arXiv:2108.05036v2 [cs.CL] 20 Aug 2021

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
[◇]Facebook AI Research
Seattle, WA, USA
sg01@cs.washington.edu

Abstract

We introduce a new domain expert mixture (DEMIX) layer that enables conditioning a language model (LM) on the domain of the input text. A DEMIX layer is a collection of expert feedforward networks, each specialized to a domain, that makes the LM *modular*: experts can be mixed, added or removed after initial training. Extensive experiments with autoregressive transformer LMs (up to 1.3B parameters) show that DEMIX layers reduce test-time perplexity, increase training efficiency, and enable rapid adaptation with little overhead. We show that mixing experts during inference, using a parameter-free weighted ensemble, allows the model to better generalize to heterogeneous or unseen domains. We also show that experts can be added to iteratively incorporate new domains without forgetting older ones, and that experts can be removed to restrict access to unwanted domains, without additional training. Overall, these results demonstrate benefits of explicitly conditioning on textual domains during language modeling.

1 Introduction

Conventional language model (LM) training algorithms assume data homogeneity: all parameters are updated to minimize the loss on all of the data. We refer to this approach as *dense training*. Yet human language is as varied as human experience, a fact researchers often refer to obliquely when they use the term *domain* to describe distinct underlying subpopulations of the corpus. Dense training leaves variation in the data to be implicitly discovered (Aharoni and Goldberg, 2020), assuming that models will be able to fit all domains equally well.

While dense training is convenient, and densely trained LMs achieve impressive results (Brown et al., 2020), the approach has drawbacks with respect to generalization, efficiency, and flexibility. Even if training data is sourced from many domains, dense training can in practice emphasize subsets of the data in proportion to their ease of access (Oren et al., 2019; Fan et al., 2020), limiting generalization to less prevalent domains. Updating all parameters of the network gets substantially more expensive as model size grows (Strubell et al., 2019), making fine-tuning or domain-adaptive pretraining (DAPT; Gururangan et al., 2020) harder to perform with smaller computational budgets. It is also difficult to adapt to new domains without forgetting the original data (McCloskey and Cohen, 1989; Aghajanyan et al., 2021) or restrict access to certain domains the LM has been exposed to during training (e.g., those that contain hate speech; Bender et al. 2021), leading to risks of unwanted behavior (Gehman et al., 2020).

To address these limitations of dense training, we argue that LMs should be designed with *modularity*. We propose a modular LM that has components specialized to distinct domains in the training data, and can be customized at inference-time by mixing, adding, or removing these separated components as needed. This design principle emphasizes the ability to rapidly adapt the LM after training, a need that has been broadly advocated for language systems (Dinan et al., 2021; Lazaridou et al., 2021).

We introduce modularity into an LM with a new domain expert (DEMIX) layer that explicitly conditions the LM on the domain of the input text (when it is known), or estimates the input domain during inference (when it is not known). A DEMIX layer is a drop-in substitute for a feedforward layer in a transformer LM (e.g., GPT-3), creating a specialized version of the layer (or *expert*) per domain (see Figure 1; §3).[1] We find that replacing every feed-

[1]This is an example of conditional computation (Fedus et al., 2021; Lepikhin et al., 2020; Lewis et al., 2021; Roller et al., 2021), which follow prior literature on mixture of experts (Jacobs et al., 1991; Shazeer et al., 2017). Unlike dense training, conditional computation activates different parameters for different inputs. Instead of learning how to route data to experts, the DEMIX layer routing mechanism follows from a natural, observable segmentation of the data.

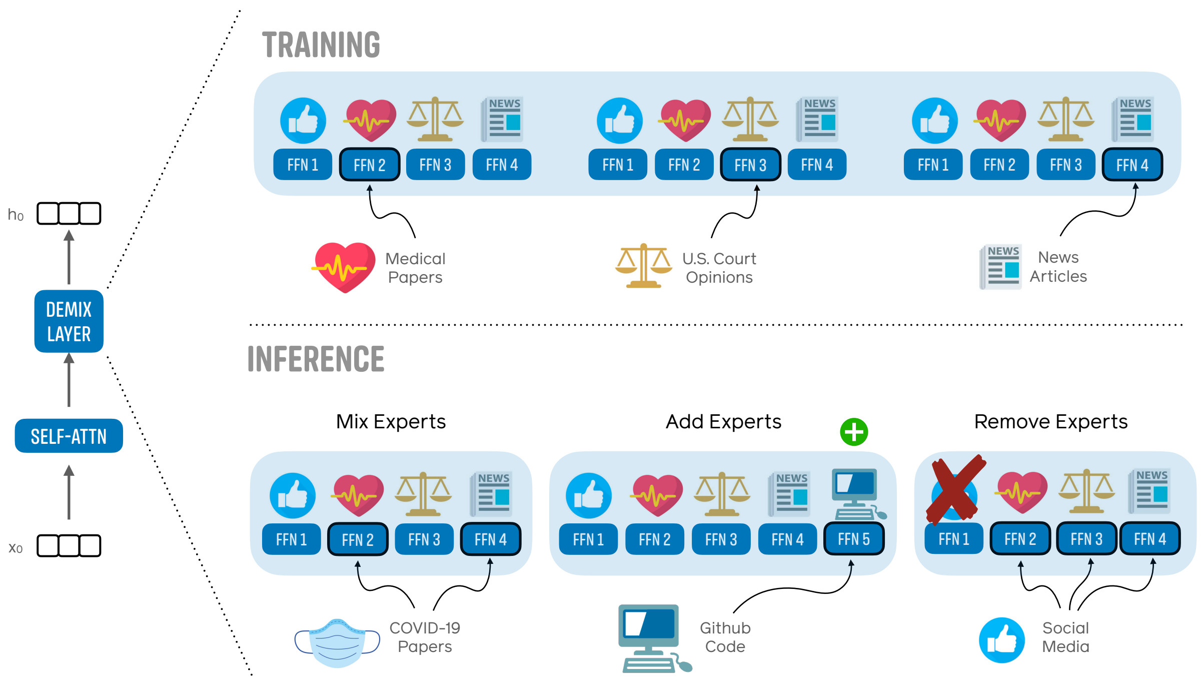


Figure 1: Illustration of a DEMIX layer in a single transformer block. During training, expert feedforward networks are conditionally activated based on the domain (here, document provenance) of the input sequence (i.e., scientific papers or court opinions). At inference time, the language model has new modular functions: domain experts can be mixed to handle heterogeneous domains, added to adapt to novel domains, or removed to "forget" unwanted domains. Image attribution: news icon from emojipedia.org; all other icons from istockphoto.com.

forward layer in the transformer with a DEMIX layer offers new affordances for modularity, addressing the challenges above, while improving performance in both training domains and novel test-time domains.

room for improvement

Although the concept of a domain lacks a rigorous definition in NLP, we use coarse provenance categories (e.g., whether a document is a medical research paper or a Reddit post) as a conditioning variable when training an LM with DEMIX layers (§2). Training on data from eight different domains, we find that DEMIX layers consistently improve in-domain performance (§4). However, because these categories may not be an optimal segmentation of the training data, or may lack coverage of test-time domains, naively selecting a single domain expert at test time can hurt generalization. Instead, we introduce a parameter-free probabilistic approach to dynamically estimate a *weighted mixture* of domains during inference (§5). Mixing experts improves DEMIX performance not only on *novel* test-time domains, but also on test data from the *training* domains, which may themselves be heterogeneous. Our results suggest that introducing modularity into an LM need not come at a cost to generalization performance.

Because DEMIX forces experts to specialize to domains, the overall model can be (partially) disentangled after training. Beyond mixing, we can add (§6) or remove (§7) domain experts, resulting in predictable changes in model behavior at inference time: adding experts allows for model adaptation without updating all parameters (hence avoiding forgetting), and removing experts allows for simulating the removal of training domains without additional training. Overall, DEMIX layers demonstrate benefits of explicitly conditioning on textual domains during language modeling, and our results suggest that these benefits persist at scale. Our code is publicly available.[2]

2 Multi-Domain Corpus

We center this study around a large, multi-domain corpus we constructed with explicit provenance metadata (Table 1). While other multi-domain corpora (Koh et al., 2021; Gao et al., 2020) cover many more domains and tasks, the corpus we introduce contains substantial metadata-tagged text for language modeling, as well as datasets with friendly licensing to support reproducibility.

[2] http://github.com/kernelmachine/demix

	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, 2018)	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). Reddit was extracted and obtained by a third party and made available on `pushshift.io`, and was anonymized by Xu et al. (2020); we use their version. See Appendix §A.1 for more details on how these data were collected.

2.1 Document Provenance as a Domain Label

While a growing body of work has attempted to address the structure and composition of language domains (Eisenstein et al., 2014; Plank, 2016; Aharoni and Goldberg, 2020; Gururangan et al., 2020), fundamentally what a domain is remains a matter of debate. In this work, we focus on the *provenance* of a document, operationalized coarsely by the dataset we used to access it, which approximates a social process that produced it. Defining domains this way is easy and intuitive, conveys a great deal about the variation in a document's language, and aligns with common practice in NLP research. However, other accounts of variation in language (e.g., Lucy and Bamman, 2021), and richer notions of relationships among domains (e.g., hierarchies; Gururangan et al., 2020), may be studied in future work.

