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

Margaret Li[*,†,◇] **Suchin Gururangan**[*,†,◇] **Tim Dettmers**[†]

Mike Lewis[◇] **Tim Althoff**[†] **Noah A. Smith**[†,♠] **Luke Zettlemoyer**[†,◇]

[†]Paul G. Allen School of Computer Science & Engineering, University of Washington
[♠]Allen Institute for AI
[◇]Meta AI

Presenters: Duong Minh Le, Avirath Tibrewala

DEMIX Layers: Disentangling Domains for Modular Language Modeling

Suchin Gururangan[†][◇] **Mike Lewis**[◇] **Ari Holtzman**[†]
Noah A. Smith[†][♠] **Luke Zettlemoyer**[†][◇]
[†]Paul G. Allen School of Computer Science & Engineering, University of Washington
[♠]Allen Institute for AI
[◇]Meta AI
Seattle, WA, USA
sg01@cs.washington.edu

Demix

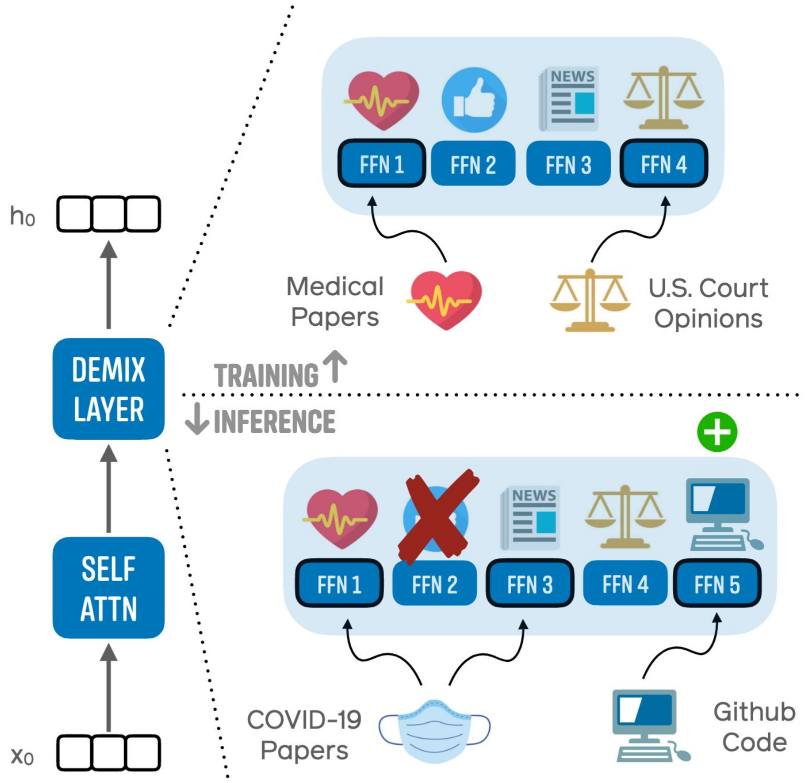
- Goal: make LLM modular
- Benefits of modular LLM?

Demix

- Goal: make LLM modular
- Benefits of modular LLM?
  - Easy to adapt to new domain without forgetting the old knowledge
  - Easy to remove or restrict access to domains that LLM has learned

Demix

- Goal: make LLM modular



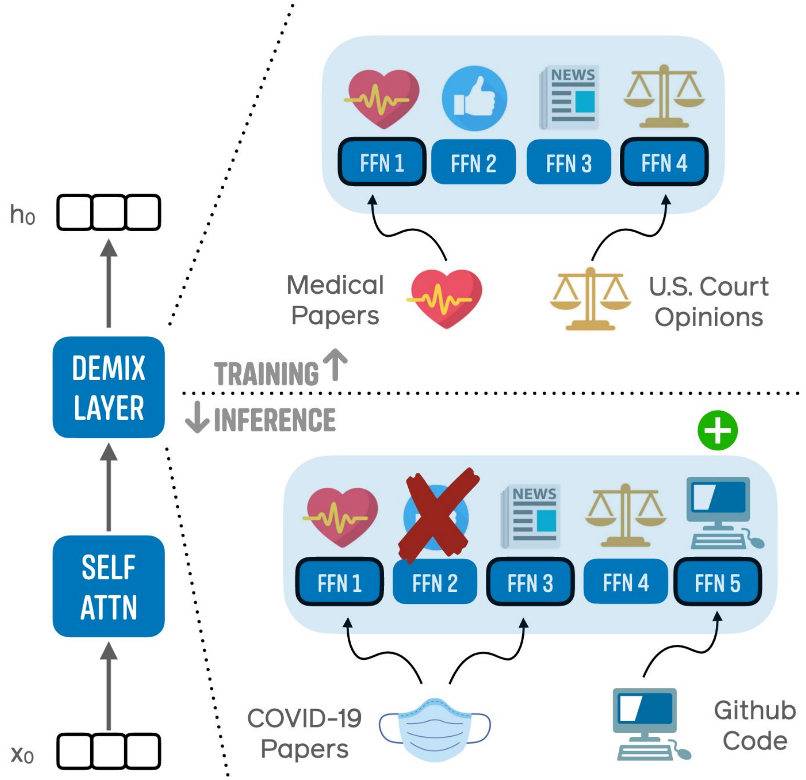
Demix at training

- Split the corpus by domain
- Train each expert on one domain
- Each expert is a FFN

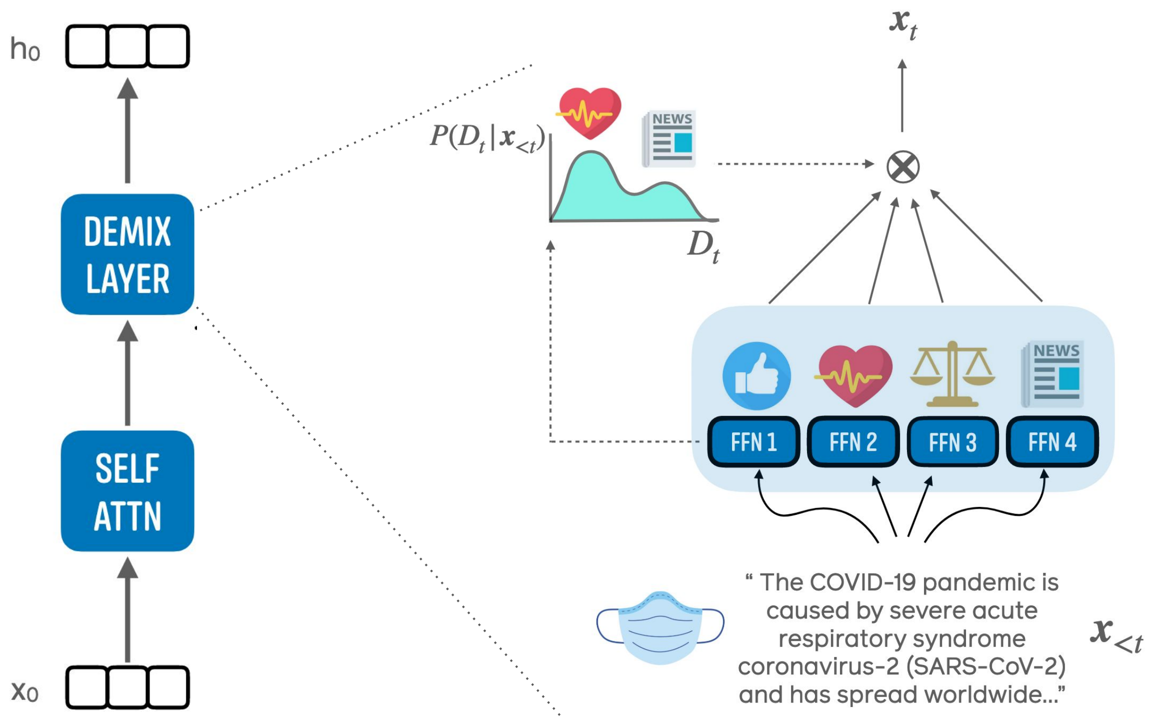
$$\mathrm{FFN}(\mathbf{h}_{t,\ell-1}) = \sum_{j=1}^{n} g_j(\mathbf{h}_{t,\ell-1}) \cdot \mathrm{FFN}_j(\mathbf{h}_{t,\ell-1})$$

$$g_j(\mathbf{h}_{t,\ell}) = \begin{cases} 1 & \text{if } j = d \\ 0 & \text{otherwise} \end{cases}$$

t: time step
l: layer
d: domain
j: expert id

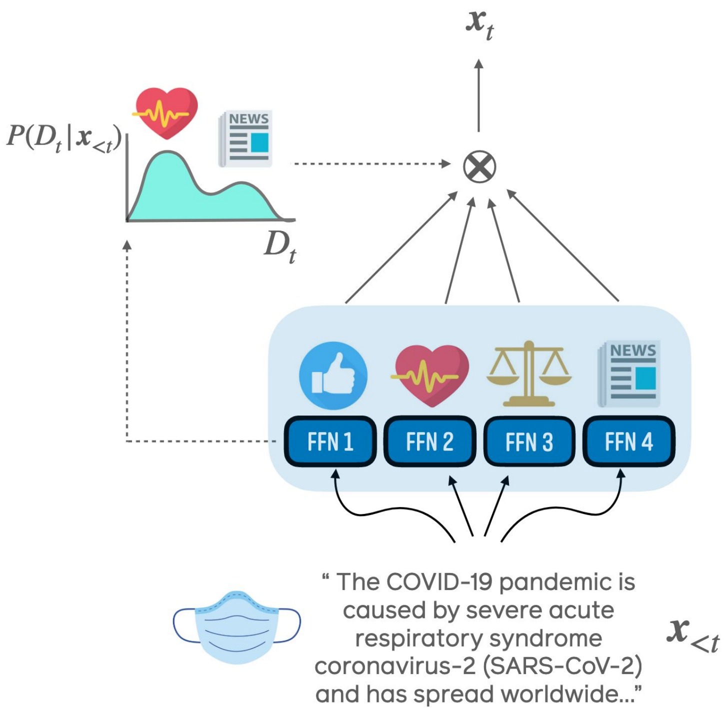


Demix at inference



h_0
DEMIX LAYER
SELF ATTN
x_0
x_t
$P(D_t | x_{<t})$
NEWS
D_t
FFN 1
FFN 2
FFN 3
FFN 4
NEWS
“ The COVID-19 pandemic is caused by severe acute respiratory syndrome coronavirus-2 (SARS-CoV-2) and has spread worldwide...”
$x_{<t}$

