DEMIX Layers: Disentangling Domains for Modular Language Modeling

Anonymous ACL submission

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 includes 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 (especially for out-of-domain data), increase training efficiency, and enable rapid adaptation. Mixing experts during inference, using a parameter-free weighted ensemble, enables better generalization to heterogeneous or unseen domains. We also show it is possible to add experts to adapt to new domains without forgetting older ones, and remove experts to restrict access to unwanted domains. Overall, these results demonstrate benefits of domain modularity in language models.

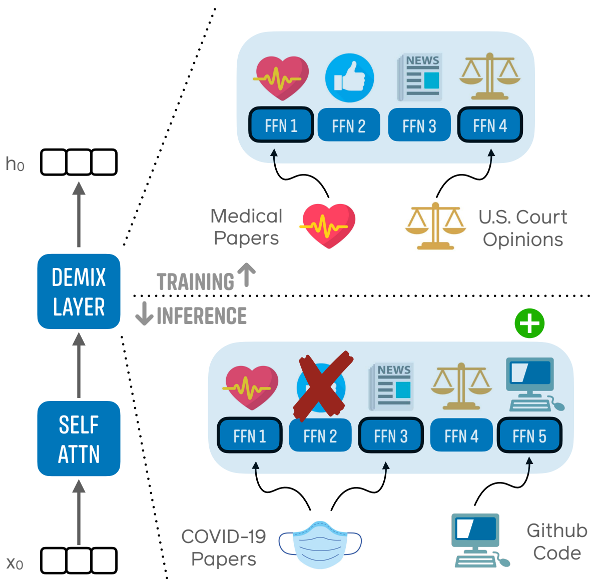


Figure 1: Illustration of a DEMIX layer in a single transformer block. During training, expert feedforward networks are conditionally activated based on the domain (here, document provenance) of the input sequence (i.e., scientific papers or court opinions). At inference time, the language model has new modular functions: domain experts can be mixed to handle heterogeneous domains (e.g., COVID-19 papers), added to adapt to novel domains (e.g., Github code), or removed to reduce the influence of unwanted domains (e.g., social media). Image attribution in §A.1.

1 Introduction

Most language models (LM) are trained with data homogeneity: all parameters are updated to minimize the loss on all of the data. We refer to this as *dense training*. Dense training leaves variation in the data, or *domains*, 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 adaptation hard 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 to restrict access to certain domains the LM has been exposed to during training (e.g., those that contain hatespeech; 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).

This is an example of conditional computation (Fedus et al., 2021; Lepikhin et al., 2020; Lewis et al., 2021; Roller et al., 2021), which follows 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.[1]

We identify domains using coarse provenance categories (e.g., whether a document is a medical research paper or a Reddit post; §2). Training on data from eight different domains, we find that replacing every feedforward layer in the transformer with a DEMIX layer consistently improves in-domain performance (§4). To improve performance in settings in which the target data does not clearly align with a single domain, we introduce a parameter-free probabilistic approach to dynamically estimate a *weighted mixture* of domains during inference (§5). We observe that expert mixing provides especially strong performance gains on *novel* test-time domains, as well as consistent performance improvements on test data from the *training* domains, which may themselves be heterogeneous.

Our results suggest that DEMIX consistently improves model generalization, especially out-of-domain, while enabling many new modular capabilities. 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, predictably changing 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. These results, in aggregate, demonstrate the considerable benefits of moving away from treating data homogeneously during language modeling. Our code is publicly available.[2]

2 Multi-Domain Corpus

To better measure domain modularity, we introduce a new multi-domain corpus constructed with *domain provenance* that records the original dataset each document appeared in (Table 10). Defining domains in this way is intuitive and conveys a great deal about the type of language that can be expected in each document. Other accounts of domains (e.g., Lucy and Bamman, 2021; Gururangan et al., 2020) may be studied in future work. While other multi-domain corpora (Koh et al., 2021; Gao et al., 2020) cover many more domains, our corpus is restricted to datasets with more permissive licensing to support reproducibility.