Scaling Expert Language Models with Unsupervised Domain Discovery

2.2 Corpus Description

The multi-domain corpus we use in this study consists of two parts. The first is a collection of **training** domains: text from eight domains of largely English text, listed at the top of Table 1, each of which vary in complexity and coverage and has been the subject of study in NLP.[3]

The second part is a collection of **novel** domains: text from eight domains also of largely English text, listed at the bottom of Table 1, which may or may not align with the training domains. The novel domains allow us to measure how models generalize to a more challenging data distribution shift, where domain boundaries may be less clear.

See Appendix §A.1 for more details on how these data were collected. To support future work with the data, we also release a standard API to download and preprocess it into a format compatible with Fairseq (Ott et al., 2019).[4] We replace user identifiable information (e.g., email addresses, user handles, social security numbers, credit card numbers, phone numbers) with dummy tokens.[5]

3 DEMix Layer

3.1 Background: Mixture-of-Experts Transformers

The transformer architecture is comprised of interleaved multi-head self-attention, layer-norms, and

[3]The metadata for each document includes at least its provenance, and in some cases more information (e.g., URLs, publication venue, or legal jurisdiction). Future work might explore more fine-grained notions of domain.

[4]`https://github.com/kernelmachine/demix-data`

[5]While it is difficult to anonymize data perfectly, especially at scale, we use a suite of regexes to identify commonly occurring identifiable information on the Internet. See Appendix §A.2 for more details.

feedforward networks (Vaswani et al., 2017). Each of these layers produces a vector representation for each of the input tokens. Our focus is on the feedforward component:

$$\mathbf{h}_{t,\ell} = \mathrm{FFN}(\mathbf{h}_{t,\ell-1}), \tag{1}$$

where $\mathbf{h}_{t,\ell}$ is the vector for the tth token produced by layer ℓ.

Shazeer et al. (2017) propose a formulation of one or more feedforward layers as an ensemble of n experts $\mathrm{FFN}_1, \ldots, \mathrm{FFN}_n$, assigned weights respectively by functions $g_1, \ldots, g_n$:

$$\mathrm{FFN}(\mathbf{h}_{t,\ell-1}) = \sum_{j=1}^{n} g_j(\mathbf{h}_{t,\ell-1}) \cdot \mathrm{FFN}_j(\mathbf{h}_{t,\ell-1}) \tag{2}$$

The g function routes tokens to different experts, usually each a separate instance of the original feedforward network. If g routes to a single expert, then the computational cost (in floating-point operations; FLOPs) will be same as the original feedforward network, even though it has slightly more than n times as many parameters.

3.2 DEMIX Routing

Previous approaches *learn* the weighting functions g at a token-level, and either assign at most one (Fedus et al., 2021) or two (Lepikhin et al., 2020) experts per token. This necessitates load balancing and other techniques to encourage the model to use all experts instead of relying on just a few (Fedus et al., 2021; Lewis et al., 2021).

Domain-level routing. Has implications on expert's specialization

We instead use domain metadata provided with training documents to route data to experts at the *document* (i.e., sequence) level. During training, every token in the same sequence is assigned to the same expert based on the domain label.

Let $\mathcal{D}$ denote the set of domain labels (i.e., the eight labels in Table 1). If we index the experts by $\mathcal{D}$ and $d \in \mathcal{D}$ is the domain label for the current training instance, then

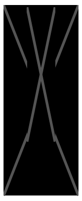
$$g_j(\mathbf{h}_{t,\ell}) = \begin{cases} 1 & \text{if } j = d \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

While we assume that each *training* document is associated with a single domain label, we relax this requirement at inference time (§5), which improves model performance in mixed and unknown domain scenarios.

3.3 DEMIX Architecture

Our design results in one expert in a DEMIX layer per domain (i.e., eight experts for eight training domains in our multi-domain corpus).

We replace *every* feedforward layer in the transformer with a DEMIX layer, in contrast to previous work (Fedus et al., 2021; Lepikhin et al., 2020) that interleaves shared and expert layers. Preliminary experiments showed that interleaving led to worse in-domain performance with DEMIX layers. We hypothesize that shared layers may serve as a bottleneck to find shared features between domains, and may impact performance adversely when training domains are highly different from one another.[6] Future work might perform careful comparisons of different architectural choices.



In this study, each expert FFN_j is a two-layer MLP with the same dimensions as the original FFN layer of the transformer. As with other conditional computation models (Fedus et al., 2021; Lepikhin et al., 2020), this means that the effective number of parameters in the overall DEMIX LM increases (Table 2). While this incurs memory costs, the computational budget we consider in this study centers around runtime costs. DEMIX layers decrease the runtime costs of training the LM.

Memory costs, # of total parameters

Training costs

Inference cost, # of active parameters

3.4 DEMIX Training

DEMIX layers increase the total parameters of the LM while also reducing GPU latency costs during training, effectively reducing runtime costs of training the LM.

DENSE training (also referred to as *data-parallel*) is usually implemented by copying model parameters to every GPU, feeding a different mini-batch of shuffled data to each GPU, computing a stochastic gradient for each mini-batch, and updating all parameters synchronously with the average stochastic gradient from across all GPUs.

To train an LM with DEMIX layers, we instead partition the GPUs among the domains, so that each GPU is assigned a single domain (along with its corresponding expert). During training, we fill a mini-batch with k sequences, where each sequence represents data from a particular domain, and we send each mini-batch to its dedicated domain expert. We use larger batch sizes by performing data-parallel training between expert parameters

[6]Indeed, preliminary experiments suggest that interleaving expert layers causes large performance hits in the most distinct domains, i.e., those with lower vocabulary overlap with other domains in the corpus.

on GPUs assigned to the same domain; we assign n/8 GPUs to each domain (Table 2). To reduce overfitting, we ensure that each of these n/8 GPUs is assigned to different shards of their domain's training data.

We compare the training efficiency of DENSE and DEMIX models up to 1.3B parameters per GPU in Table 2. Compared to DENSE LMs, DEMIX layers achieve the same or slightly higher throughput (measured in TFLOPs/GPU) for the same total FLOPs per update, despite adding significantly more parameters.

DEMIX achieves higher throughput because we only synchronize expert parameters allocated to the same domain.[7] As we increase model size, this results in a reduction of latency costs between GPUs, and hence, faster training; instead of synchronizing parameters over n GPUs, we perform eight synchronizations over n/8 GPUs.[8]

Assumes enough data is available and that loads are balanced

In this work, we assume that there is sufficient data for each training domain that each expert can be exposed to the same amount of data, and load balancing between experts is not necessary. Future work may consider how varying the amount of data per domain influences absolute and relative performance across domains, especially in the long tail of rare domains.

While the total number of parameters of DEMIX LMs are substantially larger than their DENSE counterparts, since the practical training costs are essentially the same, we compare baselines in all subsequent experiments based on parameters *per GPU*, as we do in Table 2.

4 In-Domain Performance

The first set of experiments in this study considers the impact of replacing the conventional feedforward layers in a transformer LM with DEMIX layers. We run all experiments in this section with the training domains (Table 1).

4.1 Experimental Setup

Architecture and Input The model architecture is a randomly-initialized LM with the GPT-3 (Brown et al., 2020) architecture implemented in Fairseq (Ott et al., 2019). We experiment with multiple architectures (i.e., those of GPT-3 small, medium, large, and XL), at a maximum size of about 1.3B parameters per GPU. We use the GPT-2 (Radford et al., 2019) vocabulary of 50,264 BPE types, and train with 1,024-token sequences, with cross-document boundaries. Each document has a beginning-of-sentence token prepended to it.

[7]Shared parameters are synchronized across all GPUs.

[8]While this technique reduces latency costs, the bandwidth costs are the same between DEMIX and DENSE models.

		Parameters per GPU			
		125M	350M	760M	1.3B
DENSE	**GPUs**	32	64	128	128
	Total Experts	0	0	0	0
	GPUs/expert	0	0	0	0
	Total params	125M	350M	760M	1.3B
	TFLOPs/update	556	3279	13,637	23,250
	TFLOPs/GPU	**31**	**37**	45	51
DEMIX	**GPUs**	32	64	128	128
	Total Experts	8	8	8	8
	GPUs/expert	4	8	16	16
	Total params	512M	1.8B	3.8B	7.0B
	TFLOPs/update	556	3279	13,637	23,250
	TFLOPs/GPU	**31**	**37**	**48**	**55**

Table 2: Our specifications for training DENSE and DEMIX LMs. All models are trained for about 48 hours on V100 GPUs. DEMIX layers increase the total parameters of the LM while maintaining (or increasing) throughput, measured in TFLOPs/GPU. We use the formula described in Narayanan et al. (2021) to calculate these metrics. See Appendix §A.3 for more details.

Hyperparameters We set the total number of training steps based on this allocated runtime, set 8% of these steps to be warm-up, and use the Adam optimizer (Kingma and Ba, 2017) with a polynomial learning rate decay. Learning rates are tuned for each model separately over {0.0001, 0.0003, 0.0005}, taking the fastest learning rate that avoids divergence. Each worker processes two sequences of length 1,024, and gradients are accumulated over 8 updates. We clip gradients if their L_2 norm exceeds 0.1. See Appendix §A.4 for more details. These settings are inspired by Lewis et al. (2021).