Domain posterior

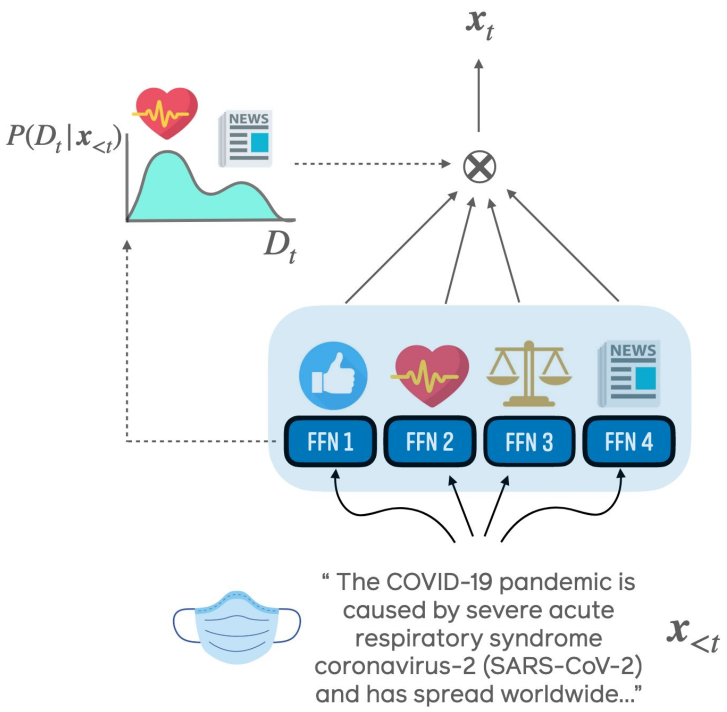

x_t
$P(D_t|x_{<t})$
NEWS
D_t
NEWS
FFN 1
FFN 2
FFN 3
FFN 4
" The COVID-19 pandemic is caused by severe acute respiratory syndrome coronavirus-2 (SARS-CoV-2) and has spread worldwide..."
$x_{<t}$

LM objective

$$p(\boldsymbol{x}_t \mid \boldsymbol{x}_{<t}) = \sum_{j=1}^{n} p(\boldsymbol{x}_t \mid \boldsymbol{x}_{<t}, D_t = j) \cdot \underbrace{p(D_t = j \mid \boldsymbol{x}_{<t})}_{g_j}$$

Domain posterior

Domain posterior


x_t
$P(D_t|x_{<t})$
NEWS
D_t
NEWS
FFN 1
FFN 2
FFN 3
FFN 4
" The COVID-19 pandemic is caused by severe acute respiratory syndrome coronavirus-2 (SARS-CoV-2) and has spread worldwide..."
$x_{<t}$

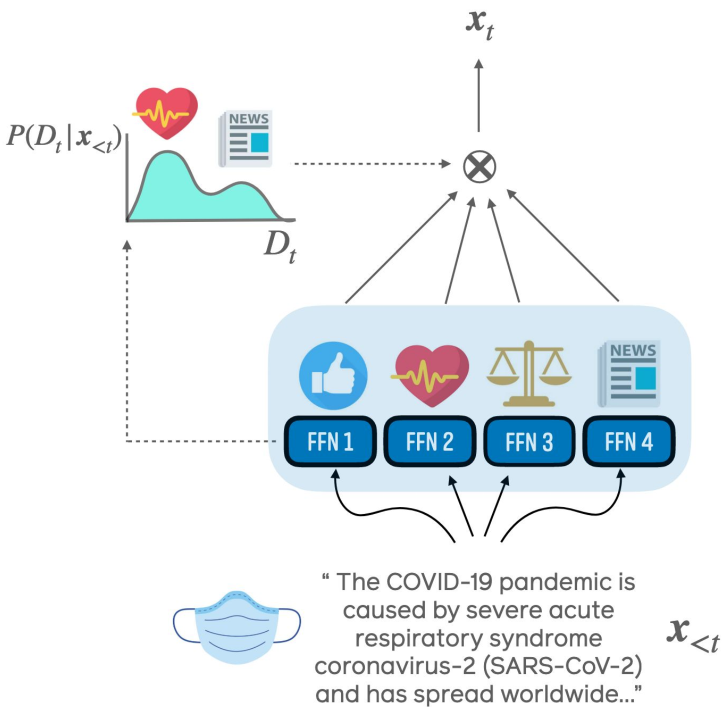
$$p(\boldsymbol{x}_t \mid \boldsymbol{x}_{<t}) = \sum_{j=1}^{n} p(\boldsymbol{x}_t \mid \boldsymbol{x}_{<t}, D_t = j) \cdot \underbrace{p(D_t = j \mid \boldsymbol{x}_{<t})}_{g_j}$$

$$p(D_t = j \mid \boldsymbol{x}_{<t}) = \frac{p(\boldsymbol{x}_{<t} \mid D_t = j) \cdot p(D_t = j)}{p(\boldsymbol{x}_{<t})} = \frac{p(\boldsymbol{x}_{<t} \mid D_t = j) \cdot p(D_t = j)}{\sum_{j'=1}^{n} \underline{p(\boldsymbol{x}_{<t} \mid D_t = j')} \cdot \underline{p(D_t = j')}} \qquad (5)$$

Estimate by each expert LM

Domain prior?

Domain posterior


x_t
$P(D_t|x_{<t})$
NEWS
D_t
FFN 1
FFN 2
FFN 3
FFN 4
NEWS
" The COVID-19 pandemic is caused by severe acute respiratory syndrome coronavirus-2 (SARS-CoV-2) and has spread worldwide..."
$x_{<t}$

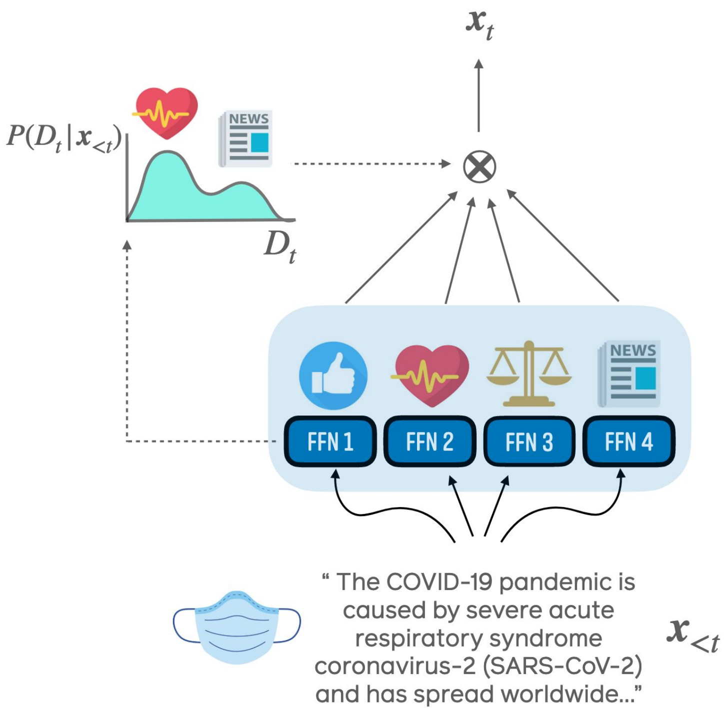
$$p(\boldsymbol{x}_t \mid \boldsymbol{x}_{<t}) = \sum_{j=1}^{n} p(\boldsymbol{x}_t \mid \boldsymbol{x}_{<t}, D_t = j) \cdot \underbrace{p(D_t = j \mid \boldsymbol{x}_{<t})}_{g_j}$$

$$p(D_t = j \mid \boldsymbol{x}_{<t}) = \frac{p(\boldsymbol{x}_{<t} \mid D_t = j) \cdot p(D_t = j)}{p(\boldsymbol{x}_{<t})} = \frac{p(\boldsymbol{x}_{<t} \mid D_t = j) \cdot p(D_t = j)}{\sum_{j'=1}^{n} p(\boldsymbol{x}_{<t} \mid D_t = j') \cdot p(D_t = j')} \quad (5)$$

Domain prior

$$p(D_t = j) \propto \sum_{t'=1}^{t-1} \lambda^{t-t'} \cdot p(D_{t'} = j \mid \boldsymbol{x}_{<t'})$$

Domain posterior



x_t
$P(D_t | x_{<t})$
NEWS
D_t
NEWS
FFN 1
FFN 2
FFN 3
FFN 4
" The COVID-19 pandemic is caused by severe acute respiratory syndrome coronavirus-2 (SARS-CoV-2) and has spread worldwide..."
$x_{<t}$

$$p(\boldsymbol{x}_t \mid \boldsymbol{x}_{<t}) = \sum_{j=1}^{n} p(\boldsymbol{x}_t \mid \boldsymbol{x}_{<t}, D_t = j) \cdot \underbrace{p(D_t = j \mid \boldsymbol{x}_{<t})}_{g_j}$$

$$p(D_t = j \mid \boldsymbol{x}_{<t}) = \frac{p(\boldsymbol{x}_{<t} \mid D_t = j) \cdot p(D_t = j)}{p(\boldsymbol{x}_{<t})} = \frac{p(\boldsymbol{x}_{<t} \mid D_t = j) \cdot p(D_t = j)}{\sum_{j'=1}^{n} p(\boldsymbol{x}_{<t} \mid D_t = j') \cdot p(D_t = j')} \qquad (5)$$

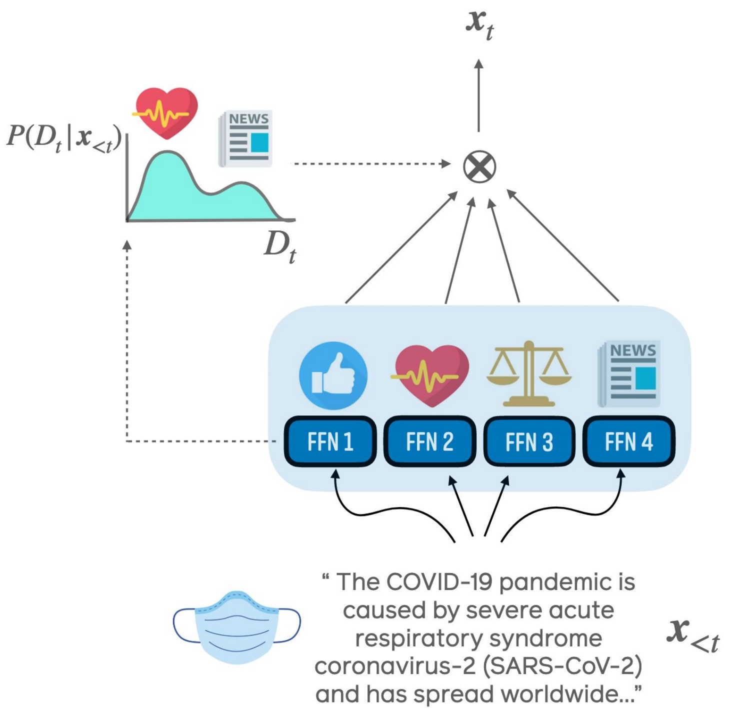
Domain prior

$$p(D_t = j) \propto \sum_{t'=1}^{t-1} \lambda^{t-t'} \cdot p(D_{t'} = j \mid \boldsymbol{x}_{<t'})$$