We divide our data into training and test domains. The **training** domains text from eight English corpora (top of Table 10), each of which varies in complexity and coverage, totaling 73.8B whitespace-separated tokens (§A.2). Our test (or **novel**) domains include eight collections of English text (bottom of Table 10), 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.

§A.2 has more details on how these data were collected, as well as per-domain token counts. For larger domains, we use an additional 10M tokens for the validation and test sets each. Smaller domains have 1M tokens in each (Table 10). To support future work with the data, we also release an API to download and preprocess it into a format compatible with Fairseq (Ott et al., 2019).[3]

3 DEMIX Layer

3.1 Background: Mixture-of-Experts Transformers

The transformer architecture interleaves multi-head self-attention, layer-norms, and feedforward networks (Vaswani et al., 2017). Our focus is on the

[1]We perform a detailed comparison of learned and DEMIX routing in §A.5.

[2]`anonymous.com`

[3]`anonymous.com`

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 to replace dense feedforward layers with 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 dense feedforward network. If g routes to a single expert, then the computational cost (in floating-point operations; FLOPs) will be same as a corresponding DENSE network, even though it has 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 requires careful load balancing to encourage the model to use all experts, motivating work on explicit balancing mechanisms (Lewis et al., 2021).

Instead of learning g, we use domain metadata to route data to experts at the *document* (i.e., sequence) level. During training, every token in an input text 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 10). If we index the experts by $\mathcal{D}$ and $d \in \mathcal{D}$ is the domain label for the current training instance, then

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

We assume that each *training* document is associated with a single domain label. However, we relax this requirement at inference time (§5), to model unseen or heterogeneous domains.

We perform a detailed comparison of DEMIX routing with GSHARD (Lepikhin et al., 2020), a mixture-of-experts transformer LM with learned token-level routing, in §A.5. Our results suggest that learned token-level routing does not enable modularity, underperforms DEMIX at similar computational budgets (especially on novel domains), and is much less efficient to train and evaluate.

		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 1: 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 §A.4 for more details.

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 separate 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 (see §A.6 for more details). Future work may comprehensively compare different architectural choices.

Each expert FFN_j is a two-layer MLP with the same dimensions as the original FFN layer of the transformer. This means that the effective number of parameters in the overall DEMIX LM increases (Table 1), without increasing inference runtime.

3.4 DEMIX Training

To train an LM with DEMIX layers, we partition the GPUs among the domains, so that each GPU is assigned a single domain (along with its corresponding expert). We fill a mini-batch with k sequences from a particular domain, and we send each mini-batch to its dedicated expert. We use larger batch sizes with distributed data parallel between expert parameters on GPUs assigned to the same domain; we assign n/8 GPUs to each domain (Table 1).

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 (Table 1). DEMIX achieves higher throughput because we while we sync shared parameters across all GPUs, we only sync expert parameters allocated to the same domain. DENSE models sync all parameters across all GPUs. As we increase model size, this reduces latency costs between GPUs, and hence, faster training.

3.5 Comparison with Adapters

DEMIX experts are related to adapters (Bapna and Firat, 2019), which add a small feedforward network into a frozen pretrained LM to enable parameter efficient finetuning. In contrast, our focus is on efficiently training all of the parameters of a modular LM from scratch, and as such is not directly comparable to existing adapter schemes. Although adapter-style architectures could be used for more fine-grained control over which parts of the feedforward networks are domain specific, we leave exploring such architectural variants to future work.

4 In-Domain Performance

Our first set of experiments measure in-domain performance when replacing the feedforward layers in a transformer LM with DEMIX layers.

4.1 Experimental Setup

Architecture, Input and Hyperparameters We follow the GPT-3 (Brown et al., 2020) architecture (small, medium, large, and XL) implemented in Fairseq (Ott et al., 2019). We use the GPT-2 (Radford et al., 2019) vocabulary of 50,264 BPE types, and train with 1,024-token sequences. See §A.7 for training hyperparameters.

