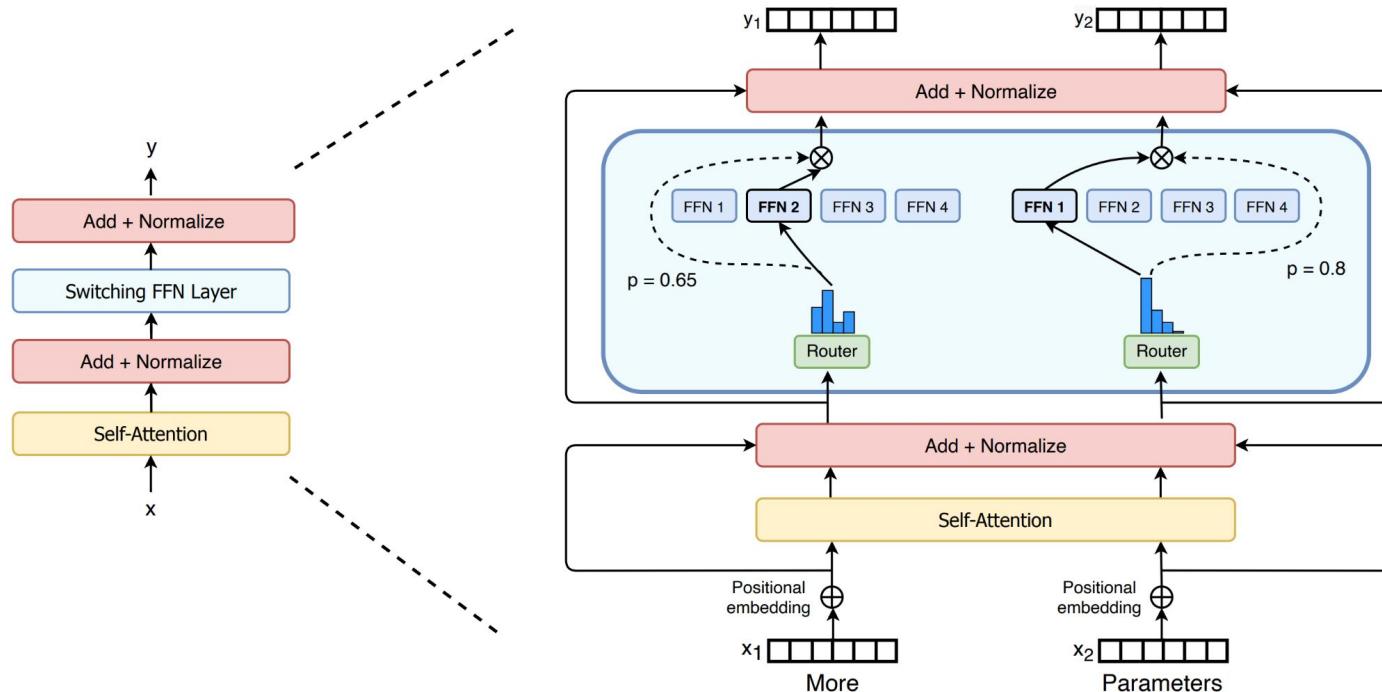
Sparse Language Models

- Mixture-of-Experts (MoE)



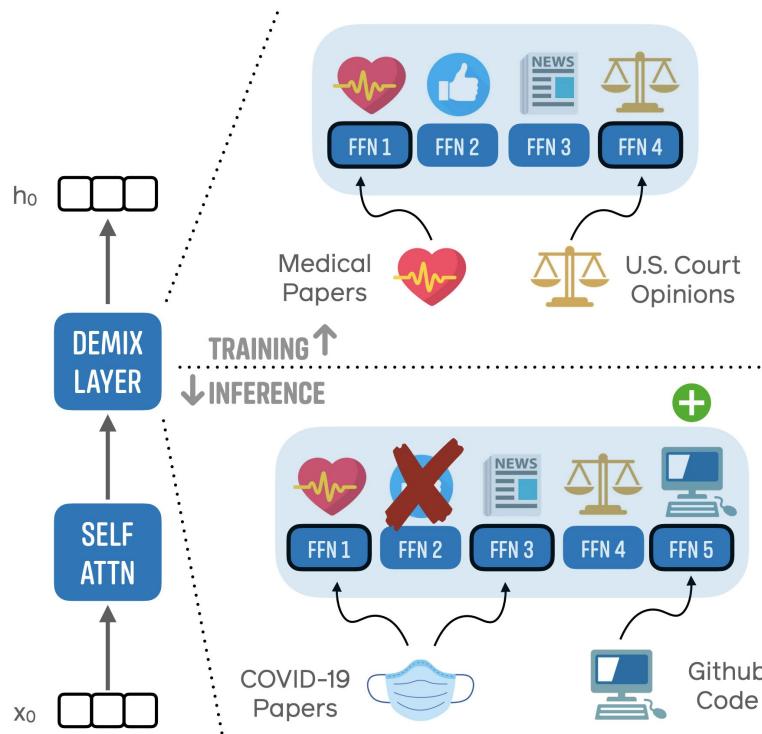
Sparse Language Models

- Mixture-of-Experts (MoE)