Cached prior

* Pick 100 sequences from the dev data from a domain to compute the domain prior
* The prior is computed at the end of each sequence
* This prior is fixed/cache in inference

Domain posterior



x_t
$P(D_t|x_{<t})$
NEWS
D_t
NEWS
FFN 1
FFN 2
FFN 3
FFN 4
" The COVID-19 pandemic is caused by severe acute respiratory syndrome coronavirus-2 (SARS-CoV-2) and has spread worldwide..."
$x_{<t}$

LLM objective

$$p(\boldsymbol{x}_t \mid \boldsymbol{x}_{<t}) = \sum_{j=1}^{n} p(\boldsymbol{x}_t \mid \boldsymbol{x}_{<t}, D_t = j) \cdot \underbrace{p(D_t = j \mid \boldsymbol{x}_{<t})}_{g_j}$$

Domain posterior

$$p(D_t = j \mid \boldsymbol{x}_{<t}) = \frac{p(\boldsymbol{x}_{<t} \mid D_t = j) \cdot p(D_t = j)}{p(\boldsymbol{x}_{<t})} \qquad (5)$$

$$= \frac{p(\boldsymbol{x}_{<t} \mid D_t = j) \cdot p(D_t = j)}{\sum_{j'=1}^{n} p(\boldsymbol{x}_{<t} \mid D_t = j') \cdot p(D_t = j')}$$

Domain prior

$$p(D_t = j) \propto \sum_{t'=1}^{t-1} \lambda^{t-t'} \cdot p(D_{t'} = j \mid \boldsymbol{x}_{<t'})$$

(+) Parameter free routing
(-) Require to have access a small amount of data in the test domain

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

Margaret Li[*†◇] **Suchin Gururangan**[*†◇] **Tim Dettmers**[†]

Mike Lewis[◇] **Tim Althoff**[†] **Noah A. Smith**[†♠] **Luke Zettlemoyer**[†◇]

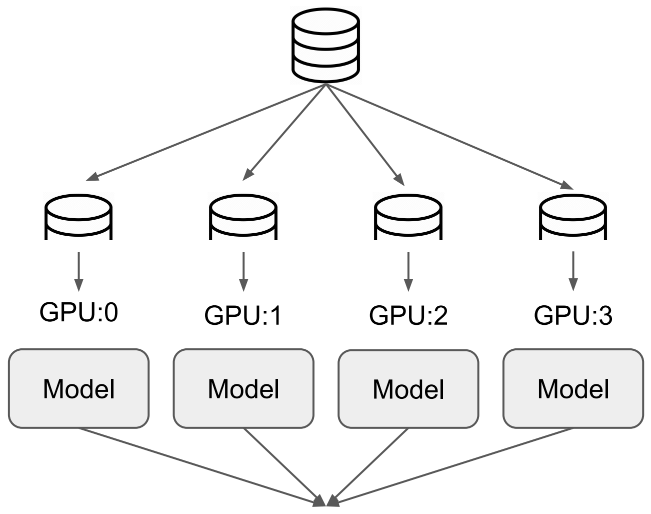
[†]Paul G. Allen School of Computer Science & Engineering, University of Washington
[♠]Allen Institute for AI
[◇]Meta AI

Key idea: Train an ensemble of LLMs in parallel then merge to eliminate the GPU communication in training

Distributed data parallelism

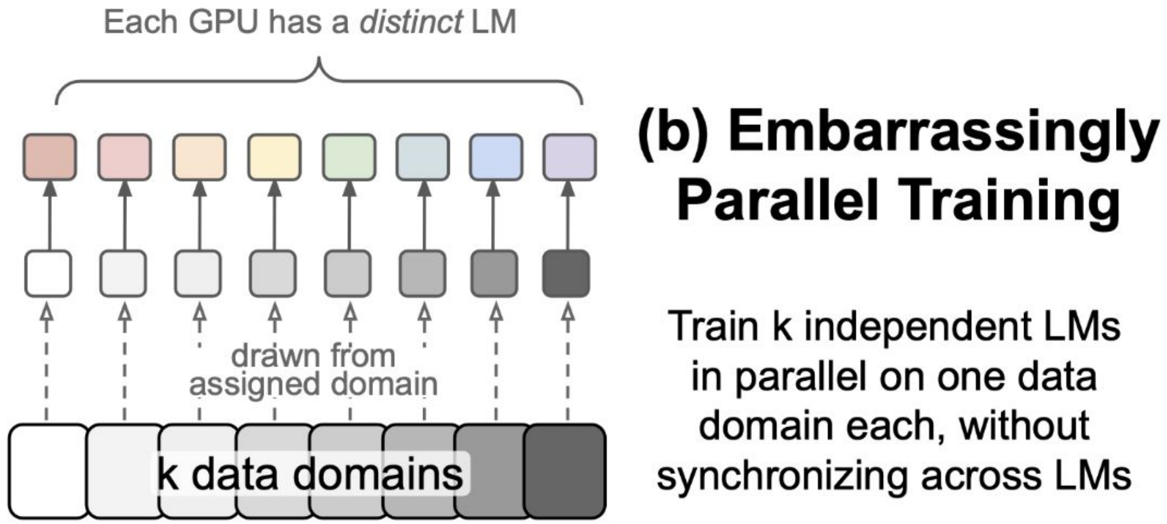
One minibatch from the corpus

One training step

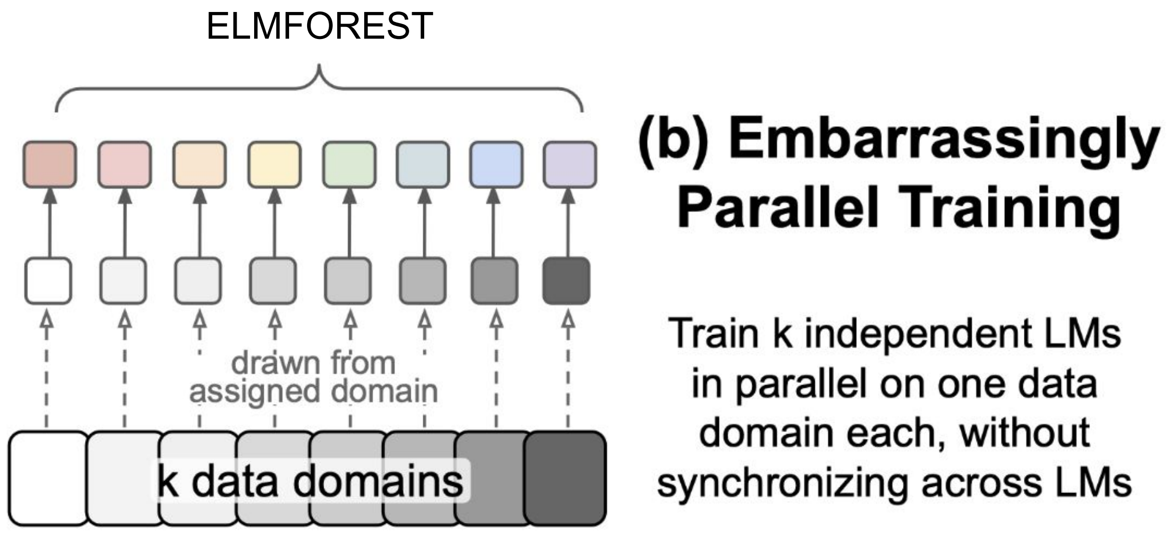


Compute gradients then synchronize

Key idea: Train an ensemble of LLMs in parallel then merge to eliminate the GPU communication in training


Each GPU has a *distinct* LM
drawn from assigned domain
k data domains
(b) Embarrassingly Parallel Training
Train k independent LMs in parallel on one data domain each, without synchronizing across LMs

Key idea: Train an ensemble of LLMs in parallel then merge to eliminate the GPU communication in training


ELMFOREST
drawn from assigned domain
k data domains
(b) Embarrassingly Parallel Training
Train k independent LMs in parallel on one data domain each, without synchronizing across LMs

Ensembling the ELMFOREST

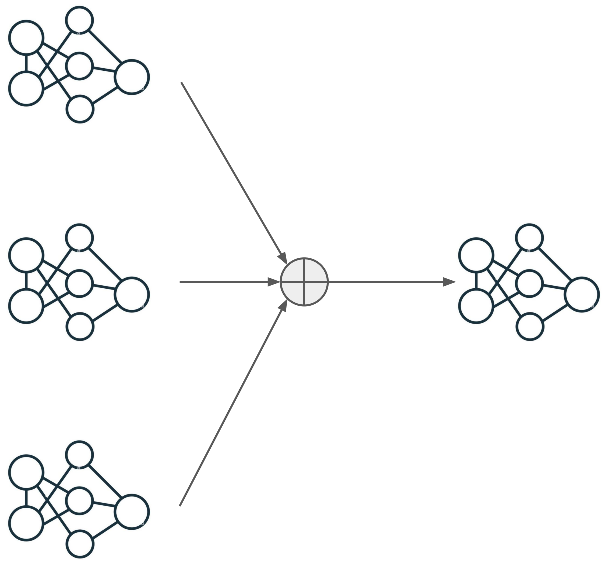
$$p(X_t \mid \boldsymbol{x}_{<t}) = \sum_{j=1}^{n} p(X_t \mid \boldsymbol{x}_{<t}, D = j) \cdot \underbrace{p(D = j \mid \boldsymbol{x}_{<t})}_{\text{Domain posterior}}$$

$$p(D = j \mid \boldsymbol{x}_{<t}) = \frac{p(\boldsymbol{x}_{<t} \mid D = j) \cdot p(D = j)}{\sum_{j'=1}^{k} p(\boldsymbol{x}_{<t} \mid D = j') \cdot p(D = j')}$$