Evaluation We follow previous work by reporting performance for a given computational budget (Lewis et al., 2021). For each model, we report test perplexity after a single run of 48 hours of training on differing numbers of NVIDIA V100 32GB GPUs (Table 1).

4.2 Models

DENSE The first baseline is a DENSE LM, implemented with distributed data parallel (Li, 2021). There is no explicit conditioning on domain.

	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 2: Average in-domain test-set perplexity across the 8 domains in the training data. We discuss the last row in §5.4. See §A.8 for per-domain results.

DENSE (balanced) We train dense models on an equal amount of data from each domain. While there is still no explicit conditioning on domain, the gradient updates during training average across all domains represented in a batch.

+DOMAIN-TOKEN This model is trained identically to DENSE (balanced), but we prepend a token to every sequence indicating its domain (during training and test time). We ignore the domain token when computing perplexity during evaluation.

DEMIX (naive) We replace every feedforward layer in the transformer with a DEMIX layer, and use DEMIX training (§3). Under this *naive* setting, the test data domain is known (e.g., the CS expert is used for CS test data). We relax this assumption in the next section.

4.3 Results

Table 2 shows test perplexities, averaged across the eight training domains. Domain balancing is consistently helpful for DENSE training. Additional domain information always helps (i.e, domain tokens or DEMIX layers), but the effects are largest for the smaller models. Overall, domain information enables the model to better specialize to different training domains. However, as the model size grows, the DENSE baseline improves, catching up to the DEMIX (naive) model, at least when considering the average perplexity across domains.

4.4 Domain Hetereogeneity

However, a more complete view of the experiments with the largest model is shown in Table 3. 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 hypothesize

	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 3: Test perplexity by domain for largest models. We discuss the last column in §5.4.

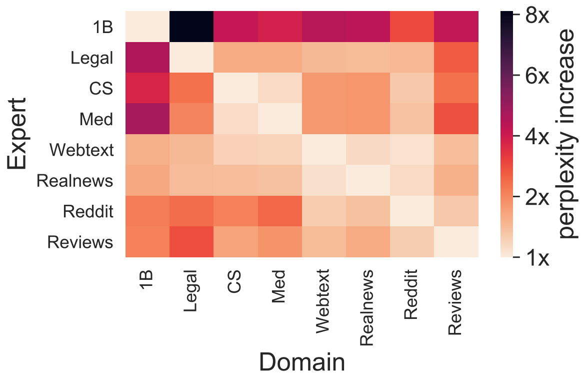


Figure 2: Heatmap of expert performance ratios, using the largest DEMIX LM (1.3B parameters per GPU). The diagonal indicates that expert specialization to their own domain. While some experts (e.g., 1B, MED) do not transfer well to most domains in the training corpus, WEBTEXT and REALNEWS experts transfer much better, confirming the heterogeneity of those domains.

that DENSE training is advantageous for heterogenous domains. Heterogeneous domains have a higher overlap with other training domains, and therefore, benefit from parameter sharing.

We support this explanation with a matrix of performance ratios across all domain experts (Figure 2), comparing the performance of all experts against the expert explicitly trained for each domain. Experts perform best on their assigned domain. However, the experts assigned to domains that benefit from DENSE training perform relatively well on many training domains.

These findings suggest overall that a discrete notion of domain is too rigid. In the next section, we soften Equation 3 into a mixture of experts.

5 Mixing Experts at Inference Time

The previous section established that incorporating DEMIX layers improves LM performance on test data from *known* training domains. 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.

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 is parameter-free and computed only at test time.

5.1 Dynamic Domain Mixtures

Consider the probabilistic view of language modeling, where we estimate $p(\boldsymbol{x}_t \mid \boldsymbol{x}_{<t})$. 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:

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

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

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

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 Set a uniform prior across domains.

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

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

	Parameters per GPU			
	125M	350M	760M	1.3B
DENSE	25.9	21.4	18.4	17.8
DENSE (balanced)	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 4: Average perplexity on novel domains. Mixing domain experts with a prior estimated using a small amount of data in the target domain outperforms all other baselines. See §A.8 for per-domain results.