Computational Budget We follow previous work in using runtime as the primary computational budget, which provides a better comparison of the practical costs of training conditional compute and dense models (Lewis et al., 2021). We assume a fixed budget of about 48 hours on NVIDIA V100 32GB GPUs. We display the number of GPUs used for each model size in Table 2; we chose these GPU budgets because larger models require more compute to train properly (Lewis et al., 2021; Kaplan et al., 2020), and found these GPU budgets to result in stable training for each model size given mostly fixed hyperparameters.

	Parameters per GPU			
	125M	350M	760M	1.3B
DENSE	20.6	16.5	14.5	13.8
DENSE (Balanced)	19.9	15.8	14.3	13.6
+DOMAIN-TOKEN	19.2	15.9	14.3	**13.4**
DEMIX (naive)	18.4	15.5	14.2	13.8
DEMIX (cached; §5.4)	**17.8**	**14.7**	**13.9**	**13.4**

Table 3: Average of in-domain test-set perplexity. We discuss the last row in §5.4.

Evaluation We report test-set perplexities after about 48 hours of training. In all tables, we report each result with respect to a set number of parameters per GPU, as in Table 2. As mentioned in §3.4, DEMIX LM will have a larger effective size than the DENSE LM at the same increased throughput.

4.2 Compared Models

DENSE The first baseline is a DENSE model that treats the data as homogeneous, i.e., it shares all parameters across all domains. Under this setup, the language model parameters are copied across all GPUs, and gradients computed during training are all-reduced across every GPU. There is no explicit conditioning on domain.

DENSE (Balanced) Under this setting, we train densely but ensure that the model is exposed to an equal amount of data from each domain. While there is still no explicit conditioning on domain, the gradient updates that the model makes during training are an average of those computed across all domains represented in a batch.

+DOMAIN-TOKEN This model is trained identically to DENSE (Balanced), but we prepend a token indicating the sequence's domain to every sequence block (during training and test time). A variant of this domain token is explored in some previous studies (Zellers et al., 2019; Keskar et al., 2019). This baseline provides domain information to the language model in the form of input supervision. We ignore the domain token when computing perplexity during evaluation.

Oracle that provides the domain of the input

DEMIX (naive) We replace every feedforward layer in the transformer with a DEMIX layer, as detailed in §3. Under this setting, the domain of the test data is known and revealed to the model (e.g., the CS expert is used for CS test data), which we refer to as *naive*. We also ensure that the model is exposed to an equal amount of data from each domain.

	1.3B parameters per GPU		
Domain	DENSE	DEMIX (naive)	DEMIX (cached prior; §5.4)
1B	11.8	11.5	**11.3**
CS	13.5	12.2	**12.1**
LEGAL	6.8	**6.7**	**6.7**
MED	9.5	9.2	**9.1**
WEBTEXT	**13.8**	14.6	14.3
REALNEWS	**12.5**	13.3	13.1
REDDIT	28.4	30.6	**28.1**
REVIEWS	14.0	12.6	**12.5**
Average	13.8	13.8	**13.4**

Table 4: Test-set perplexity by domain, for an LM with 1.3B parameters per GPU. We discuss the last column in §5.4.

4.3 Results

Table 3 shows test-set perplexities, averaged across the eight training domains. First, we observe that domain balancing is consistently helpful for DENSE training. We find that balancing is especially important in cases in which there is an imbalance of domain prevalence, confirming similar observations from previous studies (Arivazhagan et al., 2019).[9]

Next, we observe that the benefits of additional domain information (i.e, domain tokens or DEMIX layers) are clearest for the smallest model; for larger models, the benefits are smaller but consistent. This result suggests that domain-specific information enables the model to better specialize to different domains in its training data. However, as the model size grows, the DENSE baseline becomes increasingly better at fitting the training domains, catching up to models with additional domain information, in the average case.

As the model grows, it gains the ability to identify the input's domain

4.4 Domain Hetereogeneity

A more complete view of the experiments with the largest model is shown in Table 4. We see that even at scale, most training domains benefit from DEMIX layers in a naive setting (where the domain label is revealed at test time), but some do not; WEBTEXT, REALNEWS, and REDDIT fare worse than the DENSE baseline. We believe that this variation can be explained by heterogeneity within domains and varying degrees of similarity between them. DENSE training may be advantageous for

[9]Balancing improves performance on most domains, but hurts performance relative to a DENSE baseline on the REDDIT domain (Appendix §A.5). In the multi-domain corpus, there is far more REDDIT text than anything else; see Table 1.

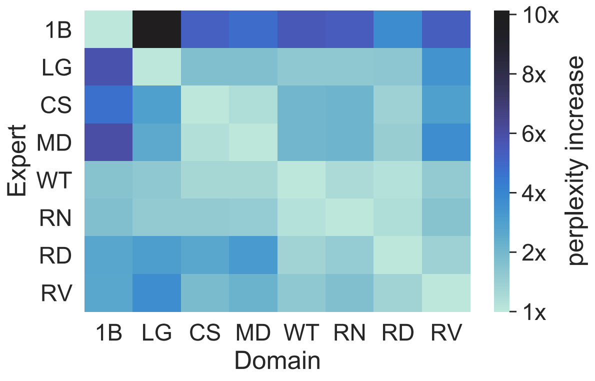


Figure 2: Domain experts in DEMIX specialize to their domain. We compute the above heatmap with a DEMIX LM with 1.3B parameters per GPU. Each cell of the heatmap is a ratio between an expert's test perplexity on a domain to that of the expert trained on that domain. The diagonal indicates that each expert has the best performance on its assigned domain. While some experts (e.g., 1B, MED) do not transfer well to most domains in the training corpus, WEBTEXT and REALNEWS experts transfer much better, confirming their heterogeneity. Key: LG $\to$ LEGAL, MD $\to$ Med, WT $\to$ WEBTEXT, RN $\to$ REALNEWS, RD $\to$ REDDIT, RV $\to$ REVIEWS.

domains that have a higher degree of overlap with other domains in the corpus (and therefore, benefit from parameter sharing).

To provide further evidence for this explanation, we measure the hetereogeneity of domains in the multi-domain corpus, according to a DEMIX LM. We plot a matrix of the perplexity changes across all domain experts in Figure 2, comparing all experts against the expert explicitly trained for each domain. As the perplexity change tends lower, the corresponding expert has higher affinity to the target domain.

First, we observe that domain experts have the highest affinity to their assigned domain, indicating that they do specialize. We also observe that some experts, e.g., WEBTEXT, REALNEWS, and REDDIT, have relatively high affinities to many domains, suggesting that these domains are heterogeneous. Separately we observe that an expert's affinity to a domain correlates positively with bigram overlap between the expert domain and target domain (r=0.40, t=3.45, p=0.001). This further suggests that similar domains have more closely aligned domain experts.

These findings suggest that a discrete notion of domain, while usually helpful on average (in our artificially constructed population of eight training domains), is too rigid. In the next section, we introduce new ways of softening Equation 3 into a mixture over domain experts, to improve performance on heterogeneous domains.

5 Mixing Experts at Inference Time

The previous section establishes that incorporating DEMIX layers improves LM performance on test data from *known* training domains. At inference time, the domain label was revealed to the model and used to select an expert within each DEMIX layer. In practice, however, text may not come with a domain label, may straddle multiple domains, or may not belong to any of the domains constructed at training time; the provenance of the data may even be unknown.

In these cases, rather than a hard choice among experts (Equation 3), we propose to treat $g_1, \ldots, g_n$ as mixture coefficients, transforming the domain membership of an input text into a matter of probabilistic belief. Unlike previously proposed mixture-of-experts formulations (Shazeer et al., 2017; Lepikhin et al., 2020), this approach introduces no new parameters and the weights are computed only at test time.[10]

To analyze inference-time behavior in mixed or unknown domain scenarios, we turn to the corpus of novel domains in the multi-domain corpus (Table 1). As mentioned in §2, these domains have fuzzier boundaries, compared to the training domains.

5.1 Dynamically Estimating Domain Membership

Consider the probabilistic view of language modeling, where we estimate $p(X_t \mid \boldsymbol{x}_{<t})$. We introduce a domain variable, D_t, alongside each word. We assume that this hidden variable depends on the history, $\boldsymbol{x}_{<t}$, so that:

$$p(X_t \mid \boldsymbol{x}_{<t}) = \sum_{j=1}^{n} p(X_t \mid \boldsymbol{x}_{<t}, D_t = j) \cdot \underbrace{p(D_t = j \mid \boldsymbol{x}_t)}_{g_j} \tag{4}$$

This model is reminiscent of class-based n-gram LMs (Brown et al., 1992) and their derivatives (Saul and Pereira, 1997).

[10]We choose to explore inference-time mechanisms instead of training mechanisms to mix experts because 1) we want to avoid substantially increasing training costs, i.e., GPU communication between domain experts and 2) we want to maintain the modularity of experts. Exploring mechanisms for training expert mixtures while satisfying these desiderata is a rich area for future work.