$$p(D = j) = \sum_{i=1}^{N} \lambda^i \cdot p(D = j \mid x_{<T}^{(i)})$$

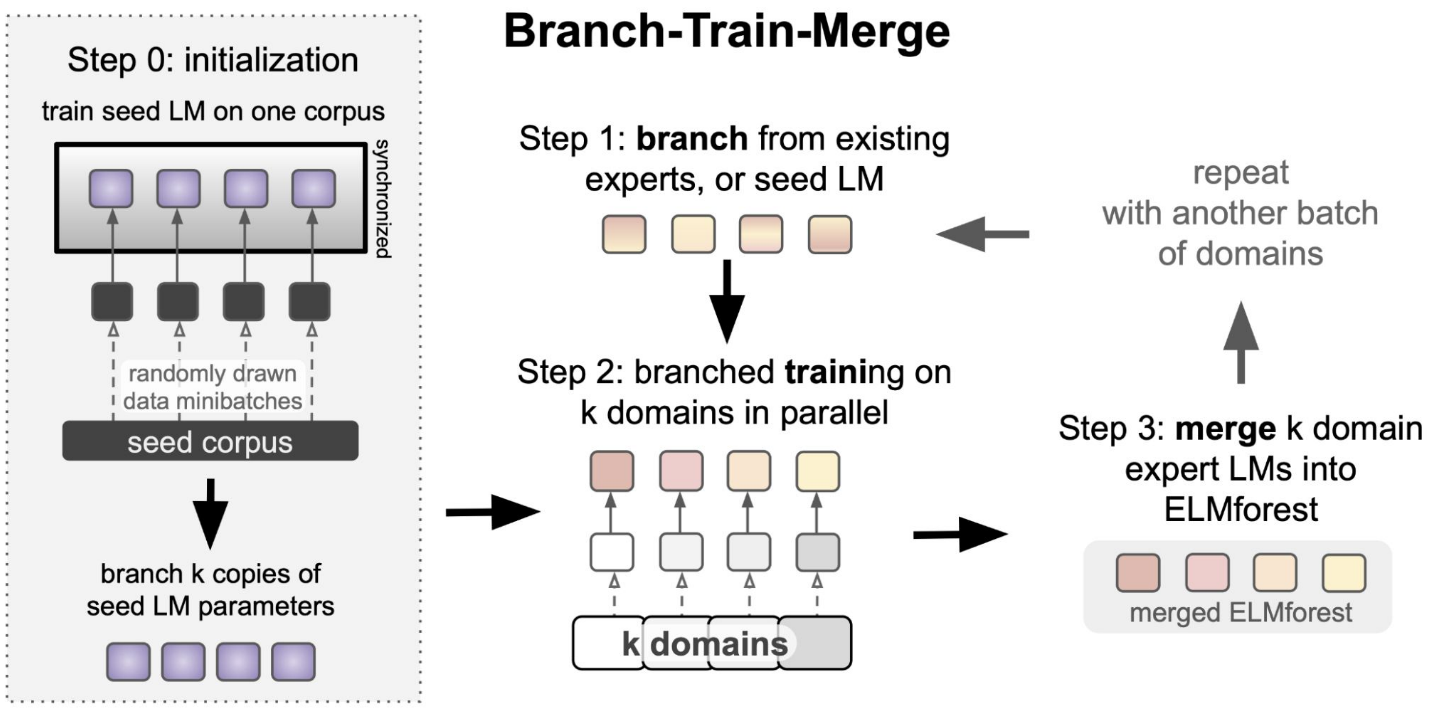
The efficiency depends on the sparsity of the domain posterior

Average ELM parameters

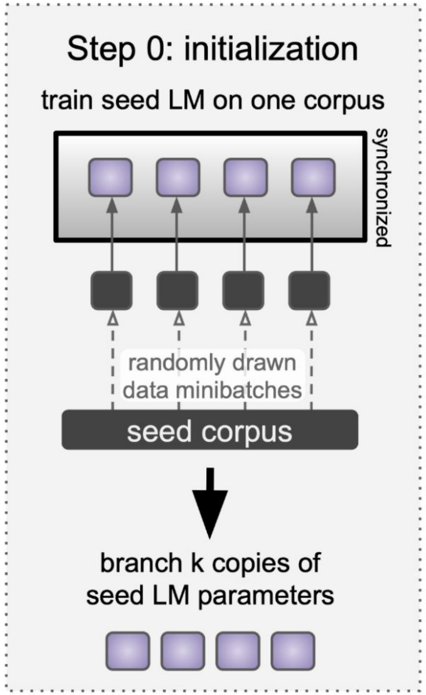


⊕ Averaging operator

The Branch-Train-Merge Iteration

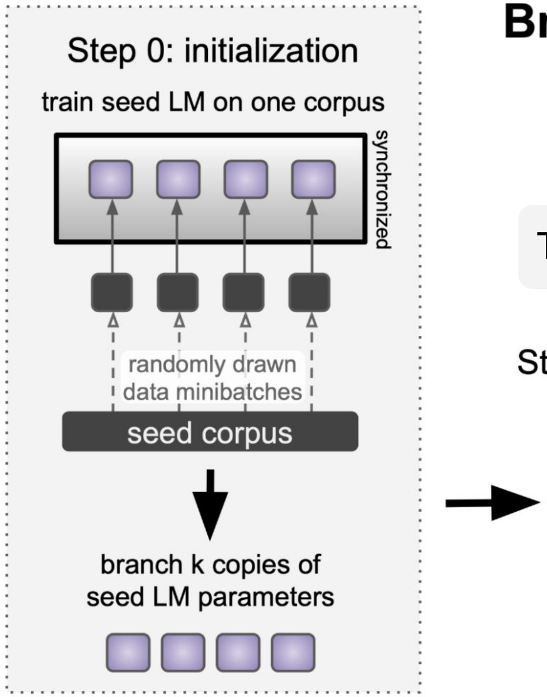

Branch-Train-Merge
Step 0: initialization
train seed LM on one corpus
synchronized
randomly drawn data minibatches
seed corpus
branch k copies of seed LM parameters
Step 1: branch from existing experts, or seed LM
Step 2: branched training on k domains in parallel
k domains
Step 3: merge k domain expert LMs into ELMforest
merged ELMforest
repeat with another batch of domains

The Branch-Train-Merge Iteration


Step 0: initialization
train seed LM on one corpus
synchronized
randomly drawn data minibatches
seed corpus
branch k copies of seed LM parameters

Branch-Train-Merge

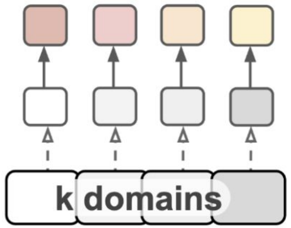
The Branch-Train-Merge Iteration


Step 0: initialization
train seed LM on one corpus
synchronized
randomly drawn data minibatches
seed corpus
branch k copies of seed LM parameters

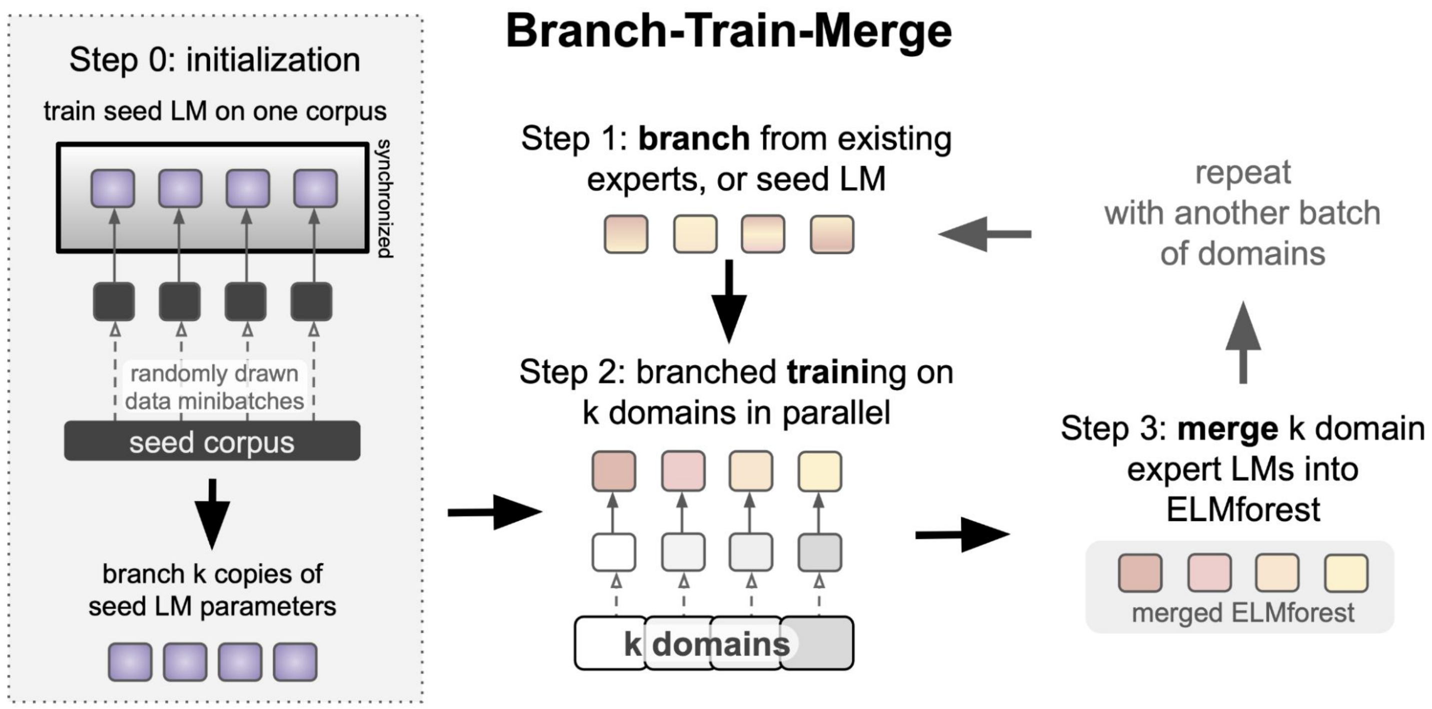
Branch-Train-Merge

Train the new experts on new data domain

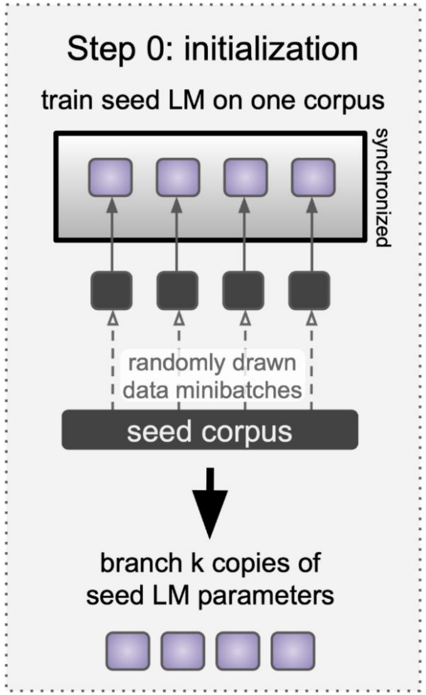
Step 2: branched **traini**ng on k domains in parallel



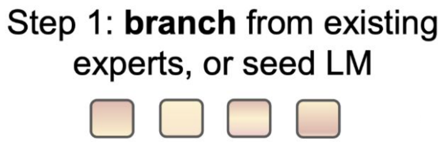
The Branch-Train-Merge Iteration


Branch-Train-Merge
Step 0: initialization
train seed LM on one corpus
synchronized
randomly drawn data minibatches
seed corpus
branch k copies of seed LM parameters
Step 1: **branch** from existing experts, or seed LM
Step 2: branched **training** on k domains in parallel
k domains
Step 3: **merge** k domain expert LMs into ELMforest
merged ELMforest
repeat with another batch of domains