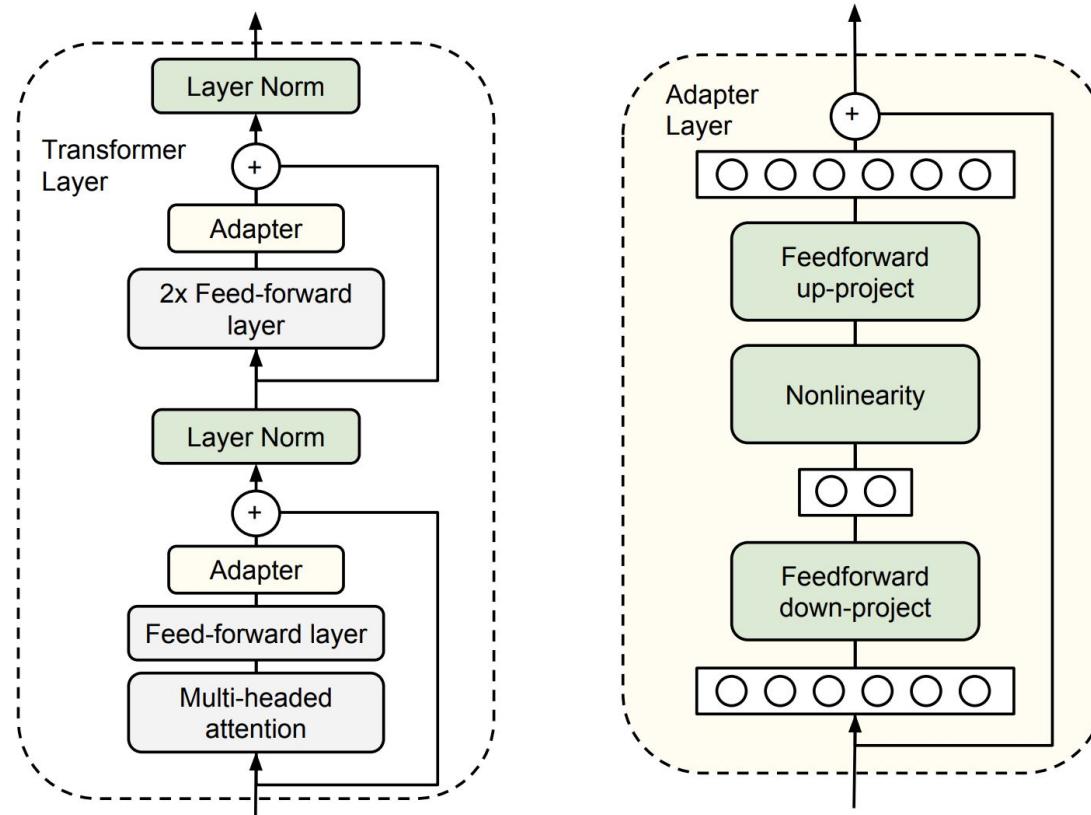


Sparse Language Models

- DEMix: specializing feedforward layers on specific domains



Adapters



Ensemble & Parameter Averaging

- Ensemble: simple uniform averaging, bagging, boosting, stacking, etc
- Parameter Averaging: uniform weight, relevance-based weight, exponential moving average (EMA), etc

Branch-Train-Merge: Embarrassingly Parallel Training of Expert Language Models

NeurIPS 2022 Workshop

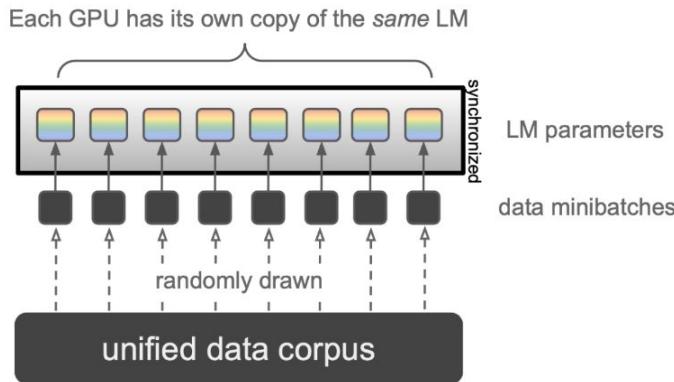
Motivations

- Training large language models (LLM) requires massive multi-node synchronization, which is time-consuming and not robust to node failure.
- It is not easy for LLM to forget specific knowledge.
- Adding new knowledge to LLM is not easy.

Embarrassingly Simple Solution

(a) Fully Synchronized Training

Train one LM on mono-corpus, synchronizing weights across all GPUs

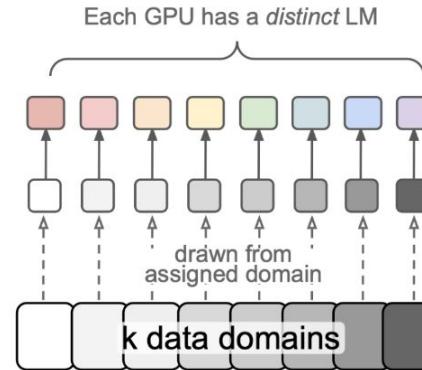


Key idea:

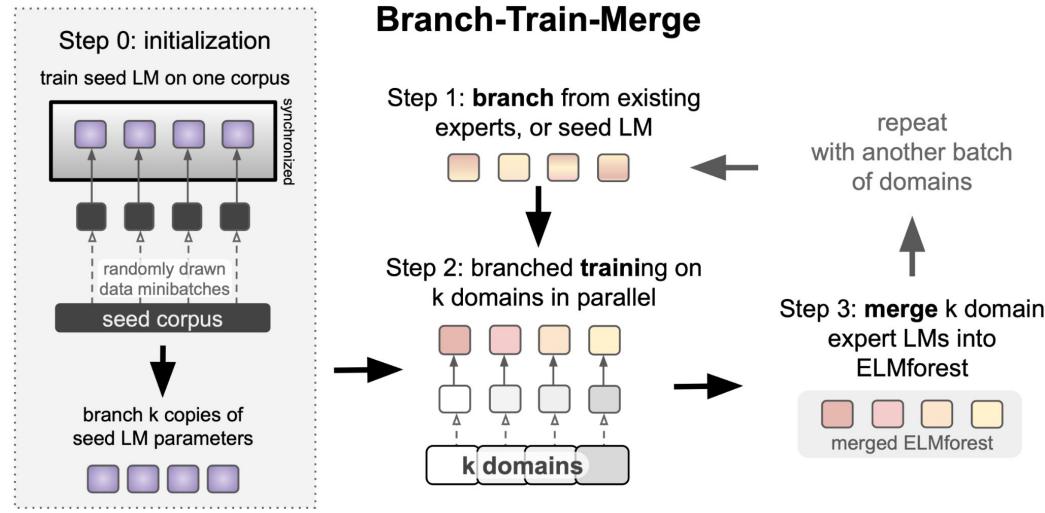
one independent LM for one domain: no parameter sharing, no synchronization

(b) Embarrassingly Parallel Training

Train k independent LMs in parallel on one data domain each, without synchronizing across LMs



Training Scheme



1. Seed training

- Train an initial LM on the combination of all data

2. Domain-specific training for Expert LM (ELM)

- Branch from seed LM: Initialize K LMs with the seed LM for K data domains, then independently train each LM on its domain-specific data with at most one/two GPU nodes

3. New domains

- Branch (initialize) from existing experts by weighted parameter averaging

Test Scheme—Prediction Ensemble & Parameter Averaging

- **Prediction ensemble**

- Use estimated domain posteriors as a weighting factor

$$p(X_t \mid \boldsymbol{x}_{<t}) = \sum_{j=1}^n p(X_t \mid \boldsymbol{x}_{<t}, D = j) \cdot p(D = j \mid \boldsymbol{x}_{<t})$$

- Pro: promising performance
 - Cons: large-scale model parameters, heavy inference cost

- **Parameter averaging**

- Uniform averaging
 - Use estimated domain posteriors for weighted averaging
 - Pro: only require a single model for inference

Findings and Analysis

- Much better results than standard Transformer-LM (T-LM) and DEMix

125M			
	T-LM	DEMIX	ELMFOREST
	125M	512M	1B
Train	19.9 _{0.23}	18.2 _{0.82}	17.2 _{0.02}
Eval	25.2 _{0.18}	23.4 _{0.54}	22.4 _{0.12}
All	22.5 _{0.14}	20.8 _{0.63}	19.8 _{0.05}

750M			
	T-LM	DEMIX	ELMFOREST
	750M	3.8B	6B
Train	14.7	13.5	13.4
Eval	19.3	17.7	16.7
All	17.0	15.6	15.0