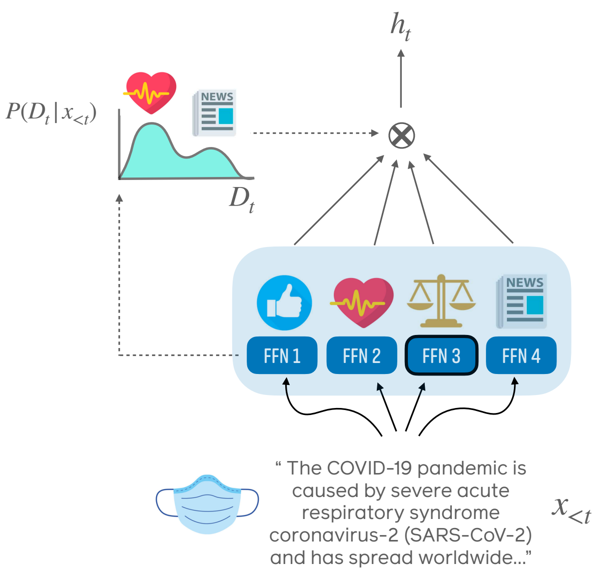

h_t
$P(D_t|x_{<t})$
NEWS
D_t
FFN 1
FFN 2
FFN 3
FFN 4
" The COVID-19 pandemic is caused by severe acute respiratory syndrome coronavirus-2 (SARS-CoV-2) and has spread worldwide..."
$x_{<t}$

Figure 3: Illustration of inference with domain expert mixing. For a given input text $\boldsymbol{x}_{<t}$ from CORD-19, we estimate a posterior domain probabilities $p(D_t \mid \boldsymbol{x}_{<t})$, informed by a prior that is either iteratively updated during inference, or is precomputed and cached on held-out data. In this example, the model assigns highest domain probabilities to the medical and news domains. We use these probabilities in a weighted mixture of expert outputs to compute the hidden representation $\mathbf{h}_t$.

We have already designed the DEMIX LM to condition on a domain label, giving a form for $p(X_t \mid \boldsymbol{x}_{<t}, D_t = j)$. The modification is to treat $g_1, \ldots, g_n$ as a posterior probability over domains, calculated at each timestep, given the history so far.

To do this, we apply Bayes' rule:

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

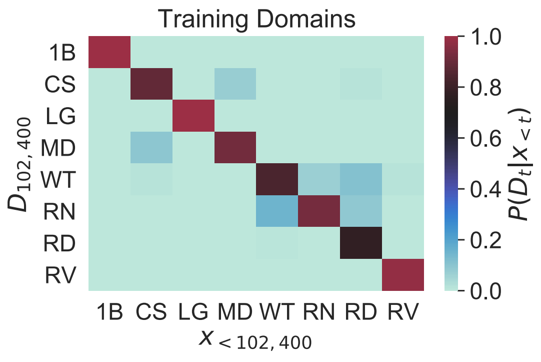
$$= \frac{p(\boldsymbol{x}_{<t} \mid D_t = j) \cdot p(D_t = j)}{\sum_{j'=1}^{n} p(\boldsymbol{x}_{<t} \mid D_t = j') \cdot p(D_t = j')} \tag{6}$$

The conditional probabilities of word sequences given a domain label, as noted above, are already defined by the DEMIX LM. For the prior over domain labels, we consider three alternatives:

Uniform Fix the prior to be uniform across the known domains.

Updating Set the prior at timestep t to be an exponentially-weighted moving average of the posteriors from previous timesteps:

$$p(D_t = j) \propto \sum_{t'=1}^{t-1} \lambda^{t-t'} \cdot p(D_{t'} = j \mid \boldsymbol{x}_{t'}) \tag{7}$$



Justifies mixing on unseen domains vs not mixing in training domains

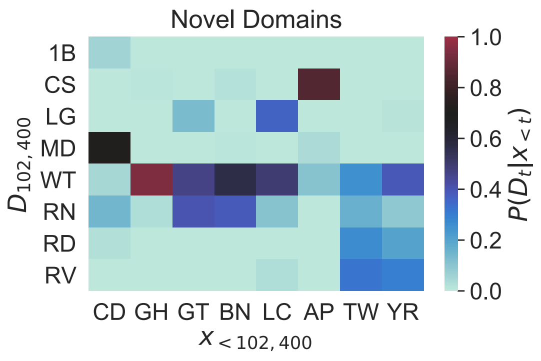


Figure 4: Estimates of posteriors $p(D_t \mid \boldsymbol{x}_{<t})$ with a DEMIX LM with 1.3B parameters per GPU, after 100 sequences (i.e., 102,400 tokens) of data in training domains (top heatmap) and new domains (bottom heatmap). Key: LG → LEGAL, MD → Med, WT → WEBTEXT, RN → REALNEWS, RD → REDDIT, RV → REVIEWS, CD → CORD-19, GH → GITHUB, GT → GUTENBERG, BN → BREAKING NEWS, LC → CONTRACTS, AP → ACL PAPERS, TW → TWEETS, YR → YELP REVIEWS.

During evaluation, this moving average is calculated over the posterior at the end of each sequence block. The decay factor avoids putting too much weight on calculations made early in the dataset, when posterior calculations are noisier (Appendix §A.6). We performed a small grid search over {0.1, 0.3, 0.5, 1.0} to set the value λ, and found that 0.3 worked well for most settings.

Cached If, prior to testing, some data from the test distribution is available, we calculate the posterior over domain labels from that data, and fix the prior to that estimate. Under this setting, we use 100 sequences (i.e., 102,400 tokens) from the development set to estimate the prior, which we found to result in stable posterior probabilities (see Appendix §A.6 for more details).

Assumes data from testing domains is available

	Parameters per GPU			
	125M	350M	760M	1.3B
DENSE	25.9	21.4	18.4	17.8
DENSE (B)	25.3	19.6	18.3	17.1
+DOMAIN-TOKEN	24.8	20.4	18.4	18.0
DEMIX (naive)	28.8	23.8	21.8	21.1
DEMIX (average)	27.2	22.4	21.5	20.1
DEMIX (uniform)	24.5	20.5	19.6	18.7
DEMIX (updating)	21.9	18.7	17.6	17.1
DEMIX (cached)	**21.4**	**18.3**	**17.4**	**17.0**

Table 5: Average perplexity on domains unseen during training. Mixing domain experts with a prior estimated using a small amount of data in the target domain outperforms all other baselines.

We display an illustration of the mixture technique in Figure 3.

5.2 Visualizing Domain Membership

In Figure 4, we plot the posteriors, calculated using the updating method above after 100 sequences of development data, each from training and novel domains. This evaluation is carried out using the DEMIX LM with 1.3B parameters per GPU from §4, with no modifications.

For known domains (top heatmap of Figure 4), the correct label has the highest posterior, but these datasets do not appear to be as distinct or mutually exclusive as we assume. For example, Reddit data is estimated to be around 80% REDDIT, 11% WEBTEXT, and 8% REALNEWS. More variation in the estimates is expected and observed for the new domains (bottom heatmap of Figure 4). While ACL PAPERS is mostly associated with the CS domain, and BREAKING NEWS mostly with the WEBTEXT and REALNEWS domains, CORD-19 is spread across MED, REALNEWS, and 1B; YELP REVIEWS across REVIEWS, WEBTEXT, and REDDIT. The alignment of multiple domains like GITHUB and CONTRACTS primarily to WEBTEXT suggests the benefit of including a relatively heterogeneous domain in training.

5.3 Experimental Setup

We experiment with the corpus of novel domains (Table 1) to test out-of-distribution performance. We evaluate the three mixture treatments of DEMIX layers (i.e., uniform, updating, and cached priors) against five baselines. Note that no new models are trained for this experiment beyond those used in §4.

DENSE and DENSE (Balanced) These are the basic baselines trained as in §4; there is no explicit reasoning about domain.

+DOMAIN-TOKEN Here test data is evaluated using each domain label token, and we choose the lowest among these perplexity values per test set.

DEMIX (naive) Similar to +DOMAIN-TOKEN, we evaluate the data separately with each of the eight experts, and report the lowest among these perplexity values per test set.

DEMIX (average) At every timestep, we take a simple average of the eight experts' predictions.

5.4 Results

Novel Domain Performance Results averaged across the eight novel domains are summarized in Table 5. Ensembling DEMIX experts outperforms DENSE baselines and using experts individually (i.e., the "naive" baseline), and caching a prior prior to evaluation results in the best average performance. While +DOMAIN-TOKEN is competitive with naively using DEMIX layers in-domain (Table 3), it consistently underperforms DEMIX with a weighted mixture on the novel domains. We observe that ensembling DEMIX experts with a cached prior allows smaller models to match or outperform much larger DENSE models. We also find that weighted ensembling outperforms simple averaging, confirming the importance of sparsity in the expert mixture.

Examining per-domain performance (Appendix §A.5), we find that DEMIX LMs with a cached prior either outperform DENSE baselines or closely match them. The largest improvement against DENSE baselines comes from the TWEETS domain, which are on average 67% better across all model sizes. This domain is heterogeneous according to the DEMIX model (Figure 4), confirming the importance of mixing experts for heterogeneous domains. These results demonstrate that conditioning the LM on domains during training need not come at a large cost to generalization to new domains, and in many cases can provide large boosts in performance over DENSE baselines.

In-Domain Performance We can also apply the expert mixture variant of inference (using a cached prior) to the training domains. We find that doing so is beneficial; see the last line of Table 3.