The Branch-Train-Merge Iteration


Step 0: initialization
train seed LM on one corpus
synchronized
randomly drawn data minibatches
seed corpus
branch k copies of seed LM parameters

Branch-Train-Merge



weighted average the weights of existing exports according to the domain posterior on the new data

$$\underbrace{\theta_{k+1}}_{\text{new expert}} \leftarrow \sum_{i=0}^{k} \underbrace{w_i}_{\text{domain posterior}} \theta_i$$

Experiments

- Baselines:
    - DEMix
    - TRANSFORMER-LM: using distributed data parallelism to train one model on the whole dataset
- Dataset:
    - 16 domain dataset: Consists of 8 training and 8 evaluation domains
    - 80 domain dataset: Consists of 64 training and 16 evaluation domains
- All models use the GPT-3 architectures:
    - 125M (small), 350M (medium), 750M (large), 1.3B (xl)

	Domain	Corpus	# Train (Eval.) Tokens
TRAINING	1B	30M NewsWire sentences (Chelba et al., 2014)	700M (10M)
	CS	1.89M full-text CS papers from S2ORC (Lo et al., 2020)	4.5B (10M)
	LEGAL	2.22M U.S. court opinions, 1658 to 2018 (Caselaw Access Project)	10.5B (10M)
	MED	3.2M full-text medical papers from S2ORC (Lo et al., 2020)	9.5B (10M)
	WEBTEXT†	8M Web documents (Gokaslan and Cohen, 2019)	6.5B (10M)
	REALNEWS†	35M articles from REALNEWS (Zellers et al., 2019)	15B (10M)
	REDDIT	Reddit comments from pushshift.io (Baumgartner et al., 2020)	25B (10M)
	REVIEWS†	30M Amazon product reviews (Ni et al., 2019)	2.1B (10M)
		Total	**73.8B (80M)**

	Domain	Corpus	# Train (Eval.) Tokens
NOVEL	ACL PAPERS	1.5K NLP papers from ACL (Dasigi et al., 2021)	1M (1M)
	BREAKING NEWS†	20K latest articles from 400 English news sites (Baly et al., 2018)	11M (1M)
	CONTRACTS†	500 commercial legal contracts (Hendrycks et al., 2021)	1.5M (1M)
	CORD-19	400K excerpts from COVID-19 research papers (Wang et al., 2020)	60M (10M)
	GITHUB	230K public Github repository contents (Github Archive Project)	200M (10M)
	GUTENBERG	3.2M copyright-expired books (Project Gutenberg)	3B (10M)
	TWEETS†	1M English tweets from 2013-2018	8M (1M)
	YELP REVIEWS†	6M Yelp restaurant reviews (Yelp Reviews)	600M (10M)

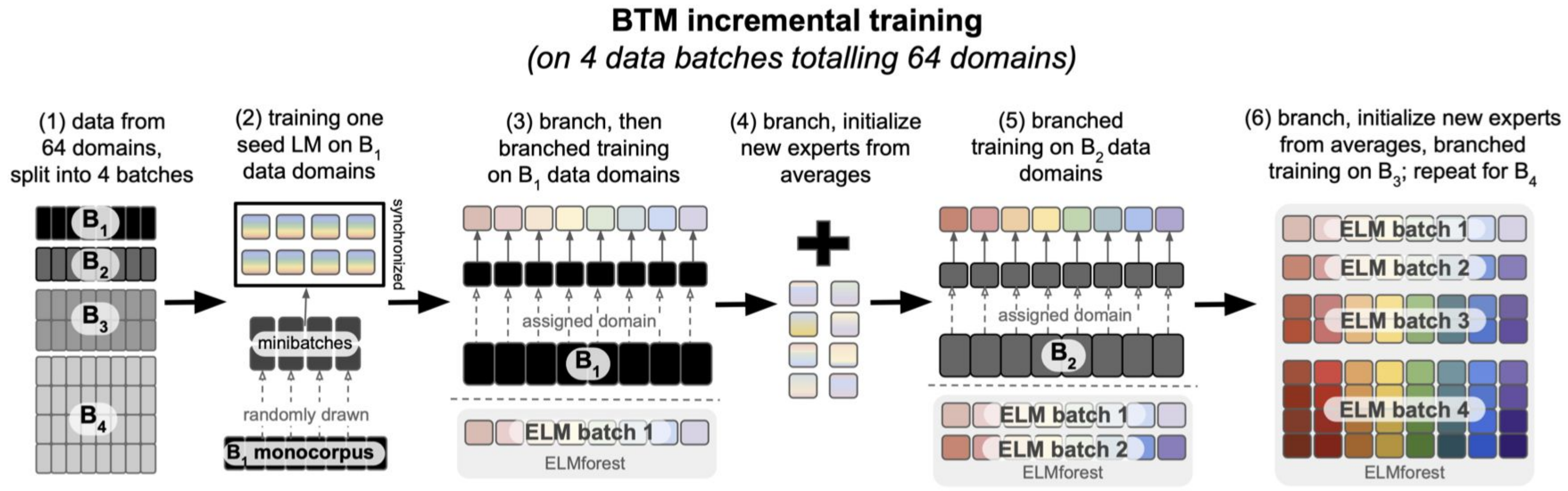
Table 1: Domains that make up our multi-domain training corpus, including the size of our training and evaluation (i.e. validation and test) data, in whitespace-separated tokens. † indicates datasets that we (partially) anonymize (§2). See Appendix §A.2 for more details on how these data were collected.

80-DOMAIN CORPUS: 192.3B WHITESPACE-SEPARATED TOKENS

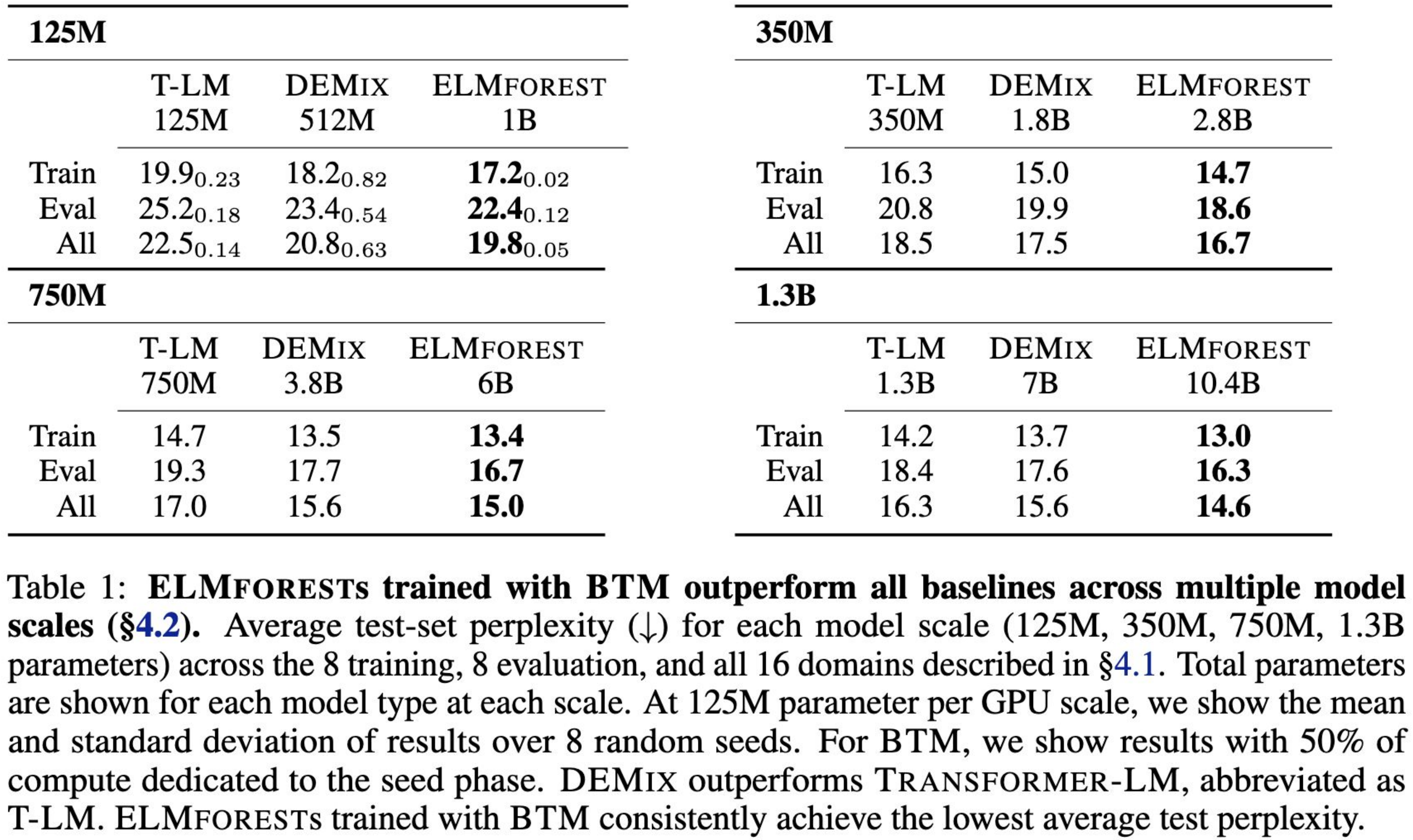
Category	**Domains**
SEMANTIC SCHOLAR (26.6%)	Medicine (5.2%), Biology (4.7%), CS (3.4%), Physics (2.7%), Math (2.3%), Unlabeled (1.3%), Psychology (1.2%), Chemistry (1.0%), Economics (0.8%), Engineering (0.7%), CORD19 (0.6%), Material Science (0.5%), Geology (0.5%), Sociology (0.5%), Business (0.3%), 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

Table 7: **Overview of the 80-domain corpus (§6.1).** The 80 domains that make up the multi-domain corpus we train and evaluate on, presented here in 8 descriptive categories for ease of inspection. For each of the 64 training domains, we include the percentage of the total number of tokens (in the entire corpus) comprising that domain. At the bottom, we include the 16 evaluation domains. All domains additionally include 1M tokens for validation and test data each. We include full details of each corpus in Appendix Table 10 and 11.