During evaluation, this moving average is calculated over the posterior at the end of each sequence. The decay factor avoids putting too much weight on calculations made early in the dataset, when posterior calculations are noisier (§A.9). We performed a small grid search to set the value λ, and found that $\lambda = 0.3$ worked well.

Cached We calculate the posterior over domain labels from additional data from the test distribution, and fix the prior to that estimate. We use 100 sequences from the validation set to estimate the prior, which we found to result in stable posterior probabilities. See §A.9 for more details, and Figure 8 for an illustration of expert mixing.

5.2 Visualizing Domain Membership

In Figure 7, we plot domain posteriors calculated using the largest DEMIX LM from §4 and the updating prior, after 100 sequences of validation data. For training domains, the associated domain label has the highest probability, but some of the domains are more hetereogeneous than we assumed. More variation is observed for the novel domains. However, generally we find the domain posterior distribution to be sparse; suggesting that after estimating the domain posterior, not all experts need to be active for test evaluation.

5.3 Experimental Setup

Here, we experiment with the corpus of novel domains (Table 10). We evaluate the three mixture treatments of DEMIX layers (§5.1) against five baselines. No new models are trained for this experiment beyond those used in §4.

DENSE and DENSE (balanced) These are the basic baselines from §4.

+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 Ensembling DEMIX experts outperforms DENSE baselines and using experts individually (i.e., the "naive" baseline), and caching a prior before evaluation results in the best average performance (Table 4). Ensembling DEMIX experts with a cached prior allows smaller models to match or outperform much larger DENSE models. Weighted ensembling outperforms simple averaging and mixing with a uniform prior, confirming the importance of sparsity in the expert mixture. These results demonstrate that modularity need not come at a cost to generalization to new domains.[4]

In-Domain Performance We can also apply the expert mixture variant of inference (using a cached prior) to the training domains; see the last line of Table 2. We see performance improvements across all training domains for every scale, though the largest gains come from hetereogeneous domains (Table 3 and §A.8; across all model sizes, REDDIT improves on average 10.7%, WEBTEXT 2.4%, REALNEWS 1.9%), confirming that domain labels may not align with the most effective boundaries.

5.5 Summary

As opposed to other token-level routing mechanisms (e.g., Lepikhin et al. 2020), expert mixing in DEMIX is introduced at test-time and is parameter-free; it instead makes use of Bayesian inference with specialized experts to improve generalization. Expert mixing dynamically increases model capacity at test-time, while avoiding the need to learn token-level routing patterns during training, which is expensive and breaks modularity (§A.5).

[4]We have separately observed that with expert mixing, our largest DEMIX LM closely approaches the performance of GPT-3 *Da-Vinci* (Brown et al., 2020) on another novel domain, the LM benchmark PTB (Marcus et al., 1993). See §A.13 for more details.

6 Domain Adaptation with New Experts

Domain adaptation is an important technique to improve LM performance in new domains that are rare or unseen during training. A popular technique for adapting LMs is domain-adaptive pretraining (DAPT; Gururangan et al. 2020), which involves continued DENSE training of the LM on the target domain. However, DAPT with DENSE training (or DENSE-DAPT) is expensive (Strubell et al., 2019) and may entail forgetting domains learned during earlier training phases (Aghajanyan et al., 2021), since it updates all parameters of the LM towards the target domain. These issues make adapting large LMs less feasible, especially in domains that change frequently over time (Luu et al., 2021).

DEMIX layers allow for *cheap* adaptation *without* forgetting through a technique we call DEMIX-DAPT (see Figure 9 for an illustration). To adapt to a new domain, we initialize a new expert in each DEMIX layer using the parameters of the nearest pretrained expert, which we identify using domain posteriors from §5. We then train the added expert on target data, *updating only the new expert parameters*. For inference, we mix experts with a cached prior (§5).

6.1 Experimental Setup

We compare DEMIX-DAPT to DENSE-DAPT on the novel domains. We report test 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 §4, except with a 10x smaller learning rate. DEMIX-DAPT updates about 10% of the total parameters in the DEMIX LM, while DENSE-DAPT updates all parameters of the DENSE LM.