350M			
	T-LM	DEMIX	ELMFOREST
	350M	1.8B	2.8B
Train	16.3	15.0	14.7
Eval	20.8	19.9	18.6
All	18.5	17.5	16.7

1.3B			
	T-LM	DEMIX	ELMFOREST
	1.3B	7B	10.4B
Train	14.2	13.7	13.0
Eval	18.4	17.6	16.3
All	16.3	15.6	14.6

Although the training budget is smaller, but ELMForest requires much more inference cost

Findings and Analysis

- Parameter-averaging can also outperform standard Transformer-LM

	Train Domains PPL (↓)			
	125M	350M	760M	1.3B
TRANSFORMER-LM	19.9	16.3	14.7	14.2
ELMFOREST parameter average (uniform weights)	47.4	19.9	19.0	18.0
Argmax ELM (one-hot posterior)	18.0	15.3	14.1	13.8
ELMFOREST parameter average (posterior weights)	18.0	15.1	13.9	13.4
ELMFOREST ensemble	17.2	14.7	13.4	13.0
	Eval Domains PPL (↓)			
	125M	350M	760M	1.3B
TRANSFORMER-LM	25.2	20.8	19.3	18.4
ELMFOREST parameter average (uniform weights)	31.0	22.4	20.8	19.5
Argmax ELM (one-hot posterior)	28.3	22.3	22.3	20.3
ELMFOREST parameter average (posterior weights)	28.5	20.3	18.0	17.0
ELMFOREST ensemble	22.4	18.6	16.7	16.3

Better performance than T-LM, and same-size parameters, same inference speed as T-LM

Findings and Analysis

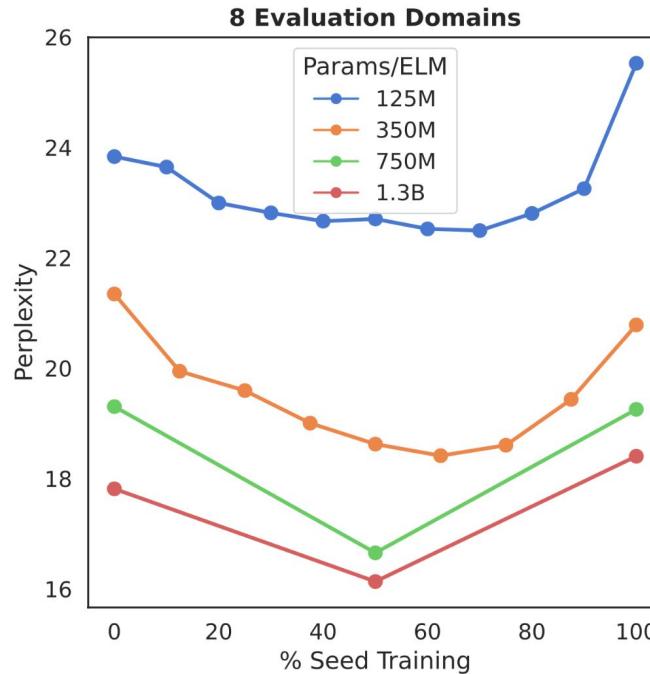
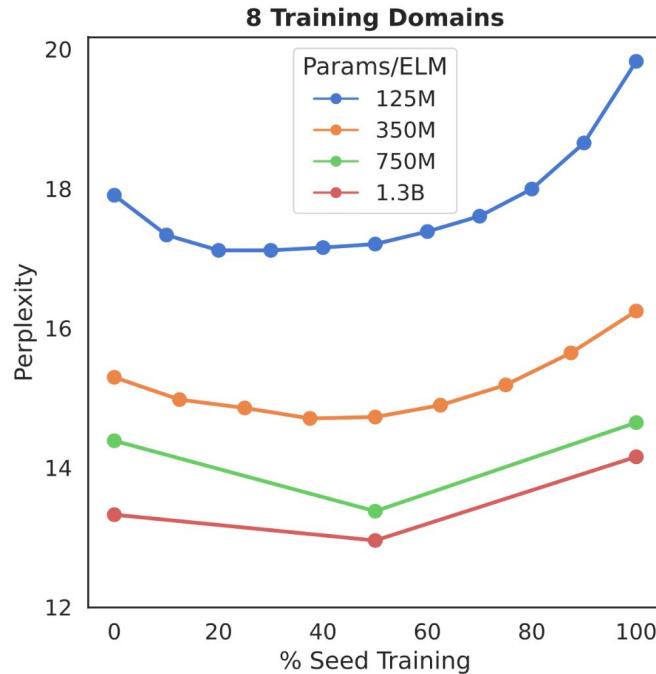
- Domain expert ensemble is very important

125M				350M			
	Random Ensemble (seed init)	ELM FOREST (random init)	ELM FOREST (seed init)		Random Ensemble (seed init)	ELM FOREST (random init)	ELM FOREST (seed init)
Train	23.0	18.2	17.2	Train	19.9	15.3	14.7
Eval	26.0	23.4	22.4	Eval	23.1	21.3	18.6
All	24.7	20.8	19.8	All	21.5	18.3	16.7
750M				1.3B			
	Random Ensemble (seed init)	ELM FOREST (random init)	ELM FOREST (seed init)		Random Ensemble (seed init)	ELM FOREST (random init)	ELM FOREST (seed init)
Train	17.4	14.4	13.4	Train	17.4	13.3	13.0
Eval	20.9	19.3	16.7	Eval	20.4	17.8	16.3
All	19.2	16.9	15.0	All	18.9	15.6	14.6