BTM Incremental Training


BTM incremental training
(on 4 data batches totalling 64 domains)
(1) data from 64 domains, split into 4 batches
B_1
B_2
B_3
B_4
(2) training one seed LM on B_1 data domains
synchronized
minibatches
randomly drawn
B_1 monocorpus
(3) branch, then branched training on B_1 data domains
assigned domain
B_1
ELM batch 1
ELMforest
(4) branch, initialize new experts from averages
(5) branched training on B_2 data domains
assigned domain
B_2
ELM batch 1
ELM batch 2
ELMforest
(6) branch, initialize new experts from averages, branched training on B_3; repeat for B_4
ELM batch 1
ELM batch 2
ELM batch 3
ELM batch 4
ELMforest

Experiments - Ensemble Comparison



125M

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

350M

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

750M

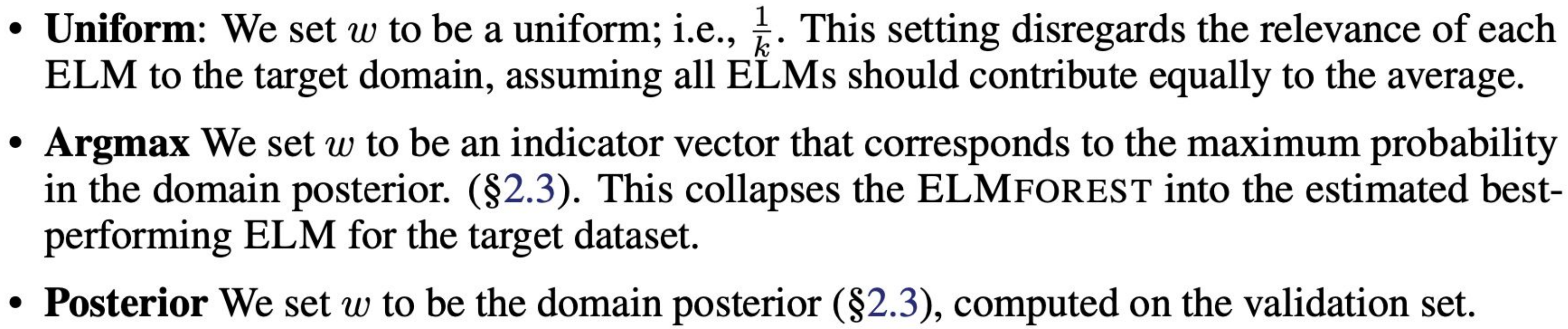
	T-LM 750M	DEMIX 3.8B	ELMFOREST 6B
Train	14.7	13.5	**13.4**
Eval	19.3	17.7	**16.7**
All	17.0	15.6	**15.0**

1.3B

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

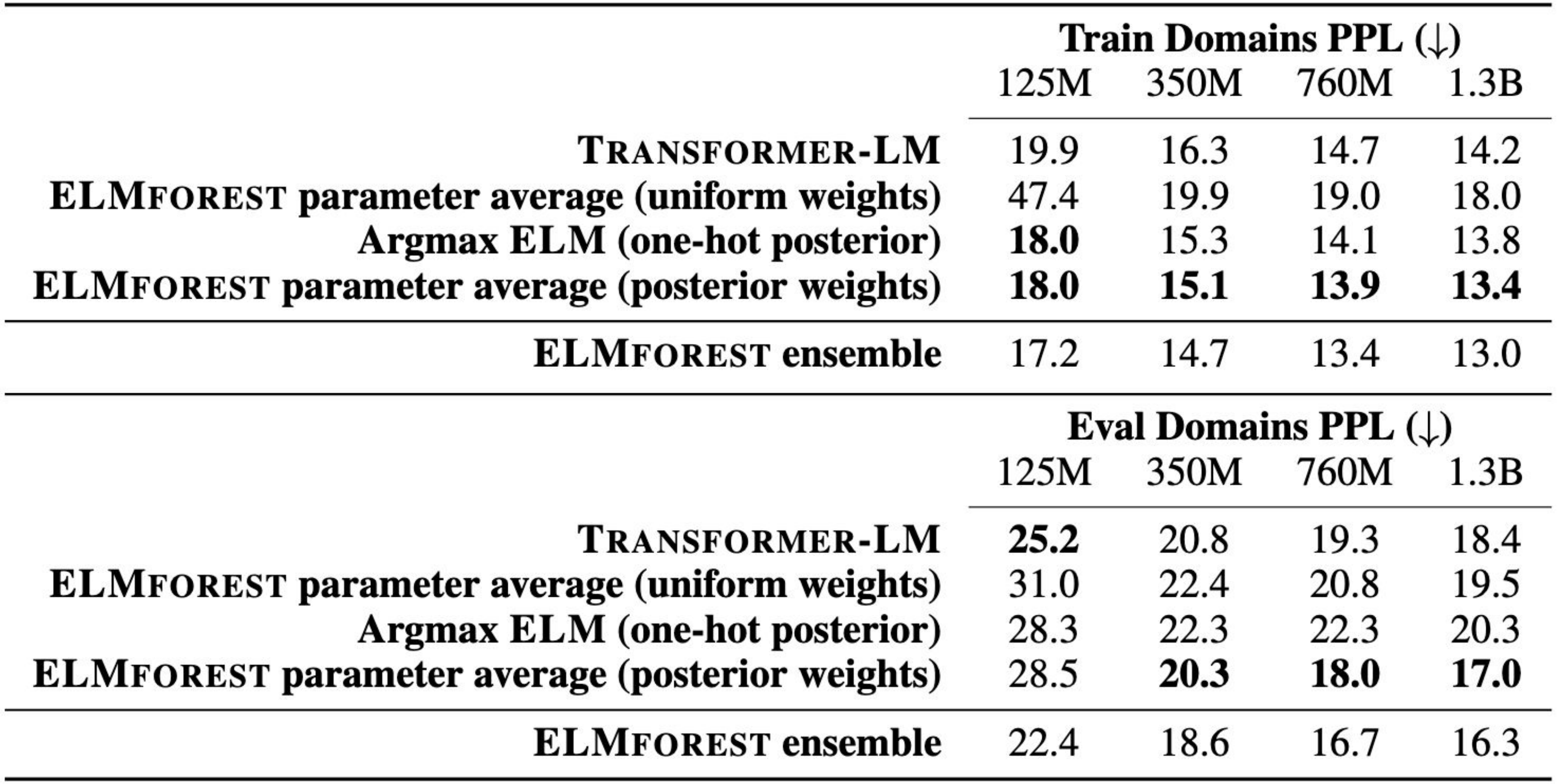
Table 1: **ELMFORESTs trained with BTM outperform all baselines across multiple model scales (§4.2).** Average test-set perplexity (↓) for each model scale (125M, 350M, 750M, 1.3B parameters) across the 8 training, 8 evaluation, and all 16 domains described in §4.1. Total parameters are shown for each model type at each scale. At 125M parameter per GPU scale, we show the mean and standard deviation of results over 8 random seeds. For BTM, we show results with 50% of compute dedicated to the seed phase. DEMIX outperforms TRANSFORMER-LM, abbreviated as T-LM. ELMFORESTs trained with BTM consistently achieve the lowest average test perplexity.

Parameter Averaging



- **Uniform**: We set w to be a uniform; i.e., $\frac{1}{k}$. This setting disregards the relevance of each ELM to the target domain, assuming all ELMs should contribute equally to the average.
- **Argmax** We set w to be an indicator vector that corresponds to the maximum probability in the domain posterior. (§2.3). This collapses the ELMFOREST into the estimated best-performing ELM for the target dataset.
- **Posterior** We set w to be the domain posterior (§2.3), computed on the validation set.

Parameter Averaging



	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

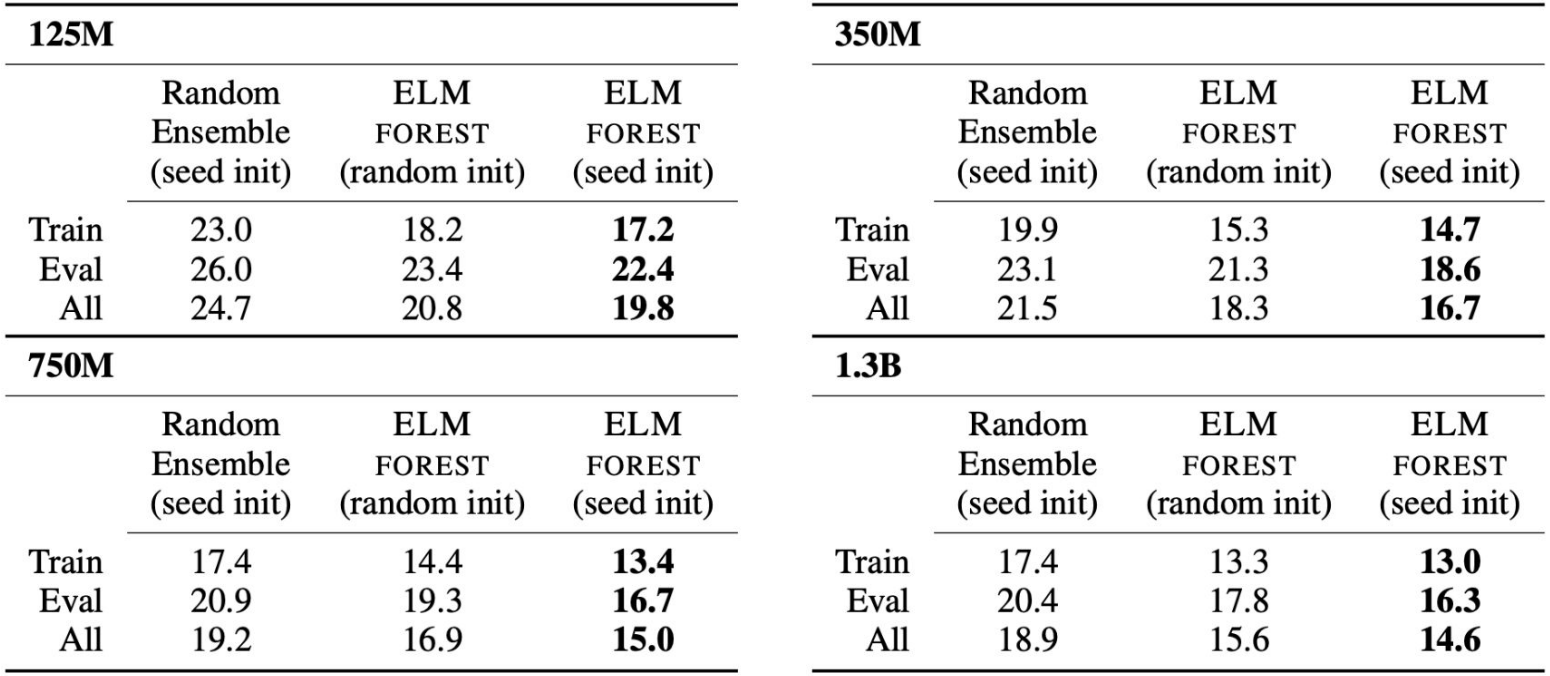
Different initialised models used for next experiments

Random Ensemble (seed init) A set of LMs trained on random data splits, to assess the effect of removing the domain specialization of ELMs. We first pool the training and development sets of our 8 train domains and divide into 8 random data splits, then execute the BTM procedure on those random splits, dedicating 50% of training to the seed phase

ELMFOREST (random init) An ELMFOREST trained with BTM where all ELMs are randomly initialized, to assess the effect of seed training. This is equivalent to setting the seed training compute budget to zero updates. We fix the random initialization across models.