We see improvements in performance across all domains for every scale, though the largest im-

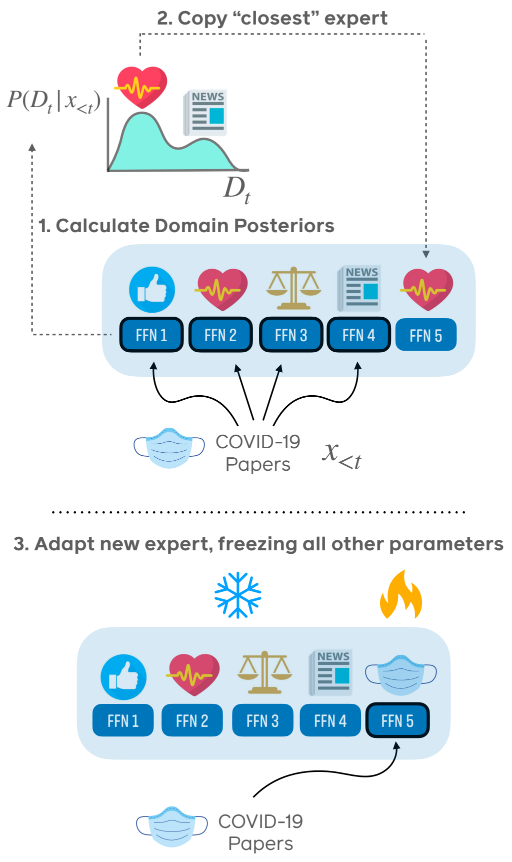


Figure 5: Illustration of DEMIX-DAPT. First, we estimate domain posteriors on a held out sample of the target domain (in this case, CORD-19). We then initialize a new expert with the parameters of the most probable expert under the domain posterior distribution. Finally, we adapt the parameters of the newly initialized expert to the target domain, keeping all other parameters in the LM frozen.

provements seem to come from hetereogeneous domains (across all model sizes, REDDIT improves on average 10.7%, WEBTEXT 2.4%, REALNEWS 1.9%), again confirming that our intuition that domain metadata may not perfectly align with the most effective domain boundaries.

6 Adaptive Pretraining with New Experts

Domain-adaptive, continued pretraining[11] of a language model (DAPT) is a way to use unannotated, in-domain text to improve task performance (Gururangan et al., 2020). However, for a large model, DAPT with DENSE training (which we refer to as DENSE-DAPT) is expensive and may not be feasible on some computational budgets. Furthermore, DENSE-DAPT may result in forgetting what was learned during earlier training phases, limiting reusability.

The modular approach of DEMIX LMs allows the model to avoid forgetting training domains and adapt cheaply: we can train a new expert and add it to the DEMIX layers of the network without updating the other experts or the shared parameters. Because the original model is not changed, forgetting is impossible. We refer to this method of adaptation as DEMIX-DAPT.[12]

We display an illustration of DEMIX-DAPT in Figure 5. We instantiate a new expert in each DEMIX feedforward layer, initialize it with the parameters of the pretrained expert nearest to the new domain. We use the posterior calculations from §5 on a held-out sample to choose the most probable expert. We then train the added expert on target data, updating only the new expert parameters. For inference, we use the weighted mixture of domain experts with a cached prior (§5).

6.1 Experimental Setup

We compare DEMIX-DAPT to DENSE-DAPT on all novel domains. We report final test-set perplexity after adapting to each domain for 1 hour with 8 NVIDIA V100 32GB GPUs, tracking validation perplexity every 10 minutes for early stopping. We adapt to each novel domain with the same hyperparameters as the original phase of training (§4), except for a 10x smaller learning rate.

6.2 Results

Adding one expert We display examples of DEMIX-DAPT and DENSE-DAPT on a single additional domain in Figure 6. We observe that while DENSE-DAPT reduces perplexity on the novel domain, its performance on the training domains progressively worsens, displaying the forgetting effect (we show similar results in larger models in Appendix §A.7). In contrast, DEMIX-DAPT reduces perplexity on the novel domain *without* forgetting.

We generally observe that DEMIX-DAPT outperforms DENSE-DAPT for some domains (e.g., CORD-19 and ACL PAPERS), while it closely approaches DENSE-DAPT for others (e.g., GUTENBERG; Appendix §A.5). Overall, the parameters for the additional expert comprise about 10% of the total parameters in the DEMIX model, and DENSE-DAPT involves updating all the parameters of the

[11]This approach typically precedes supervised fine-tuning on task data, hence *pre*training.

[12]Our proposed technique is reminiscent of *Progressive Neural Networks* (Rusu et al., 2016).

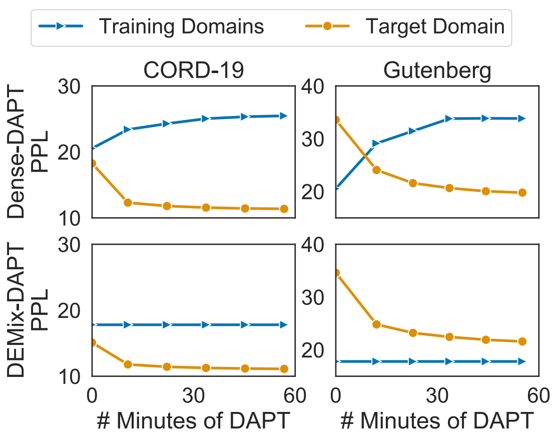


Figure 6: Adapting LMs with 125M parameters per GPU to CORD-19 or GUTENBERG. Top row: when performing DENSE-DAPT on a new domain (TARGET), average perplexity on all pretraining domains degrades. Bottom row: DEMIX-DAPT avoids that degradation while achieving close (in the case of GUTENBERG) or better (in the case of CORD-19) performance. The new CORD-19 expert was initialized with the MED expert, and the new GUTENBERG expert was initialized with a WEBTEXT expert.

		Parameters per GPU			
Domains	**# Experts**	125M	350M	760M	1.3B
TRAINING	8	17.8	14.7	13.9	**13.4**
	16	**17.7**	**14.6**	**13.7**	**13.4**
NOVEL	8	21.4	18.3	17.4	17.0
	16	**16.0**	**14.0**	**13.5**	**12.5**

Table 6: Average perplexity in training and novel domains before and after adding 8 experts adapted to the novel domains (via DEMIX-DAPT). Adding experts reduces perplexity on all domains, even those previously seen.

model towards in the target domain, so we would expect that DENSE-DAPT outperforms DEMIX-DAPT in some cases. The strong performance of DEMIX-DAPT on domains like CORD-19 and ACL PAPERS suggests that DEMIX-DAPT is especially helpful when the target domain strongly aligns with one of the experts (Figure 4).

Adding eight experts With expert mixing (§5), newly added experts can be combined with existing ones in the model at test time. To more thoroughly understand the effect of adding more experts to the system, we add all experts adapted to novel domains to the DEMIX model from §4. We display the performance of a DEMIX LM with 16 experts (8 experts trained on training domains, 8 additional experts adapted to novel domains) in Table 6. We generally observe that DEMIX-DAPT reduces perplexity on all domains for all model sizes, again without forgetting.

Adding the eight additional experts in fact reduces perplexity on *previously seen* domains. For example, across all model sizes, on average, we see an 2.4% reduction on MED, 1.8% reduction on REALNEWS, and 2% reduction on REDDIT (Appendix §A.5). These improvements are small, which is expected given that we only performed DEMIX-DAPT for at most one hour with eight GPUs. Even so, these results suggest that DEMIX layers can enable the LM to incorporate knowledge from novel domains to improve its performance on previously seen domains.

7 Language Models with Removable Parts

Current LM pretraining datasets are rife with undesirable content, from hatespeech to extremism (Gehman et al., 2020; Bender et al., 2021). Another consequence of DENSE training is that it is difficult to restrict the model's access to these problematic domains after training, as might be desirable for many user-facing tasks (Xu et al., 2020; Dinan et al., 2021).

DEMIX layers offer new capabilities for lightweight control over the domains in the training data that LMs use to make predictions at inference time. In particular, since DEMIX layer experts specialize to their domain (Figure 2), experts that are assigned to domains that are *unwanted* at test-time can be simply disabled and unused.

A key question is whether disabling an expert can simulate a model that has not been exposed to that domain, which we study in this section. However, since the self-attention and input embedding parameters in the DEMIX LM are shared across domains, removing an expert offers no guarantee of having fully forgotten content from the removed domain. Establishing such bounds is an important avenue for future work.

7.1 Experimental Setup

To evaluate whether we can simulate models that have not been exposed to a particular domain, we compare three settings:

+EXPERT A DEMIX LM with all experts active.

–EXPERT A DEMIX LM with a domain expert deactivated.

	125M Parameters per GPU		
Domain	**+EXPERT**	**–EXPERT**	**–DOMAIN**
1B	13.7	25.5	**30.4**
CS	15.7	22.4	**25.4**
LEGAL	8.9	20.9	**22.7**
MED	12.4	18.6	**21.9**
WEBTEXT	20.9	**27.3**	25.4
REALNEWS	18.9	**26.7**	25.0
REDDIT	34.4	47.8	**51.3**
REVIEWS	20.5	39.0	**43.0**
Average	18.2	28.5	**30.6**

Table 7: In a 125M parameter model, removing a domain expert (–EXPERT) results in perplexity degradation on the corresponding domain, approaching the performance of an LM that has not been exposed to that domain (–DOMAIN). Here we bold the *worst* performing model for each domain, i.e. the one that gets the *highest* perplexity.

–DOMAIN A DEMIX LM retrained from scratch without a particular domain. We replace the removed domain with GUTENBERG.[13]

We evaluate expert removal (+EXPERT and –EXPERT) with the DEMIX LM with 125M parameters per GPU from §4, with no modifications. For all baselines,we evaluate use expert mixing with a cached prior (§5).