The improvement does NOT come from naive ensemble, but domain expert ensemble

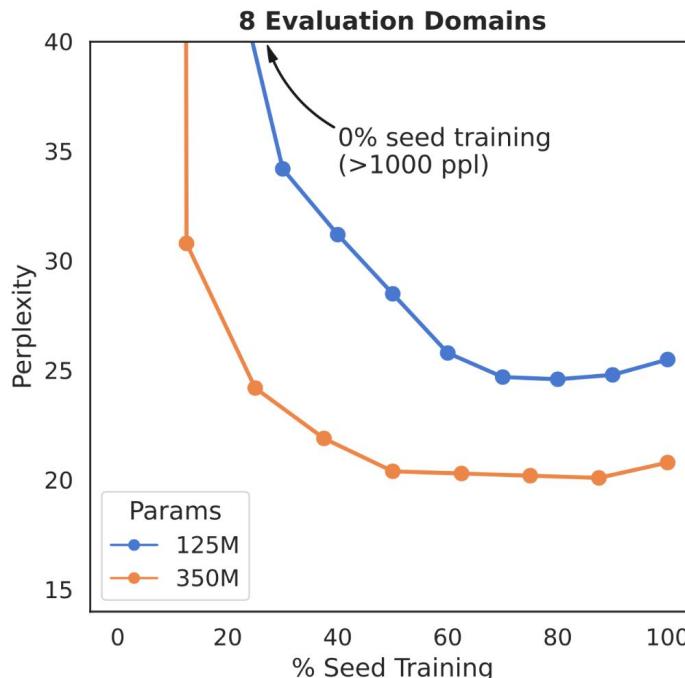
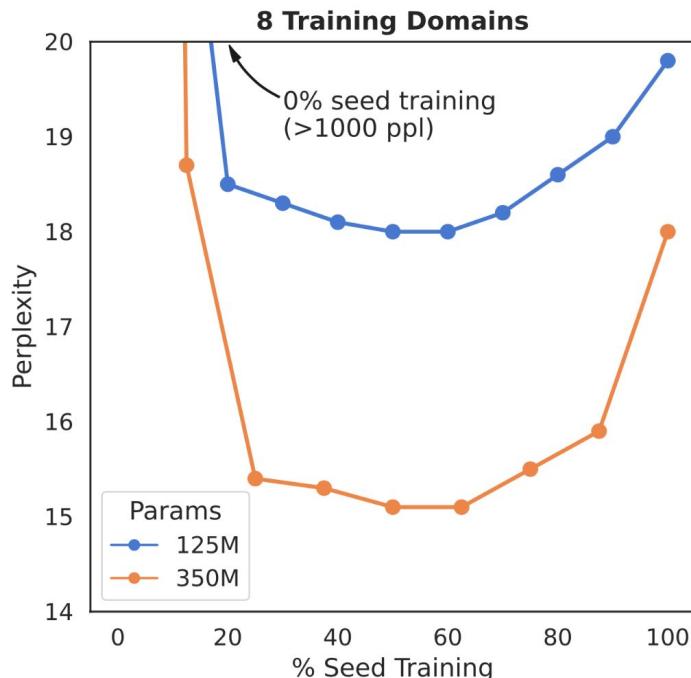
Findings and Analysis

- The final performance is robust to most seed training compute allocations



Findings and Analysis

- The seed phase is vital to parameter-level ensembled ELMs



Findings and Analysis

- Ensembling performance is robust to seed training data

seed corpus	Average Test PPL (\downarrow)		
	Train	Evaluation	Overall
TRANSFORMER-LM	19.8	25.5	22.7
8 train domains	17.2	22.7	20.0
Wikipedia	17.7	23.2	20.5
C4	17.9	23.5	20.7
StackOverflow	18.4	24.6	21.5
JavaScript	19.2	24.9	22.0

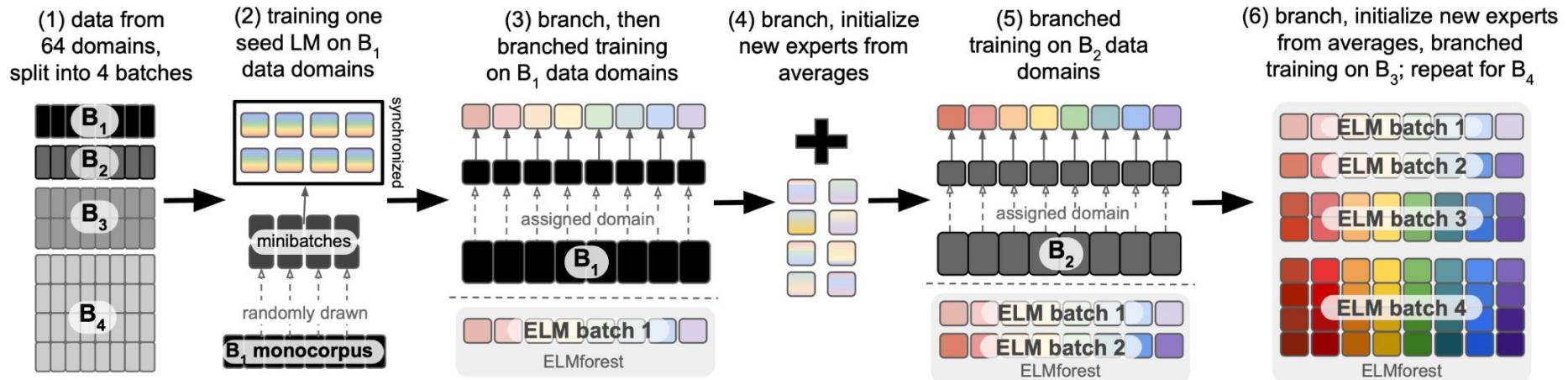
Findings and Analysis

- Domain expert removal is effective in removing domain-specific knowledge

		w/ ELM (Average Test PPL ↓)	-ELM (Δ PPL)
	8 train domains	17.2	(+9.4)
seed corpus	Wikipedia	17.7	(+11.9)
	C4	17.9	(+11.8)
	StackOverflow	18.4	(+12.7)
	JavaScript	19.2	(+13.6)

Incremental training on 64 domains, test on 16 domains

BTM incremental training (on 4 data batches totalling 64 domains)

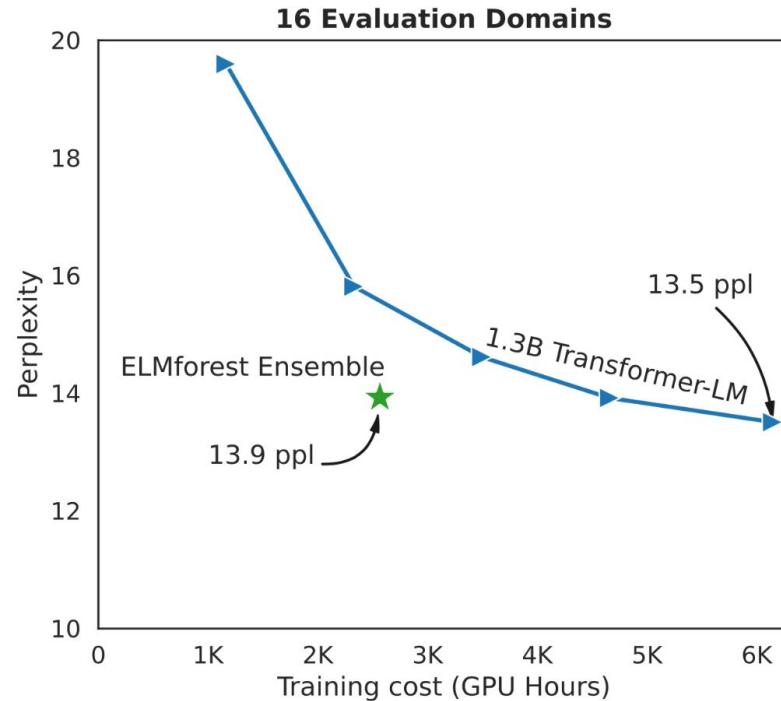
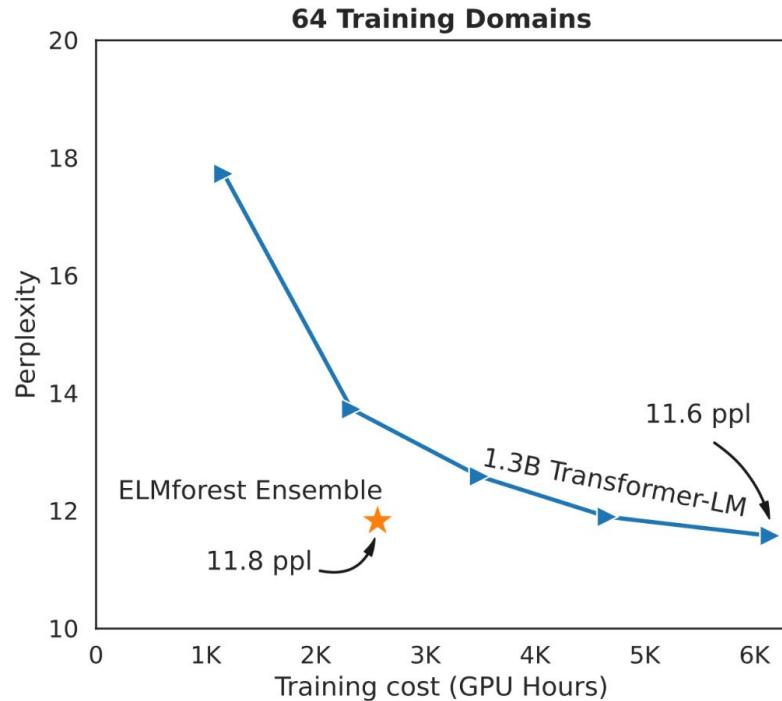