6.2 Results

Adding One Expert We display examples of DEMIX-DAPT and DENSE-DAPT on a single domain in Figure 3. As DENSE-DAPT proceeds, its performance on the training domains progressively worsens (see §A.14 for results with larger LMs). In contrast, DEMIX-DAPT reduces perplexity on the novel domain *without* forgetting.

Adding Eight Experts We find that adding *all* eight experts adapted to novel domains to the DEMIX model from §4 significantly reduces perplexity on novel *and* previously seen domains (Table 5) while also helping in-domain for smaller models (perhaps surprisingly, given the fact that their domain experts are frozen). For example, across all model sizes, on average, we see a 2.4% reduction on MED, 1.8% reduction on REALNEWS, and 2% reduction on REDDIT (see §A.8 for details).

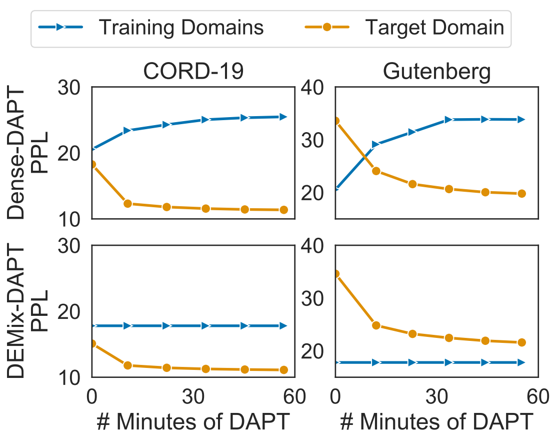


Figure 3: Adapting an LM (125M parameters per GPU) to CORD-19 or GUTENBERG. Top row: with DENSE-DAPT, average perplexity on all training domains degrades. Bottom row: DEMIX-DAPT avoids forgetting while achieving close (for GUTENBERG) or better (for CORD-19) performance on the target domain.

		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 5: 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 novel and training domains.

7 Removing Experts

DENSE LMs are also prone to unexpected behavior when deployed. For example, they may generate hatespeech (Gehman et al., 2020), which is undesirable for user-facing tasks (Xu et al., 2020).

We argue that DENSE training contributes to unexpected model behavior, as domains are learned diffusely over the parameter space, and it is difficult to restrict the model's access to certain training domains during inference. Some mechanisms have been introduced to steer a DENSE model *towards* (Keskar et al., 2019; Dathathri et al., 2020) and *away* (Welleck et al., 2019) from certain behaviors, but they tend to be expensive or require retraining the model with a different objective, which

	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 6: Removing a domain expert (–EXPERT) degrades perplexity 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.

becomes less feasible as the LM grows in size.

DEMIX layers offer a simple mechanism for cheap, lightweight control of large LMs: since domain experts specialize (Figure 2), experts assigned to unwanted domains can be disabled at test-time.[5]

7.1 Experimental Setup

Does disabling an expert simulate a model that has not been exposed to a particular training domain? To answer this question, we compare three settings: **+EXPERT**, a DEMIX LM with all experts active, **–EXPERT**, a DEMIX LM with a domain expert deactivated, and **–DOMAIN**, a DEMIX LM trained from scratch without a particular domain.[6]

For all settings, we use a DEMIX LM (125M parameters per GPU) from §4 and expert mixing with a cached prior (§5) for inference.

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 6). 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.

Our preliminary analysis here suggests that DEMIX enables LMs with *removable parts*, for quick adaptation to situations in which a particular training domain is unwanted for inference. We leave further exploration of this mechanism and its potential for LM control to future work.

[5]Removing an expert offers no guarantee of having fully forgotten content from the removed domain, since there are shared parameters in the model.

[6]We replace the removed domain with GUTENBERG, since our cluster allocates training jobs via 8-GPU nodes.