7.2 Results

Removing a domain expert harms model performance on the associated domain, in most cases approaching the performance of a model that has not been exposed to data from that domain (Table 7). In some cases (e.g., WEBTEXT and REALNEWS), –EXPERT even underperforms –DOMAIN. This leads us to conjecture that most domain-specific learning happens within the DEMIX layer, despite the fact that other parts of the model are affected by all training domains.

8 Related Work

Incorporating Metadata Document metadata has been commonly used to improve the quality of topic models (Mimno and McCallum, 2012; Ramage et al., 2009; Zhu et al., 2012), and previous works have used metadata for adapting RNN-based language models (Jaech and Ostendorf, 2018) or learning better document representations (Card et al., 2018). Zellers et al. (2019) and Keskar et al. (2019) prepend document metadata in the input text (similar to our +DOMAIN-TOKEN setting) while training transformer LMs to provide better inference-time control of text generation.

[13]Our cluster requires that jobs are allocated with eight GPUs, necessitating eight experts — hence the substitution.

Inference-time Control DEMIX layers provide a simple mechanism for inference-time control of language model behavior. Previously proposed methods for inference-time control are either expensive to use (Dathathri et al., 2020), or rely on densely trained models (e.g., Keskar et al., 2019). Liu et al. (2021) use multiple experts for inference-time text generation control. This method may be applied to DEMIX layers to steer text generation with experts trained on different domains.

Multilinguality Related to variation across domains is crosslingual variation. Past work has suggested that multilingual models benefit from language-specific parameters (Fan et al., 2020; Pfeiffer et al., 2020; Chau et al., 2020). Here, we investigate the effect of incorporating *domain*-specific parameters into the LM. Though the boundaries between languages are (often) more clear than those among domains, DEMIX layers draw inspiration from multilingual research, and future work might explore a compositional approach with both language experts and domain experts.

Continual Learning DEMIX-DAPT is a type of continual learning, in which the model learns incrementally on new data (Chen et al., 2018). Previously proposed techniques to support continual learning include regularization (Kirkpatrick et al., 2017), meta-learning (Munkhdalai and Yu, 2017), episodic memory modules (Lopez-Paz and Ranzato, 2017; de Masson d'Autume et al., 2019), and data replay (Sun et al., 2019), all of which may be combined with DEMIX layers. Model expansion techniques to incorporate new reinforcement learning or visual tasks (Rusu et al., 2016; Draelos et al., 2017) is especially related to DEMIX-DAPT. Our results suggest that continual learning in LMs is naturally enabled with modular domain experts; this may be further explored using temporally-relevant domains (Lazaridou et al., 2021).

LM Adapters Also related to DEMIX-DAPT is the line of work into adapter modules for pretrained LMs (Houlsby et al., 2019; Pfeiffer et al., 2020). Similar to the setting in which we add experts for new domains, adapter modules involve freezing the pretrained language model and updating a small number of additional parameters that are appended to certain parts of the network. This study confirms previous findings that only a subset of LM parameters need to be fine-tuned to a target dataset (Zaken et al., 2021). Expert addition may be performed with adapter modules to further improve efficiency.

Multi-Domain Models Multi-domain models have been studied extensively in the context of machine translation, first with statistical systems (Banerjee et al., 2010; Sennrich et al., 2013), and more recently with neural networks (Pham et al., 2021). Other works have explored multi-domain settings with smaller models and explicit domain labels, using supervision (e.g., Wright and Augenstein, 2020; Guo et al., 2018; Zeng et al., 2018) or dense training (e.g., Maronikolakis and Schütze, 2021). Previous studies have shown the importance considering domains when adapting LMs (Ramponi and Plank, 2020; Gururangan et al., 2020). Our study establishes the importance of considering domains when training LMs from scratch.

9 Conclusion

We introduce DEMix layers for language models, which provide modularity at inference time, addressing limitations of dense training by providing a rapidly adaptable system. DEMix layers experts can be mixed to handle heterogeneous or unseen domains, added to iteratively incorporate new domains, and removed to restrict unwanted domains.

There are many exciting directions for future work, in addition to those described throughout the paper. They include combining domain and token-level routing, to realize the benefits of modularity while scaling models efficiently. The design of DEMix layers assumes access to coarse provenance labels (or other metadata) to identify domains in pretraining data; an alternative option is to use unsupervised learning to discover domains in the corpus, which, in concert with domain metadata, may lead to better DEMix expert assignments. Furthermore, in this work, we study DEMix layers with a dataset that has a few large domains. In practice, textual domains usually contain many diverse subdomains of varying prevalence. Training DEMix layers on dataset with a long tail of domains may require automatic measures to cluster smaller domains, or hierarchical experts that are specialized to progressively narrower data distributions.

Acknowledgments

The authors thank members of UWNLP, FAIR, and AI2, specifically Shruti Bhosale, Tim Dettmers, Emily Dinan, Doug Downey, Margaret Li, Myle Ott, Ofir Press, and Swabha Swayamdipta, for helpful comments. At UW, this work was partially supported by NSF grant 1562364, the Office of Naval Research under MURI grant N00014-18-1-2670, and an Amazon research award.

References

Armen Aghajanyan, Akshat Shrivastava, Anchit Gupta, Naman Goyal, Luke Zettlemoyer, and Sonal Gupta. 2021. Better fine-tuning by reducing representational collapse. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Roee Aharoni and Yoav Goldberg. 2020. Unsupervised domain clusters in pretrained language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7747–7763, Online. Association for Computational Linguistics.

Naveen Arivazhagan, Ankur Bapna, Orhan Firat, Dmitry Lepikhin, Melvin Johnson, Maxim Krikun, Mia Xu Chen, Yuan Cao, George Foster, Colin Cherry, Wolfgang Macherey, Zhifeng Chen, and Yonghui Wu. 2019. Massively multilingual neural machine translation in the wild: Findings and challenges.

Ramy Baly, Georgi Karadzhov, Dimitar Alexandrov, James Glass, and Preslav Nakov. 2018. Predicting factuality of reporting and bias of news media sources. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3528–3539, Brussels, Belgium. Association for Computational Linguistics.

Pratyush Banerjee, Jinhua Du, Baoli Li, Sudip Naskar, Andy Way, and Josef van Genabith. 2010. Combining multi-domain statistical machine translation models using automatic classifiers. In *Proceedings of the 9th Conference of the Association for Machine Translation in the Americas: Research Papers*, Denver, Colorado, USA. Association for Machine Translation in the Americas.

Jason Baumgartner, Savvas Zannettou, Brian Keegan, Megan Squire, and Jeremy Blackburn. 2020. The pushshift reddit dataset.

Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, page 610–623, New York, NY, USA. Association for Computing Machinery.

Peter F. Brown, Vincent J. Della Pietra, Peter V. deSouza, Jenifer C. Lai, and Robert L. Mercer. 1992. Class-based *n*-gram models of natural language. *Computational Linguistics*, 18(4):467–480.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

Dallas Card, Chenhao Tan, and Noah A. Smith. 2018. Neural models for documents with metadata. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Caselaw Access Project. 2018. Caselaw access project.

Ethan C. Chau, Lucy H. Lin, and Noah A. Smith. 2020. Parsing with multilingual BERT, a small corpus, and a small treebank. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1324–1334, Online. Association for Computational Linguistics.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2014. One billion word benchmark for measuring progress in statistical language modeling.

Zhiyuan Chen, Bing Liu, Ronald Brachman, Peter Stone, and Francesca Rossi. 2018. *Lifelong Machine Learning: Second Edition*.

Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A. Smith, and Matt Gardner. 2021. A dataset of information-seeking questions and answers anchored in research papers. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4599–4610, Online. Association for Computational Linguistics.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2020. Plug and play language models: A simple approach to controlled text generation. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Cyprien de Masson d'Autume, Sebastian Ruder, Lingpeng Kong, and Dani Yogatama. 2019. Episodic memory in lifelong language learning.

Emily Dinan, Gavin Abercrombie, A. Stevie Bergman, Shannon Spruit, Dirk Hovy, Y-Lan Boureau, and Verena Rieser. 2021. Anticipating safety issues in e2e conversational ai: Framework and tooling.

T. Draelos, N. Miner, Christopher C. Lamb, Jonathan A. Cox, Craig M. Vineyard, Kristofor D. Carlson, William M. Severa, C. James, and J. Aimone. 2017. Neurogenesis deep learning: Extending deep networks to accommodate new classes. *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 526–533.

Jacob Eisenstein, Brendan O'Connor, Noah A. Smith, and Eric P. Xing. 2014. Diffusion of lexical change in social media. *PLoS ONE*, 9(11):e113114.

Angela Fan, Shruti Bhosale, Holger Schwenk, Zhiyi Ma, Ahmed El-Kishky, Siddharth Goyal, Mandeep Baines, Onur Celebi, Guillaume Wenzek, Vishrav Chaudhary, Naman Goyal, Tom Birch, Vitaliy Liptchinsky, Sergey Edunov, Edouard Grave, Michael Auli, and Armand Joulin. 2020. Beyond english-centric multilingual machine translation.

William Fedus, Barret Zoph, and Noam Shazeer. 2021. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. 2020. The pile: An 800gb dataset of diverse text for language modeling.

Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A. Smith. 2020. RealToxicityPrompts: Evaluating neural toxic degeneration in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3356–3369, Online. Association for Computational Linguistics.