Incremental training on 64 domains, test on 16 domains

80-DOMAIN CORPUS: 192.3B WHITESPACE-SEPARATED TOKENS

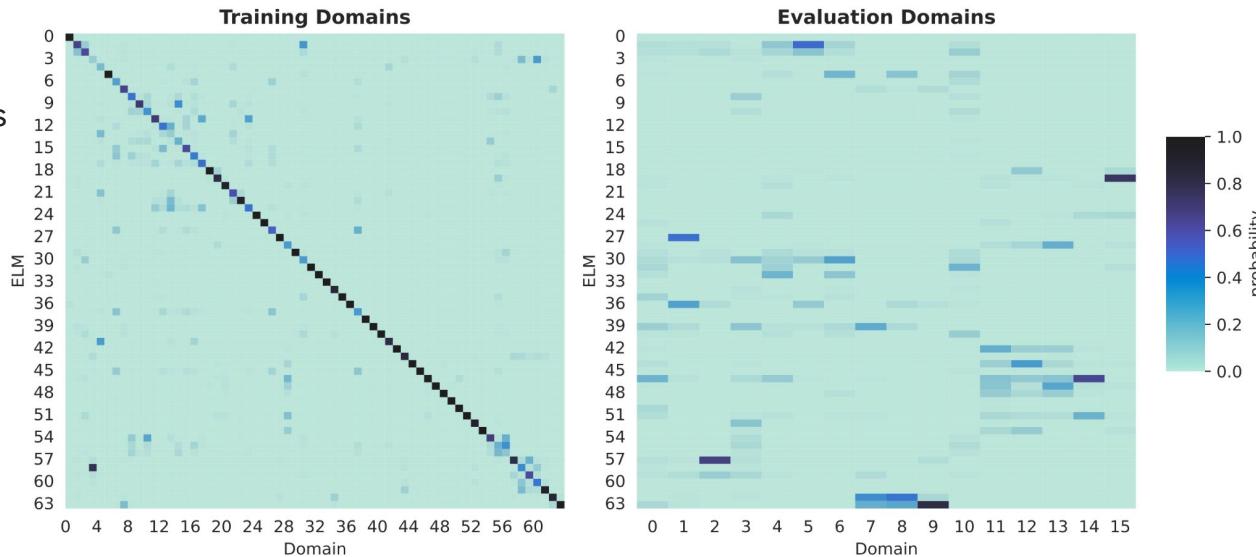
Category	Domains
SEMANTIC SCHOLAR (26.6%)	Medicine (5.2%), Biology (4.7%), CS (3.4%), Physics (2.7%), Math (2.3%), Unlabeled (1.3%), Psychology (1.2%), Chemistry (1.0%), Economics (0.8%), Engineering (0.7%), CORD19 (0.6%), Material Science (0.5%), Geology (0.5%), Sociology (0.5%), Business (0.3%), Political Science (0.2%), Geography (0.2%), Environmental Science (0.1%), History (0.1%), Philosophy (0.1%), ACL (0.1%), Art (0.05%)
GITHUB CODE (22.4%)	JavaScript (3.7%), Java (3.5%), HTML (2.7%), C (2.5%), C++ (1.9%), Python (1.5%), C# (1.2%), PHP (1.1%), Markdown (1.1%), Code Contests (1.0%), GO (1.0%), CSS (0.7%), Ruby (0.4%)
WEB FORUMS (17.5%)	Reddit Dialogues (13.0%), StackOverflow (1.7%), Twitter (0.9%), StackExchange (0.8%), HackerNews (0.4%), Gaming Subreddits (0.1%), Sports Subreddits (0.1%)
WEB CRAWL (16.0%)	C4 (5.2%), RealNews (5.2%), OpenWebText (3.4%), Wikipedia (en) (1.3%), WMT News Crawl 2021 (0.5%), 1B Words Corpus (0.4%)
BOOKS (5.8%)	Stories (3.8%), Gutenberg Books (1.6%), BookCorpus (0.4%)
LEGAL TEXT (5.5%)	Legal Case Law (5.5%), Supreme Court Opinions (HTML) (0.1%)
REVIEWS (5.0%)	Books Reviews (2.1%), Amazon Reviews (1.1%), Electronics Reviews (0.5%), Clothing, Shoes and Jewelry Reviews (0.5%), Home and Kitchen Reviews (0.4%), Yelp Reviews (0.3%), Sports and Outdoors Reviews (0.3%), Movies and TV Reviews (0.3%)
OTHER (1.3%)	DM Mathematics (0.8%), OpenSubtitles (0.4%), USPTO (0.1%)
EVALUATION DOMAINS (TEST ONLY)	Enron, #COVID-19 Tweets, IMDB, TOEFL exams, Congressional bills, Legal Contracts, /r/cscareerquestions, /r/india, /r/hiphopheads, Irish Parliamentary Speeches, SQL, Rust, Perl , TeX, FORTRAN, Breaking News

Incremental training on 64 domains, test on 16 domains



Incremental training on 64 domains, test on 16 domains

Domain posteriors



Evaluation Domain	Top-1	Top-2	Top-3
<i>Covid Tweets</i>	Twitter (0.48)	2021 Newscrawl (0.29)	HackerNews (0.05)
<i>TeX</i>	stackexchange (0.64)	Markdown (0.22)	HTML (0.05)
<i>Congressional Bills</i>	SCOTUS Opinions (0.21)	OpenWebText (0.13)	Stackexchange (0.11)
<i>IMDB</i>	Movie & TV Reviews (0.66)	Book Reviews (0.11)	RealNews (0.05)
<i>Legal Contracts</i>	C4 (0.30)	Legal (0.21)	SCOTUS Opinions (0.14)

Summary

- Propose a Branch-Train-Merge pipeline of expert language models to eliminate the need of massive multi-node synchronization
- Support flexible adding of new domains and removal of old domains
- Much better performance than standard transformer-LM, at much lower training budget and the same inference cost



Retrieval Augmented Language Modeling

10.31 2023

Retrieval Models vs. Generative Models

User: What are some good restaurants in New York City?