ELMFOREST (seed init) The ELMFOREST setting of §4, which follows the BTM training procedure on the 8 train domains, and splits the compute budget such that 50% of the updates are dedicated to seed training and 50% to branched ELM training.

Importance of seed training in BTM and not just a collection of several random ensembles



125M

	Random Ensemble (seed init)	ELM FOREST (random init)	ELM FOREST (seed init)
Train	23.0	18.2	**17.2**
Eval	26.0	23.4	**22.4**
All	24.7	20.8	**19.8**

350M

	Random Ensemble (seed init)	ELM FOREST (random init)	ELM FOREST (seed init)
Train	19.9	15.3	**14.7**
Eval	23.1	21.3	**18.6**
All	21.5	18.3	**16.7**

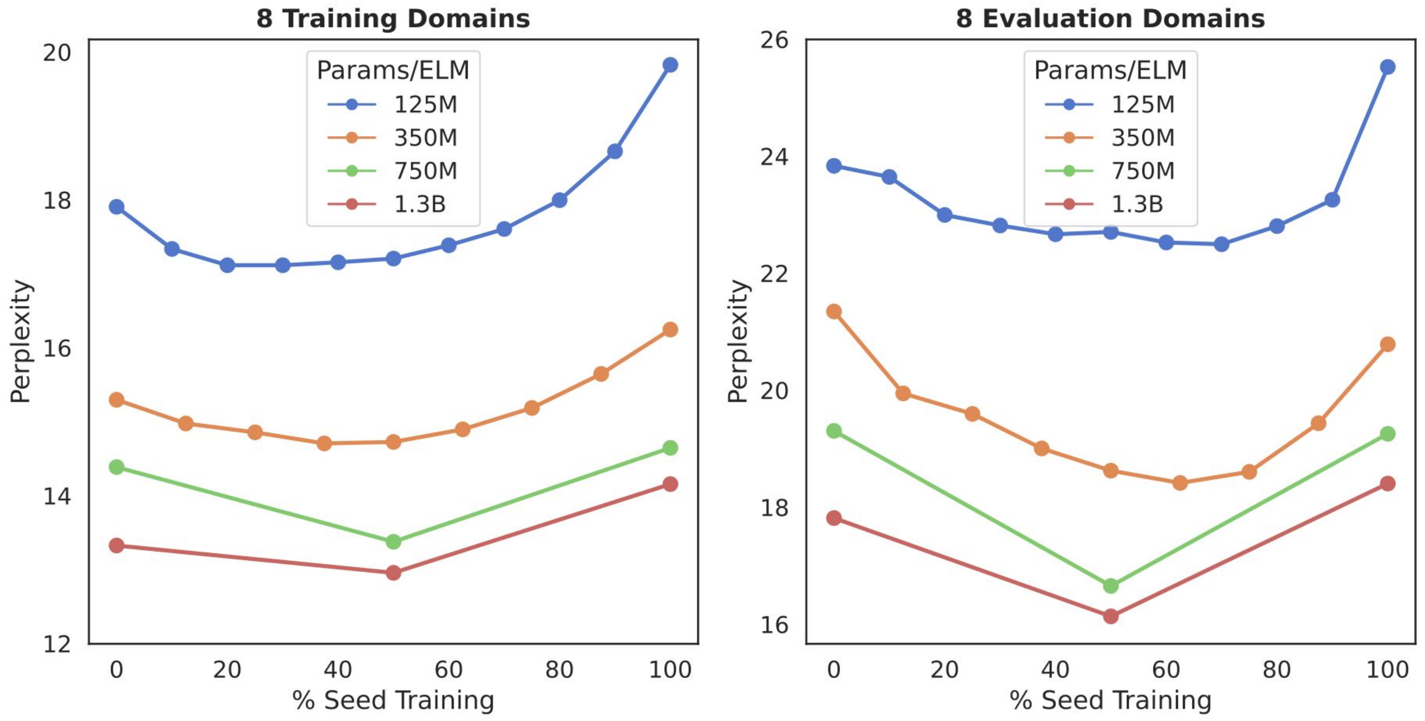
750M

	Random Ensemble (seed init)	ELM FOREST (random init)	ELM FOREST (seed init)
Train	17.4	14.4	**13.4**
Eval	20.9	19.3	**16.7**
All	19.2	16.9	**15.0**

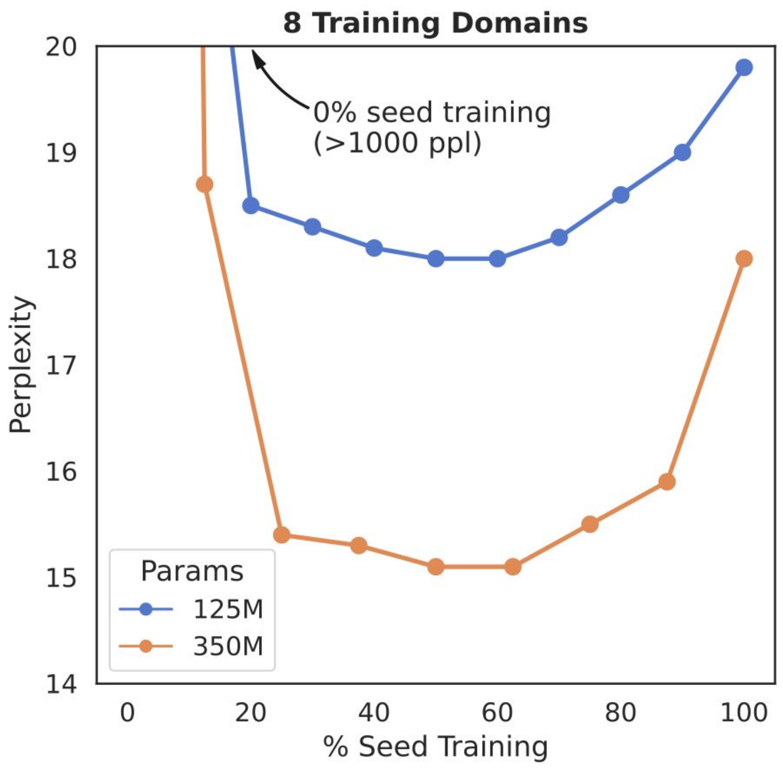
1.3B

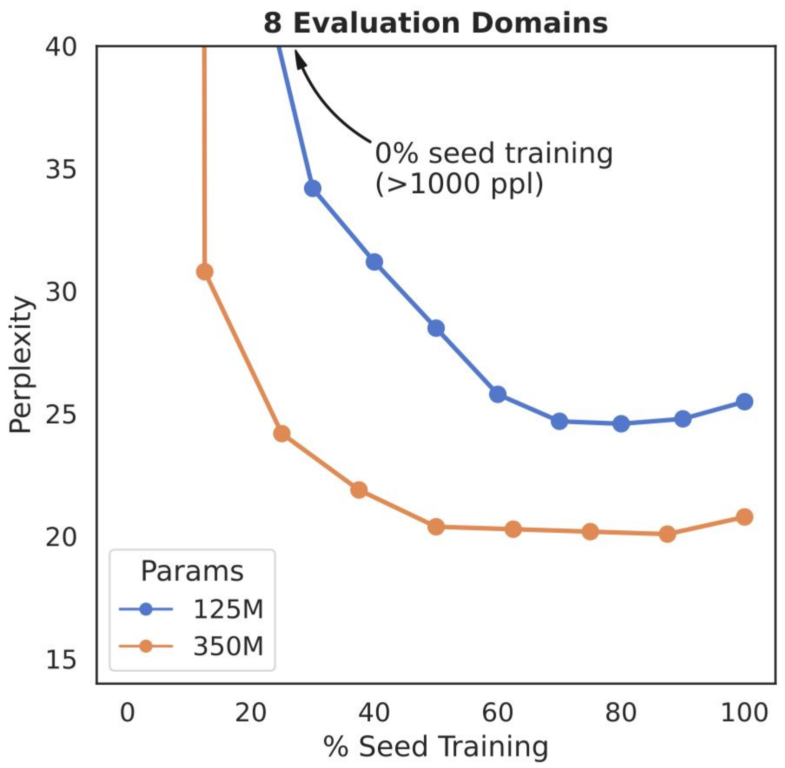
	Random Ensemble (seed init)	ELM FOREST (random init)	ELM FOREST (seed init)
Train	17.4	13.3	**13.0**
Eval	20.4	17.8	**16.3**
All	18.9	15.6	**14.6**