Github Archive Project. Github archive project.

Aaron Gokaslan and Vanya Cohen. 2019. Openwebtext corpus.

Jiang Guo, Darsh Shah, and Regina Barzilay. 2018. Multi-source domain adaptation with mixture of experts. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4694–4703, Brussels, Belgium. Association for Computational Linguistics.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, Online. Association for Computational Linguistics.

Dan Hendrycks, Collin Burns, Anya Chen, and Spencer Ball. 2021. Cuad: An expert-annotated nlp dataset for legal contract review.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp.

Robert Jacobs, Michael Jordan, Steven Nowlan, and Geoffrey Hinton. 1991. Adaptive mixture of local expert. *Neural Computation*, 3:78–88.

Aaron Jaech and Mari Ostendorf. 2018. Low-rank rnn adaptation for context-aware language modeling.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models.

Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. 2019. Ctrl: A conditional transformer language model for controllable generation.

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. 2017. Overcoming catastrophic forgetting in neural networks.

Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, Tony Lee, Etienne David, Ian Stavness, Wei Guo, Berton Earnshaw, Imran Haque, Sara M Beery, Jure Leskovec, Anshul Kundaje, Emma Pierson, Sergey Levine, Chelsea Finn, and Percy Liang. 2021. WILDS: A benchmark of in-the-wild distribution shifts. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 5637–5664. PMLR.

Angeliki Lazaridou, Adhiguna Kuncoro, Elena Gribovskaya, Devang Agrawal, Adam Liska, Tayfun Terzi, Mai Gimenez, Cyprien de Masson d'Autume, Sebastian Ruder, Dani Yogatama, Kris Cao, Tomas Kocisky, Susannah Young, and Phil Blunsom. 2021. Pitfalls of static language modelling.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2020. Gshard: Scaling giant models with conditional computation and automatic sharding.

Mike Lewis, Shruti Bhosale, Tim Dettmers, Naman Goyal, and Luke Zettlemoyer. 2021. Base layers: Simplifying training of large, sparse models.

Alisa Liu, Maarten Sap, Ximing Lu, Swabha Swayamdipta, Chandra Bhagavatula, Noah A. Smith, and Yejin Choi. 2021. Dexperts: Decoding-time controlled text generation with experts and anti-experts.

Kyle Lo, Lucy Lu Wang, Mark Neumann, Rodney Kinney, and Daniel Weld. 2020. S2ORC: The semantic scholar open research corpus. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4969–4983, Online. Association for Computational Linguistics.

David Lopez-Paz and Marc'Aurelio Ranzato. 2017. Gradient episodic memory for continual learning.

Li Lucy and David Bamman. 2021. Characterizing english variation across social media communities with bert.

Antonios Maronikolakis and Hinrich Schütze. 2021. Multidomain pretrained language models for green NLP. In *Proceedings of the Second Workshop on Domain Adaptation for NLP*, pages 1–8, Kyiv, Ukraine. Association for Computational Linguistics.

M. McCloskey and N. Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of Learning and Motivation*, 24:109–165.

David Mimno and Andrew McCallum. 2012. Topic models conditioned on arbitrary features with dirichlet-multinomial regression.

Tsendsuren Munkhdalai and Hong Yu. 2017. Meta networks. *Proceedings of machine learning research*, 70:2554–2563.

Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Anand Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, Amar Phanishayee, and Matei Zaharia. 2021. Efficient large-scale language model training on gpu clusters using megatron-lm.

Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 188–197, Hong Kong, China. Association for Computational Linguistics.

Yonatan Oren, Shiori Sagawa, Tatsunori Hashimoto, and Percy Liang. 2019. Distributionally robust language modeling. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4227–4237, Hong Kong, China. Association for Computational Linguistics.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*

(Demonstrations), pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

Jonas Pfeiffer, Ivan Vulić, Iryna Gurevych, and Sebastian Ruder. 2020. MAD-X: An Adapter-Based Framework for Multi-Task Cross-Lingual Transfer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7654–7673, Online. Association for Computational Linguistics.

MinhQuang Pham, Josep Maria Crego, and François Yvon. 2021. Revisiting multi-domain machine translation. *Transactions of the Association for Computational Linguistics*, 9:17–35.

Barbara Plank. 2016. What to do about non-standard (or non-canonical) language in nlp.

Project Gutenberg. Project gutenberg.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Daniel Ramage, David Hall, Ramesh Nallapati, and Christopher D. Manning. 2009. Labeled LDA: A supervised topic model for credit attribution in multi-labeled corpora. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 248–256, Singapore. Association for Computational Linguistics.

Alan Ramponi and Barbara Plank. 2020. Neural unsupervised domain adaptation in NLP—A survey. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6838–6855, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Stephen Roller, Sainbayar Sukhbaatar, Arthur Szlam, and Jason Weston. 2021. Hash layers for large sparse models.

Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. 2016. Progressive neural networks.

Lawrence Saul and Fernando Pereira. 1997. Aggregate and mixed-order Markov models for statistical language processing. In *Second Conference on Empirical Methods in Natural Language Processing*.

Rico Sennrich, Holger Schwenk, and Walid Aransa. 2013. A multi-domain translation model framework for statistical machine translation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 832–840, Sofia, Bulgaria. Association for Computational Linguistics.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, Florence, Italy. Association for Computational Linguistics.

Fan-Keng Sun, Cheng-Hao Ho, and Hung-Yi Lee. 2019. Lamol: Language modeling for lifelong language learning.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

Lucy Lu Wang, Kyle Lo, Yoganand Chandrasekhar, Russell Reas, Jiangjiang Yang, Doug Burdick, Darrin Eide, Kathryn Funk, Yannis Katsis, Rodney Kinney, Yunyao Li, Ziyang Liu, William Merrill, Paul Mooney, Dewey Murdick, Devvret Rishi, Jerry Sheehan, Zhihong Shen, Brandon Stilson, Alex Wade, Kuansan Wang, Nancy Xin Ru Wang, Chris Wilhelm, Boya Xie, Douglas Raymond, Daniel S. Weld, Oren Etzioni, and Sebastian Kohlmeier. 2020. Cord-19: The covid-19 open research dataset.

Dustin Wright and Isabelle Augenstein. 2020. Transformer based multi-source domain adaptation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7963–7974, Online. Association for Computational Linguistics.

Jing Xu, Da Ju, Margaret Li, Y-Lan Boureau, Jason Weston, and Emily Dinan. 2020. Recipes for safety in open-domain chatbots.

Yelp Reviews. Yelp reviews.

Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. 2021. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2019. Defending against neural fake news. In *NeurIPS*.

Jiali Zeng, Jinsong Su, Huating Wen, Yang Liu, Jun Xie, Yongjing Yin, and Jianqiang Zhao. 2018. Multi-domain neural machine translation with word-level domain context discrimination. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 447–457, Brussels, Belgium. Association for Computational Linguistics.

Jun Zhu, Amr Ahmed, and Eric P. Xing. 2012. Medlda: Maximum margin supervised topic models. *Journal of Machine Learning Research*, 13(74):2237–2278.

A Appendix

A.1 Collecting Domains

For most domains, we use the associated sources, listed in Table 1, without modification. For TWEETS, we use the Twitter Academic API. For GUTENBERG, we use the scraping tool provided in `https://github.com/aparrish/gutenberg-dammit`. For BREAKING NEWS, we identify a list of factually reliable English news sources, using the list curated by Baly et al. (2018). Specifically, we filter on "high" factuality in the data provided in this repository: `https://github.com/ramybaly/News-Media-Reliability`. We then use Newspaper3K (`https://newspaper.readthedocs.io/en/latest/`) to scrape the latest 1000 articles from each site. After dropping duplicates, we arrive at about 20K articles from 400 news sources. We provide downloading links and general instructions at `https://github.com/kernelmachine/demix-data/blob/main/DOWNLOAD_DATA.md`.

A.2 Dataset Anonymization

To anonymize certain datasets, we apply a suite of regexes that aim to identify common patterns of user-identifiable data and substitute them with dummy tokens. We display anonymization regexes and associated dummy tokens in Table 8.

A.3 Calculating TFLOPs/GPU

We use the formula presented in Narayanan et al. (2021) to calculate TFLOPs/GPU and TFLOPs/update. The spreadsheet that contains the calculations and formula can be accessed here: `https://docs.google.com/spreadsheets/d/1NO-Lz_VqZGF2fpJTFxtXyjhmaoYi6qnz50Xr8W8hgGw/edit?usp=sharing`.

A.4 Hyperparameter Assignments

We display hyperparameter assignments for LM pretraining in Tables 9, 10,11, and 12.

A.5 Per-Domain Results

We display per-domain test results in the spreadsheets at the following link: `https://docs.google.com/spreadsheets/d/1yNMZGSPAvhTi3JttLamiCULaOIGTJ4QGEOajO3b5kt8/edit?usp=sharing`

A.6 Domain Posterior Calculations

We track calculated domain posteriors over blocks of development data in Figure 7 (training domains) and Figure 8 (novel domains). The calculate domain posteriors are noisier for earlier blocks, stabilizing usually after around 50 blocks. For all experiments, we conservatively use 100 blocks of data to compute the domain posterior, though one may be able to accurately calcuate the domain posterior for some domains with less data.