Retrieval Model: 1) ABC Kitchen, 2) Le Bernardin, 3) Peter Luger Steak House, 4) Gramercy Tavern, 5) Eleven Madison Park.

User: I'm planning a trip to New York City. Can you recommend some good restaurants for dinner?

ChatGPT: Of course! New York City is known for its **diverse culinary** scene. Here are a few highly recommended restaurants for dinner: 1) Per Se - a renowned **fine dining** experience with exceptional tasting menus, 2) Momofuku Ko - offering innovative and delicious **Asian-inspired** cuisine, 3) L'Artusi - known for its **modern Italian** dishes and cozy ambiance, 4) Cosme - serving innovative **Mexican cuisine** with a contemporary twist, and 5) Gramercy Tavern - a beloved restaurant known for its seasonal **American** fare and **warm hospitality**.

Why augment a LM with a retrieval model?

- ❖ Retrieval Model
 - Explicitly find specific knowledge
 - Easy to add, edit or delete
 - Lack the ability to generate creative or novel content
 - Hard to build a retrieval model for a new task

- ❖ Language Model
 - Generating creative and coherent text
 - Can do zero/few-shot generalization, instruction following, CoT reasoning.
 - Hallucination

Why augment a LM with a retrieval model?

User: who was the sole survivor of Titanic?

ChatGPT: The sole survivor of the Titanic disaster was a man named **Charles Joughin**. He was a chef on board the ship and managed to survive by clinging to a capsized lifeboat until he was eventually rescued by the RMS Carpathia.

User: Can you give me a bibliography of sources that prove he was the sole survivor of Titanic?

ChatGPT: I apologize for any confusion, but it seems there might be a misunderstanding. Charles Joughin was not the sole survivor of the Titanic disaster. There were approximately **705 survivors** out of the estimated 2,224 passengers and crew on board.

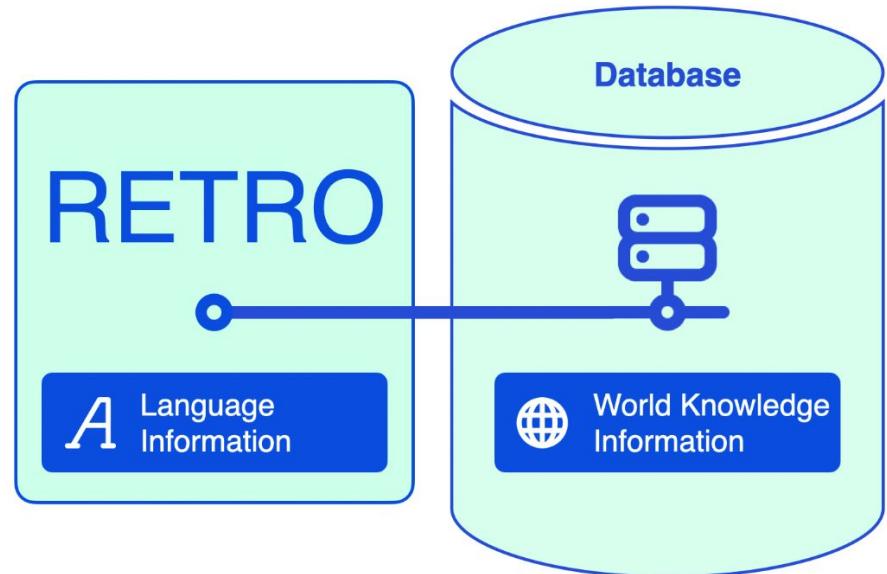
A retrieval-augmented LM combines these two models to overcome their individual limitations



Improving language models by retrieving from trillions of tokens



❑ High computational cost



- ★ Separate Language Information from World Knowledge Information
- ★ LM + Retrieval Can Equal LLM

Improving language models by retrieving from trillions of tokens

➤ Key-Value Database Construction

Value $[N, F]$:

Neighbor: compute the key (N)

Completion: the continuation of N in the original document. (F)

Key $BERT(N)$: BERT sentence embedding of N

	Dune is a 2021 American epic science fiction film directed by Denis Villeneuve	NEIGHBOR
	It is the first of a planned two-part adaptation of the 1965 novel by Frank Herbert	COMPLETION

Improving language models by retrieving from trillions of tokens

2 trillion multi-lingual tokens

Table 1 | **MassiveText**. The last column indicates the sampling weight during training. The multilingual subsets include documents in 10 languages. The full breakdown is given in §A.1.

Source	Token count (M)	Documents (M)	Multilingual	Sampling frequency
Web	977,563	1,208	Yes	55%
Books	3,423,740	20	No	25%
News	236,918	398	No	10%
Wikipedia	13,288	23	Yes	5%
GitHub	374,952	143	No	5%

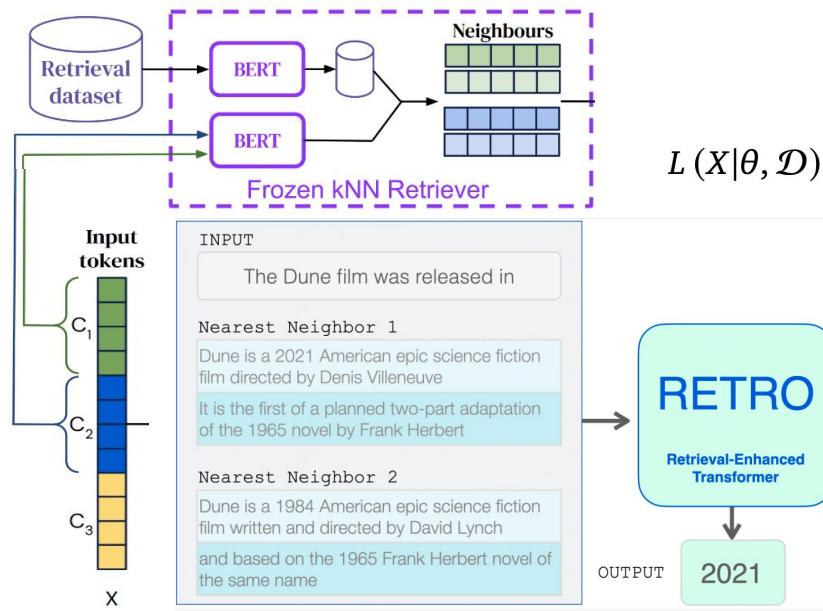


Improving language models by retrieving from trillions of tokens

- Approximate Nearest Neighbor Search
- Retrieval-Enhanced Autoregressive token model