ELMForest Ensemble with different levels of seed Training

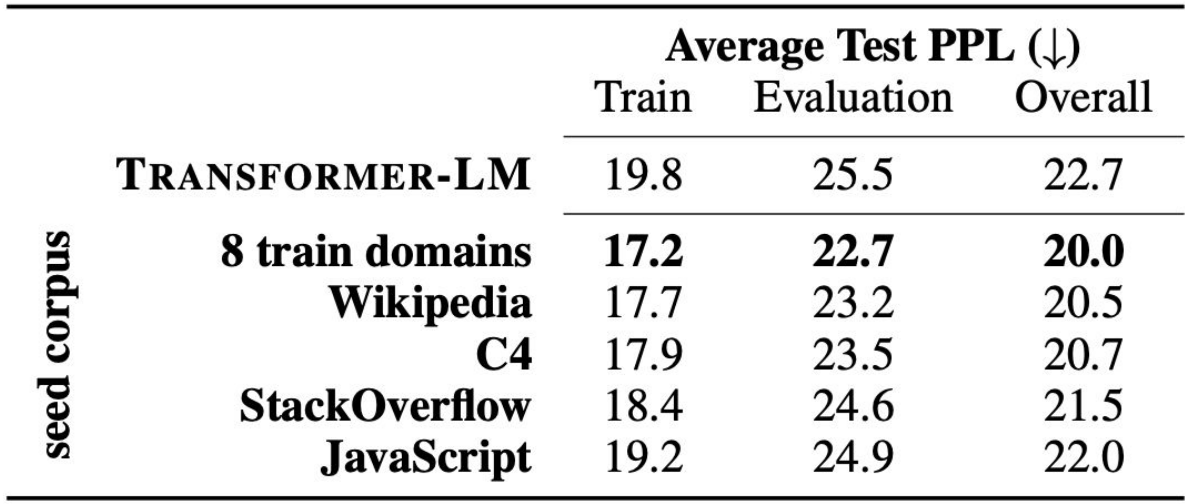


Seed Training with varying compute for parameter averaging





BTM is robust to seed training corpus



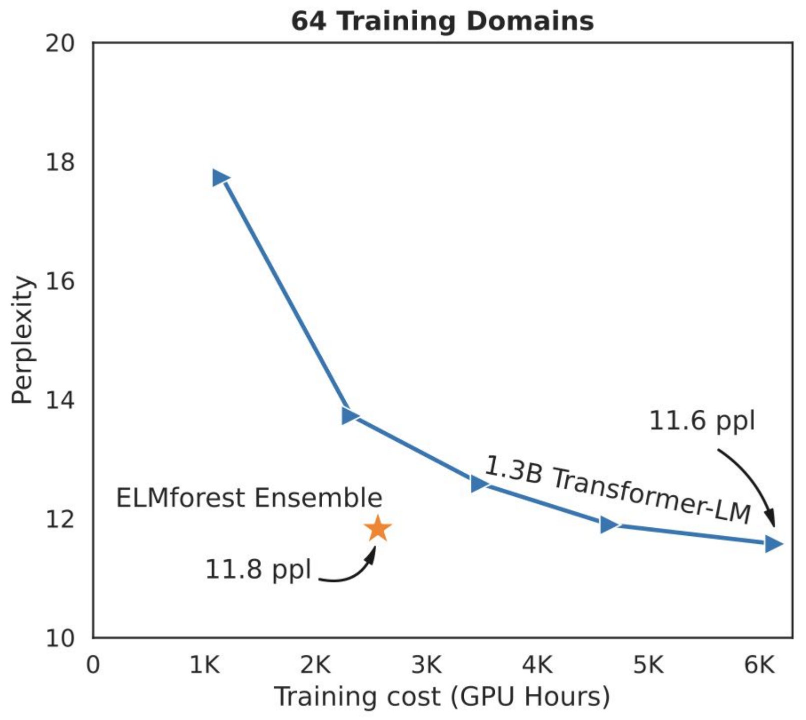
		Average Test PPL (↓)		
		Train	Evaluation	Overall
	Transformer-LM	19.8	25.5	22.7
seed corpus	**8 train domains**	**17.2**	**22.7**	**20.0**
	Wikipedia	17.7	23.2	20.5
	C4	17.9	23.5	20.7
	StackOverflow	18.4	24.6	21.5
	JavaScript	19.2	24.9	22.0

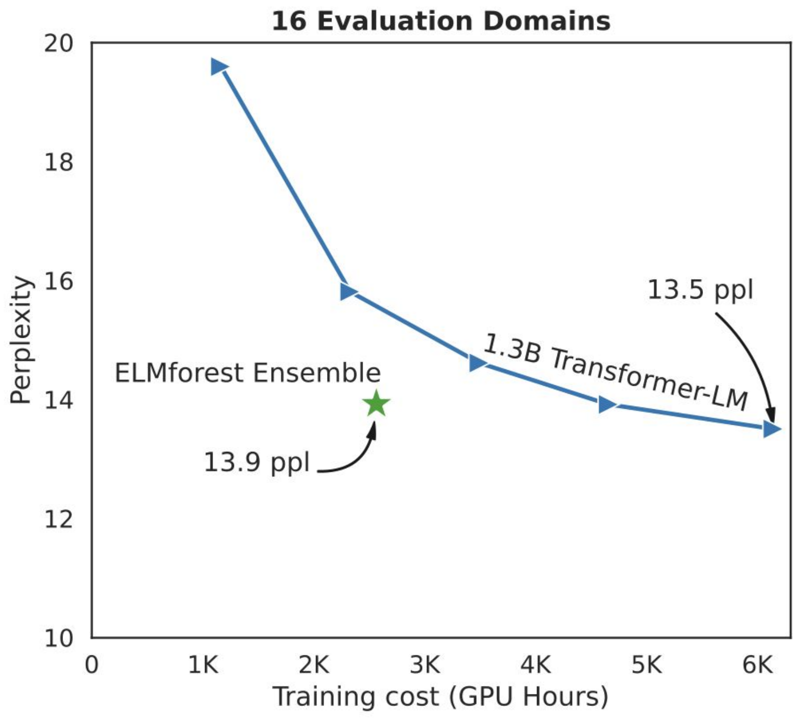
For next set of experiments

TRANSFORMER-LM - 1.3B parameter transformer LM, trained for 6144 GPU hours (with 128 GPUs) on all 64 domains

ELMFOREST - 40% of GPU Hour training compared to Transforner LM - 75% seed training and 25% branched training

Performance Comparison





Sparse Activation in Training/Evaluation

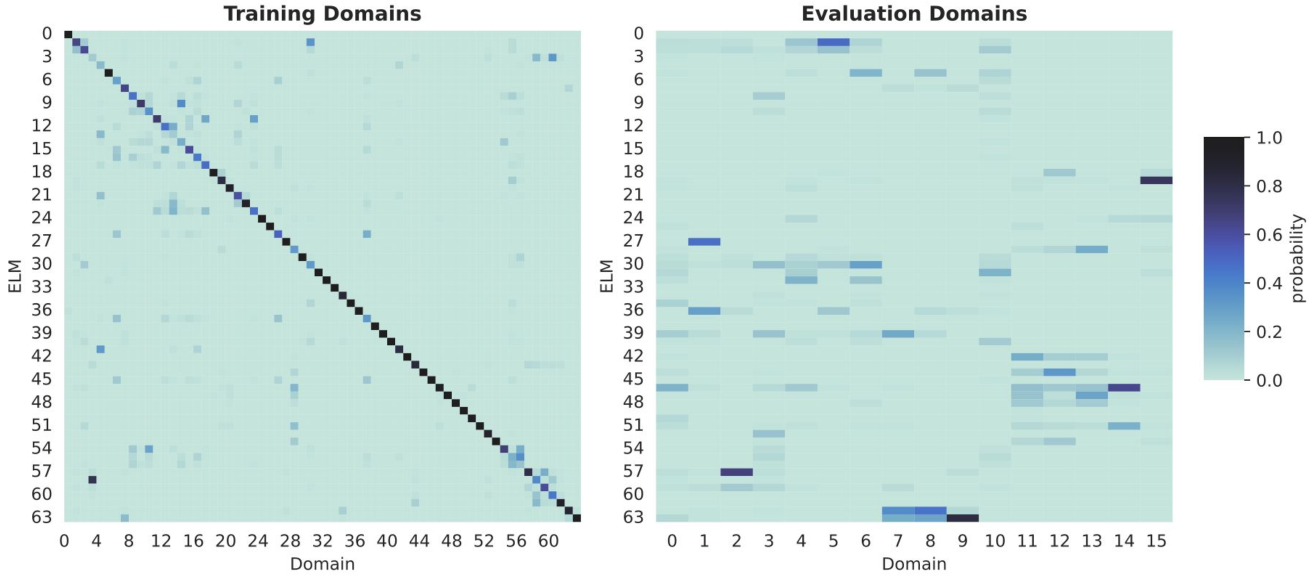
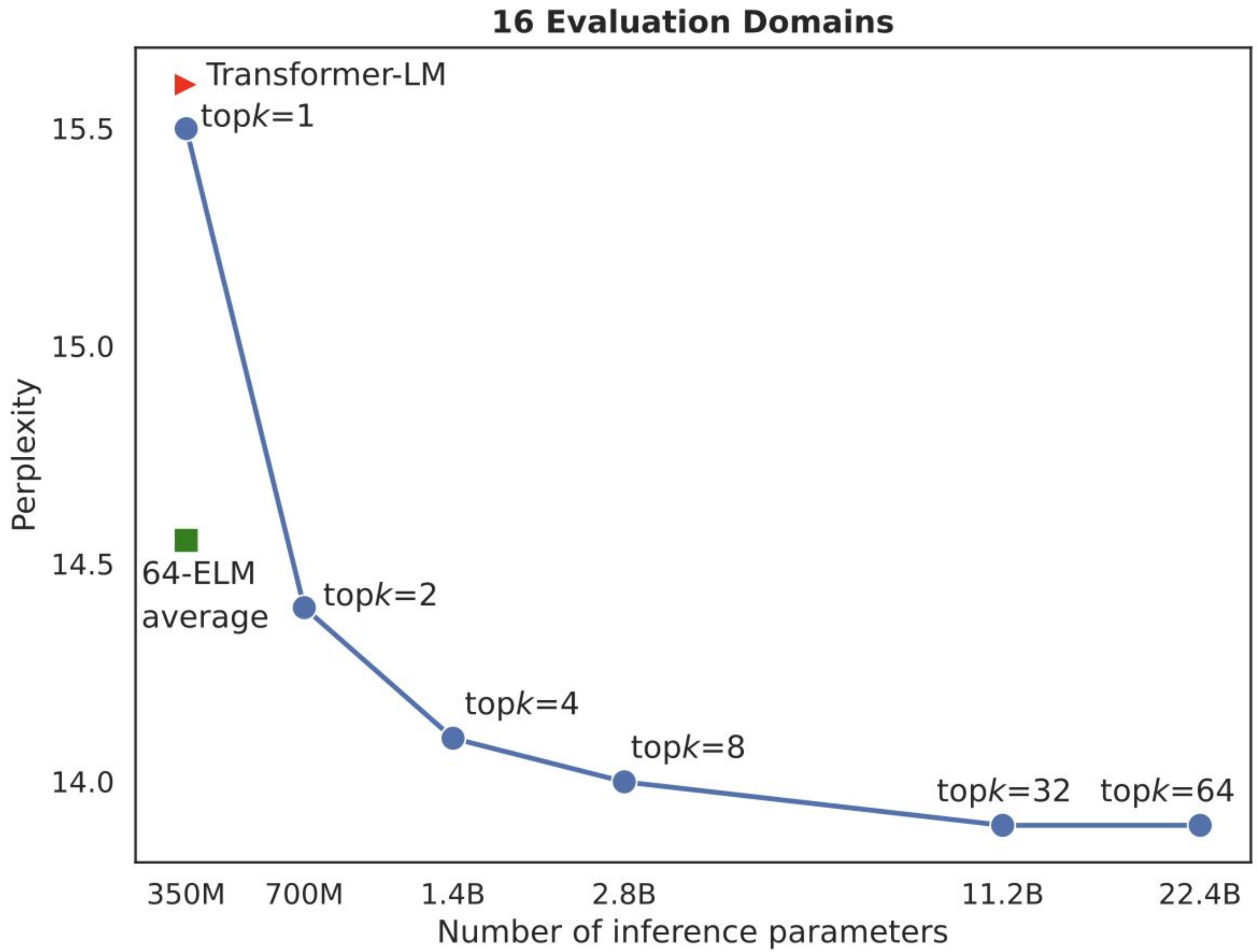


Figure 8: **Training and evaluation domain inference both use ELMs sparsely (§6.4).** Domain posterior visualization for 22.4B parameter ELMFOREST trained on 64 domains. ELM activation is extremely sparse for both training and evaluation domains.


16 Evaluation Domains
Transformer-LM
topk=1
topk=2
topk=4
topk=8
topk=32
topk=64
64-ELM average
Perplexity
15.5
15.0
14.5
14.0
350M
700M
1.4B
2.8B
11.2B
22.4B
Number of inference parameters