A.7 Perplexity changes after DENSE-DAPT

In Table 13, we display the average perplexity change after performing DENSE-DAPT on a new domain. We observe that across all model sizes, DENSE-DAPT improves performance in the novel domain, at the cost of a large performance hit in the training domains.

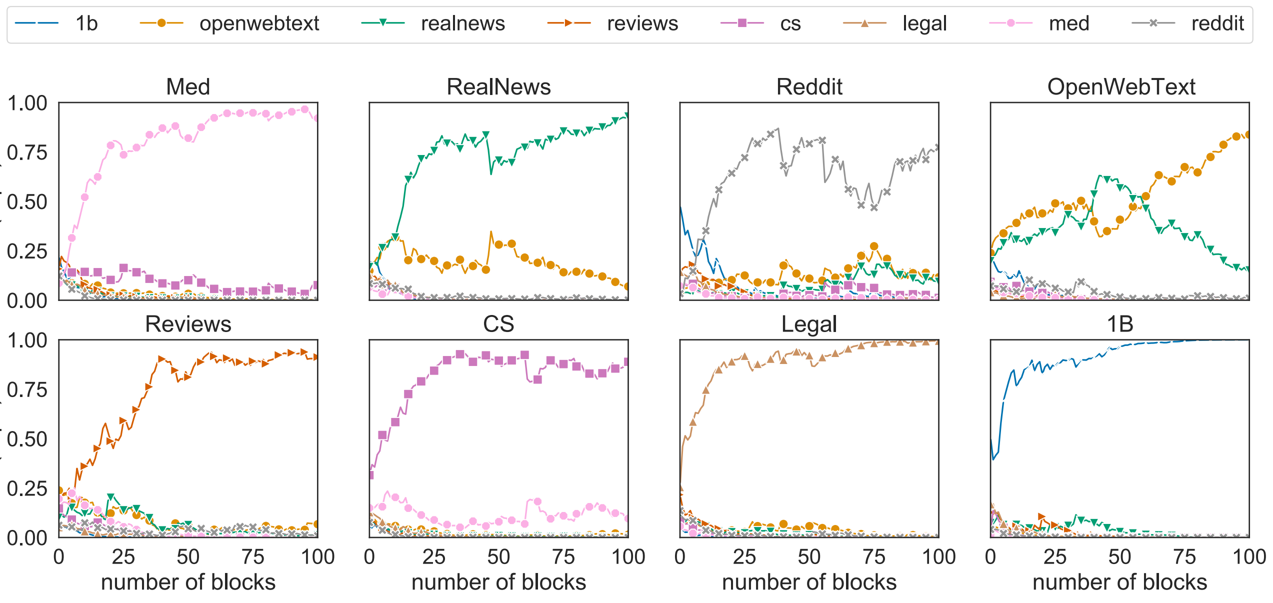


Figure 7: Calculated domain posteriors for 8 training domains.

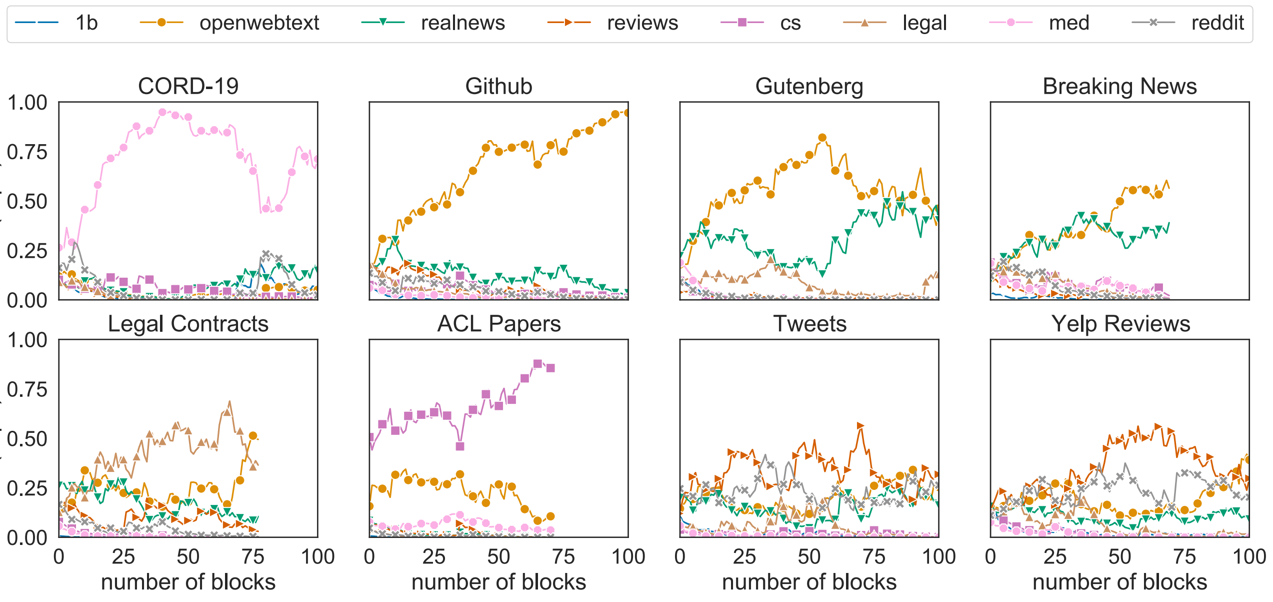


Figure 8: Calculated domain posteriors for 8 novel domains.

Category	**Link to Regex**	Dummy Token
Email	`https://regex101.com/r/ZqsF9x/1`	`<EMAIL>`
DART	`https://regex101.com/r/0tQ6EN/1`	`<DART>`
FB User ID	`https://regex101.com/r/GZl5EZ/1`	`<FB_USERID>`
Phone Number	`https://regex101.com/r/YrDpPD/1`	`<PHONE_NUMBER>`
Credit Card Number	`https://regex101.com/r/9NTO6W/1`	`<CREDIT_CARD_NUMBER>`
Social Security Number	`https://regex101.com/r/V5GPNL/1`	`<SSN>`
User handles	`https://regex101.com/r/vpey04/1`	`<USER>`

Table 8: Anonymization schema. We anonymize text using the regexes provided in the above links for the categories listed.

Computing Infrastructure	32 Volta 32GB GPUs

Hyperparameter	**Assignment**
architecture	GPT-3 small
tokens per sample	1024
batch size	2
number of workers	2
learning rate	[5e–4, 3e–4, 1e–4]
clip norm	0.1
gradient acculumation steps	8
number of steps	300,000
save interval updates	6,000
validation interval	3,000
number of warmup steps	24,000
learning rate scheduler	polynomial decay
learning rate optimizer	Adam
Adam beta weights	(0.9, 0.95)
Adam epsilon	10e-8
weight decay	0.1

Table 9: Hyperparameters for pretraining the LM with 125M parameters per GPU. All hyperparameters are the same for DEMIX and DENSE training.

Computing Infrastructure	64 Volta 32GB GPUs

Hyperparameter	**Assignment**
architecture	GPT-3 medium
tokens per sample	1024
batch size	2
number of workers	2
learning rate	[5e–4, 3e–4, 1e–4]
clip norm	0.1
gradient acculumation steps	8
number of steps	120,000
save interval updates	3,000
validation interval	2,000
number of warmup steps	9,600
learning rate scheduler	polynomial decay
learning rate optimizer	Adam
Adam beta weights	(0.9, 0.95)
Adam epsilon	10e-8
weight decay	0.1

Table 10: Hyperparameters for pretraining the LM with 350M parameters per GPU. All hyperparameters are the same for DEMIX and DENSE training.

Computing Infrastructure	128 Volta 32GB GPUs

Hyperparameter	**Assignment**
architecture	GPT-3 large
tokens per sample	1024
batch size	2
number of workers	2
learning rate	[5e–4, 3e–4, 1e–4]
clip norm	0.1
gradient acculumation steps	8
number of steps	65,000
save interval updates	2,000
validation interval	1,000
number of warmup steps	5,200
learning rate scheduler	polynomial decay
learning rate optimizer	Adam
Adam beta weights	(0.9, 0.95)
Adam epsilon	10e-8
weight decay	0.1

Table 11: Hyperparameters for pretraining the LM with 760M parameters per GPU. All hyperparameters are the same for DEMIX and DENSE training.

Computing Infrastructure	128 Volta 32GB GPUs

Hyperparameter	**Assignment**
architecture	GPT-3 XL
tokens per sample	1024
batch size	2
number of workers	2
learning rate	[5e–4, 3e–4, 1e–4]
clip norm	0.1
gradient acculumation steps	8
number of steps	50000
save interval updates	2,000
validation interval	500
number of warmup steps	4000
learning rate scheduler	polynomial decay
learning rate optimizer	Adam
Adam beta weights	(0.9, 0.95)
Adam epsilon	10e-8
weight decay	0.1

Table 12: Hyperparameters for pretraining the LM with 1.3B parameters per GPU. All hyperparameters are the same for DEMIX and DENSE training.

		Parameters			
		125M	350M	760M	1.3B
DENSE-DAPT	T	+70.1%	+21.4%	+16.7%	+20.6%
	N	–55.1%	–46.6%	–38.3%	-44.4%

Table 13: Average change in perplexity in training (T) and novel (N) domains after DENSE-DAPT. Negative values indicate better performance relative to the original DENSE LM. While average perplexity in the novel domains decreases more for DENSE-DAPT, this comes at the cost of a significant deterioration in performance in training domains.