ScaNN

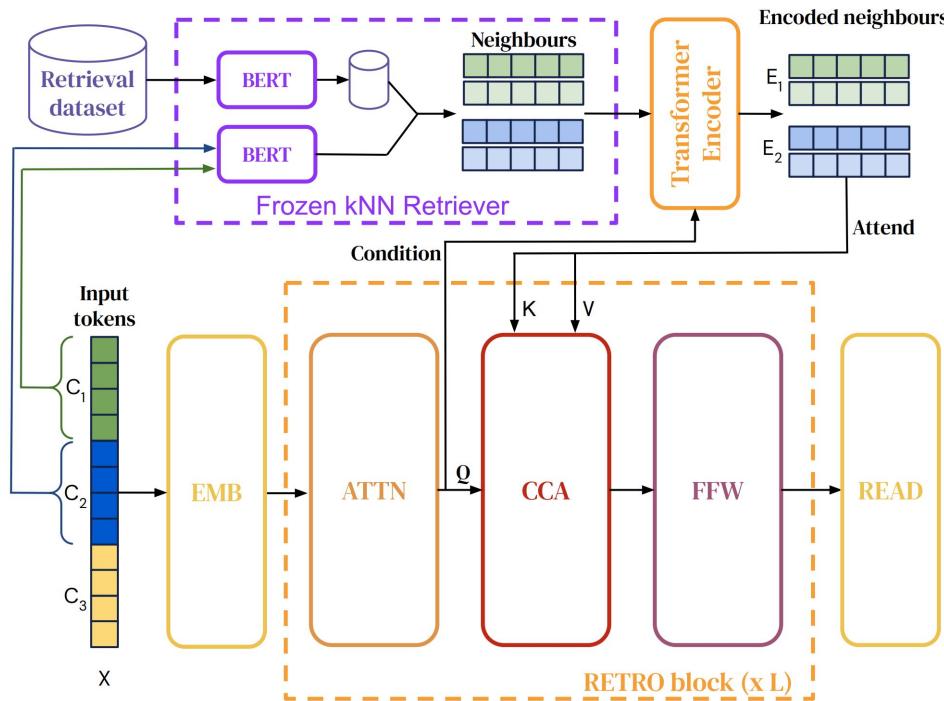
$O(\log T)$



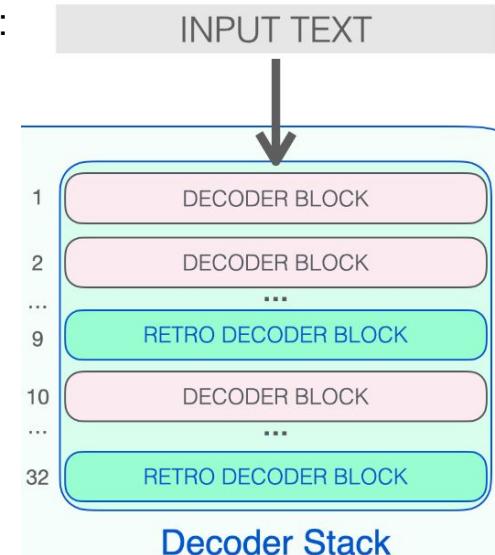
$$L(X|\theta, \mathcal{D}) \triangleq \sum_{u=1}^l \sum_{i=1}^m \ell_\theta(x_{(u-1)m+i} | (x_j)_{j < (u-1)m+i}, (\text{RET}_{\mathcal{D}}(C_{u'}))_{u' < u}).$$

Improving language models by retrieving from trillions of tokens

➤ Retro Model Architecture



- Encoder: Encode the neighbours.
- Decoder:

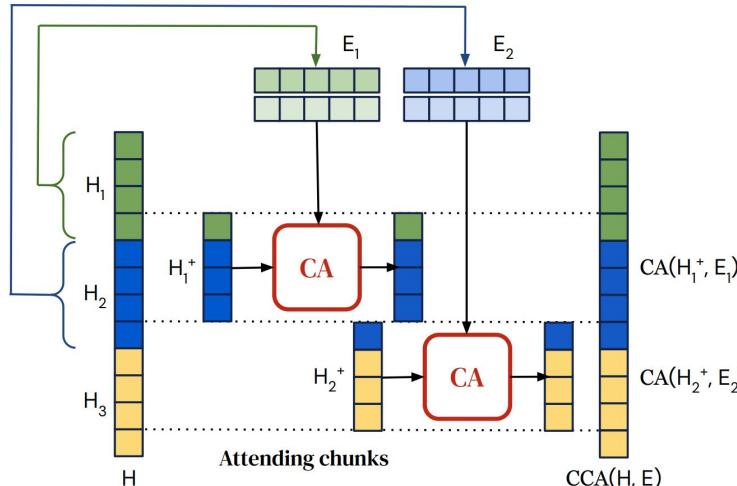


Improving language models by retrieving from trillions of tokens

➤ Chunked Cross-Attention

Chunked cross-attention (CCA)