8 Related Work

Document metadata has been used to improve topic models (Mimno and McCallum, 2012), adapt RNN-based LMs (Jaech and Ostendorf, 2018), learn document representations (Card et al., 2018), and improve text generation control (Zellers et al., 2019; Keskar et al., 2019). Other inference-time methods (Dathathri et al., 2020; Liu et al., 2021) may be used to steer text generation with DEMIX experts.

Future work may explore applying DEMIX to multilingual settings, as multilingual models benefit from language-specific parameters (Fan et al., 2020; Pfeiffer et al., 2020; Chau et al., 2020).

DEMIX-DAPT is closely related to model expansion techniques in reinforcement learning or vision (Rusu et al., 2016; Draelos et al., 2017) as well as adapter modules for pretrained LMs (Houlsby et al., 2019; Pfeiffer et al., 2020). DEMIX-DAPT may be combined with continual learning methods, such as regularization (Kirkpatrick et al., 2017), meta-learning (Munkhdalai and Yu, 2017), episodic memory (de Masson d'Autume et al., 2019), and data replay (Sun et al., 2019).

Multi-domain models have been studied in machine translation (Pham et al., 2021) and supervised settings (Wright and Augenstein, 2020), and with smaller dense LMs (Maronikolakis and Schütze, 2021). Previous studies have shown the importance of 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, which provide modularity to an LM 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. Future work may combine domain and token-level routing, discover domains automatically with unsupervised learning, or scale the number of training domains.

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.

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.

Ankur Bapna and Orhan Firat. 2019. Non-parametric adaptation for neural machine translation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1921–1931, Minneapolis, Minnesota. Association for Computational Linguistics.

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. 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.

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.

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.

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.

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.

Shen Li. 2021. Getting started with distributed data parallel.

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.

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

Kelvin Luu, Daniel Khashabi, Suchin Gururangan, Karishma Mandyam, and Noah A. Smith. 2021. Time waits for no one! analysis and challenges of temporal misalignment. *ArXiv*, abs/2111.07408.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Comput. Linguist.*, 19(2):313–330.

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.

Project Gutenberg. Project gutenberg.

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

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*.

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.

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. 2019. Neural text generation with unlikelihood training.

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.

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

A Appendix

A.1 Image attribution

Images retrieved from emojipedia.org or istockphoto.com.

A.2 Collecting Domains

For most domains, we use the associated sources, listed in Table 10, without modification. 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 anonymous.com.

A.3 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 11.

A.4 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: anonymous.com

A.5 Gshard Comparison

Here we describe empirical comparisons between DEMIX and GSHARD, the token-level mixture of experts transformer proposed by Lepikhin et al. (2020). As opposed to DEMIX, which uses domain labels to route data to experts, GSHARD learns a token-level routing mechanism during training. Each token in every other layer is sent to two of k experts, and this routing is updated via backpropagation.

As GSHARD is emblematic of an learned routing procedure, we are generally interested if GSHARD naturally learns to specialize experts to domains, whether its experts are modular, and how GSHARD LM generally performs compared to DEMIX and DENSE models on our multi-domain corpus.

		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**
GSHARD	**GPUs**	32	64	128	-
	Total Experts	32	64	128	-
	GPUs/expert	1	1	1	-
	Total params	1B	6.7B	29.5B	-
	TFLOPs/update	675	4120	30,000	-
	TFLOPs/GPU	**15**	**16**	**19**	-

Table 7: Our specifications for training DENSE, DEMIX, and GSHARD 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 §A.4 for more details.

Experimental Setup We aim to make minimal changes to the overall architecture of the model, to focus on the differences afforded by token-level routing (vs. DEMIX routing). As such, we keep all architecture and computational budgets the same as our DEMIX and DENSE baselines (we generally display results for the 125M, 350M, and 760M parameter LMs). We only add the GSHARD routing procedure to every other layer, which involves routing each token to the top-2 experts of that layer. This additionally necessitates a load balancing loss to prevent only a minority of experts from being used (Lepikhin et al., 2020). All GSHARD experts are of the same size as our DEMIX experts, i.e., each expert is a two layer MLP with the same dimensions as the original feedforward layer of the transformer. We display hyperparameters used to train GSHARD in §A.7.