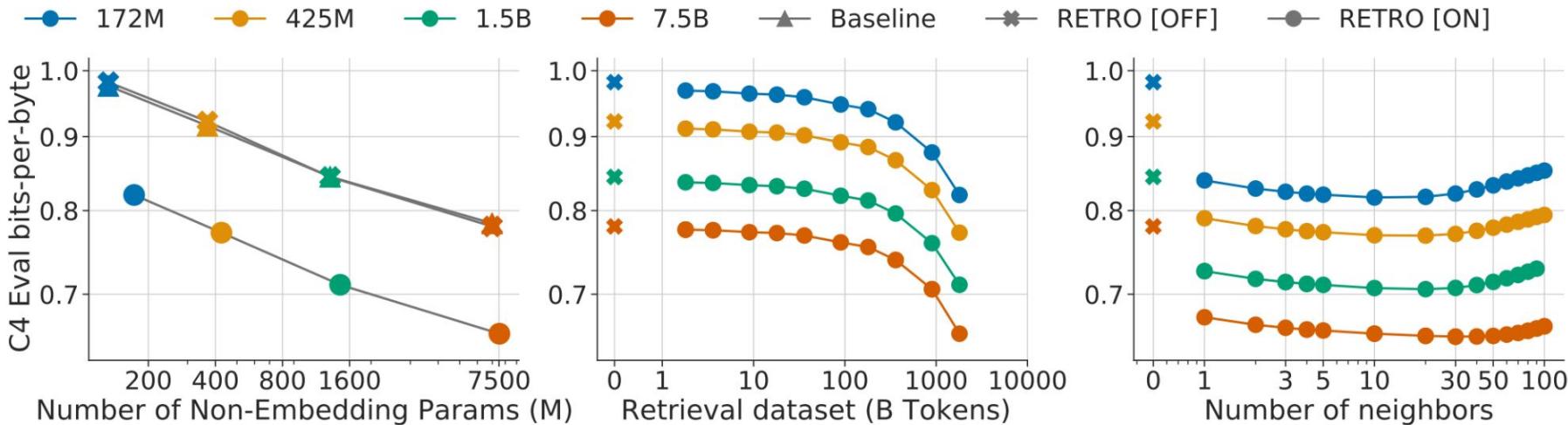
Encoded neighbours



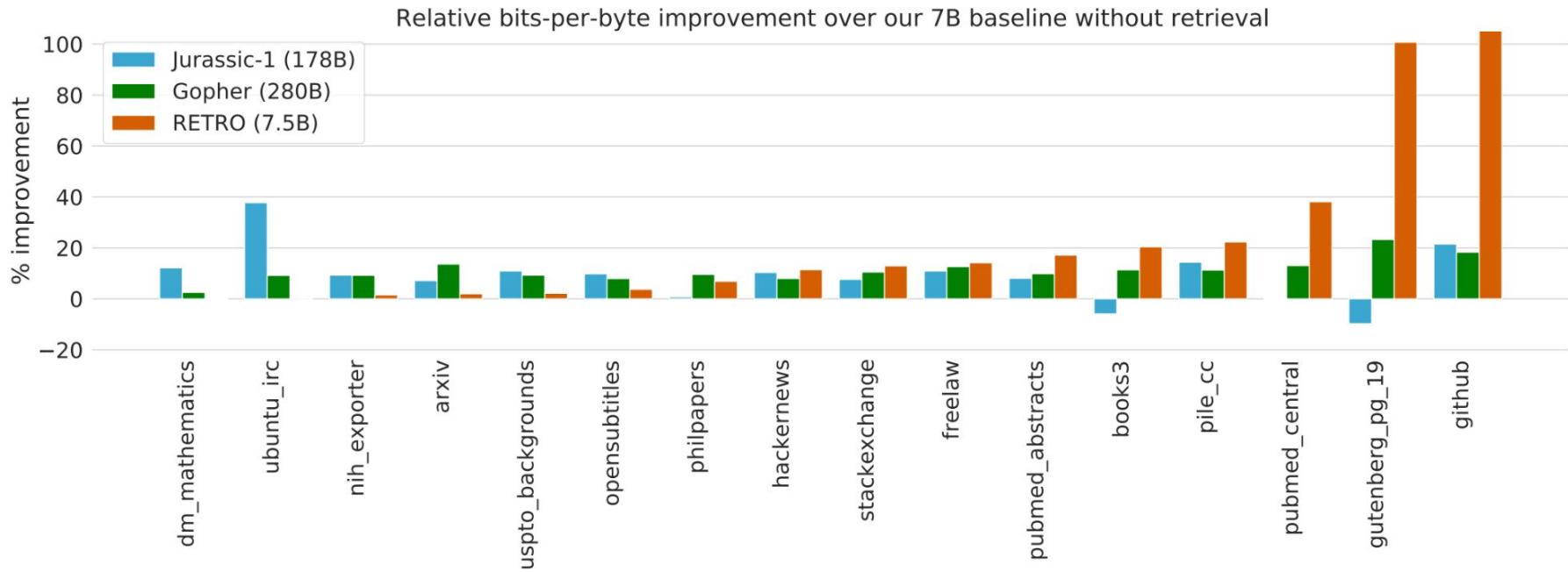
- $I-1$ attending trunks
- $H \leftarrow \text{EMB}(X)$
for $p \in [1, L]$ **do**
 $H \leftarrow \text{ATTN}(H)$ // Causal attention
 if $p = \min(P)$ **then**
 $E = \text{ENCODER}(\text{RET}(C_u)_{1 \leq u \leq l}, H)$
 if $p \in P$ **then**
 $H \leftarrow \text{CCA}(H, E)$
 $H \leftarrow \text{FFW}(H)$
 $O \leftarrow \text{READ}(H)$

Improving language models by retrieving from trillions of tokens

➤ Scaling of Ratio

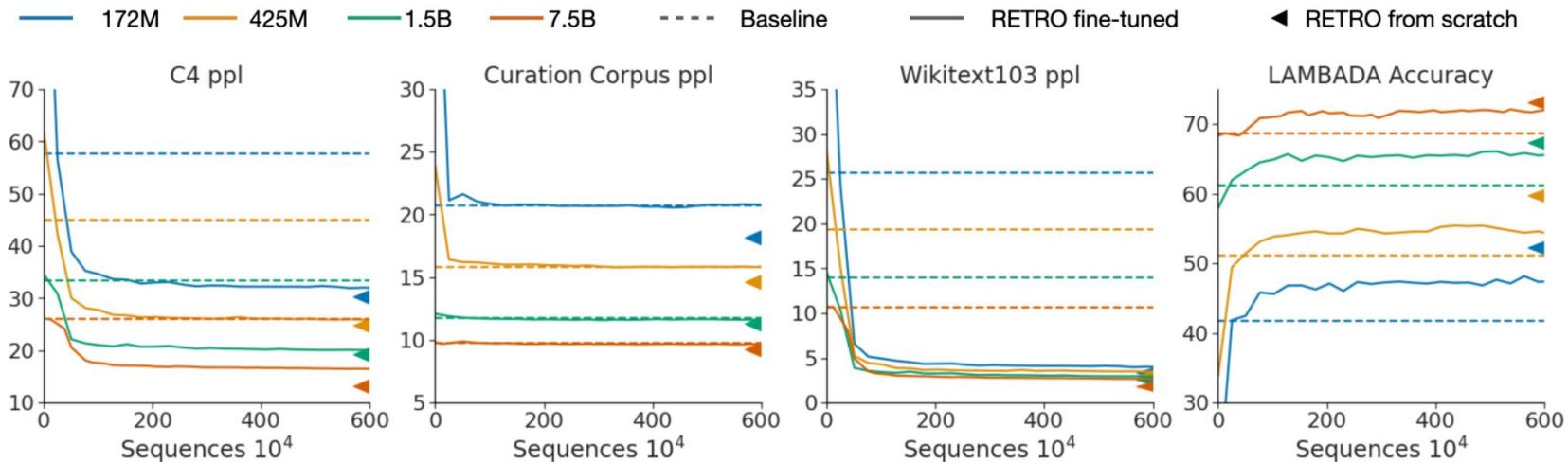


Improving language models by retrieving from trillions of tokens



Improving language models by retrieving from trillions of tokens

- Retro-fitting a baseline transformer



Discussion

- What is the drawback of the proposed method (Branch-Train-Merge)?
- Pros and cons of training retrieval models and language models independently.
- A technical problem of mine: what's the purpose of designing the attending trunks to have one word overlap in Retro?