Model Scale In DEMIX, we always add the same number of experts as the number of training domains (in our case 8), and use extra computation to increase the batch size for each expert. Our GSHARD implementation, on the other hand, allocates one expert per GPU. This means that GSHARD adds many more experts to the system, which results in a substantially larger increase in model size (Table 7). Unlike DEMIX, GSHARD results in an increase in FLOP count relative to the DENSE model, due to a variety of additional computation during training, like load balancing and routing to two experts for every token, which DEMIX does not need.

Training efficiency However, unlike DEMIX, which increases model size while maintaining or improving GPU throughput, GSHARD in fact *reduces* GPU throughput during training (Table 7). This is due to the necessity of expensive *all-to-all* operations in GSHARD which mediate communication between experts on different GPUs that are activated for different tokens of the same document.[7] These *all-to-all* operations are bottlenecked by the quality of GPU communication channels on the cluster. We also found that additional inefficiencies are introduced via GSHARD's load balancing, since some experts are not used at test time. DEMIX has no load balancing or all-to-all communication. It uses all experts to maximum efficiency, because we simply assign GPUs to domains for our routing protocol.

Evaluation efficiency Another benefit to DEMIX is that its experts specialize to their domain, and only a sparse subset of them are activated at test time. Does token-level routing via GSHARD also result in a modular model? We explore this question by computing the average gating probabilities in the GSHARD router across all experts for all test data in each domain. We generally find that gating probabilities in GSHARD have high entropy across experts regardless of domain, suggesting that the token-level routing procedure does not in fact result in modularity out-of-the-box and all experts are needed for all input texts (Figure 4). As we increase computational budget, this issue is exacerbated; we need 128 GPUs to evaluate on the test data for the final model. Whereas with DEMIX, we only need 8 GPUs to compute the domain posterior on a subset of the validation data. Moreover, because the domain posterior is usually sparse, one can use an even smaller number of GPUs for evaluating on test data, loading only those experts with non-zero probabilities.

[7] https://images.nvidia.com/events/sc15/pdfs/NCCL-Woolley.pdf

	Parameters per GPU			
	125M	350M	760M	1.3B
DENSE (balanced)	19.9	15.8	14.3	13.6
DEMIX	**17.8**	**14.7**	**13.9**	**13.4**
GSHARD	17.2	14.3	14.2	-

Table 8: Average in-domain test-set perplexity across the 8 domains in the training data. We discuss the last row in §5.4. See §A.8 for per-domain results.

	Parameters per GPU			
	125M	350M	760M	1.3B
DENSE (balanced)	25.9	21.4	18.4	17.8
DEMIX	**21.4**	**18.3**	**17.4**	**17.0**
GSHARD	24.0	19.5	18.9	-

Table 9: Average perplexity on novel domains. Mixing domain experts with a prior estimated using a small amount of data in the target domain outperforms all other baselines. See §A.8 for per-domain results.

Model performance As noted earlier, our GShard implementation substantially increases the effective parameter count of the model relative to DEMIX (Table 7). While this expansion of model size by GShard translates to better in-domain performance than DEMix for the 32 and 64 GPU settings, we observe the DEMix LMs consistently outperform GShard on the novel domains regardless of computational budget (Table 9). Surprisingly, GSHARD underperforms DEMIX even in-domain for the 760M parameter model (Table 8), despite being 4x larger in effective parameter count (Table 7). This suggests that domain-modularity is an important mechanism to improve model generalization, in addition to model size. We believe there is a rich area of future work to investigate how to combine token- and domain-level routing, to realize the benefits of increasing parameter count while maintaining domain modularity at scale.

Summary Our results suggest that while GSHARD is an effective method for substantially increasing model size under a fixed budget, it comes with large costs to training and evaluation efficiency, does not result in a modular LM. The lack of modularity also implies that GSHARD suffers from similar downstream issues as DENSE models, e.g., forgetting after adaptation and lack of lightweight controllability, though we leave a close exploration of those phenomena to future work.

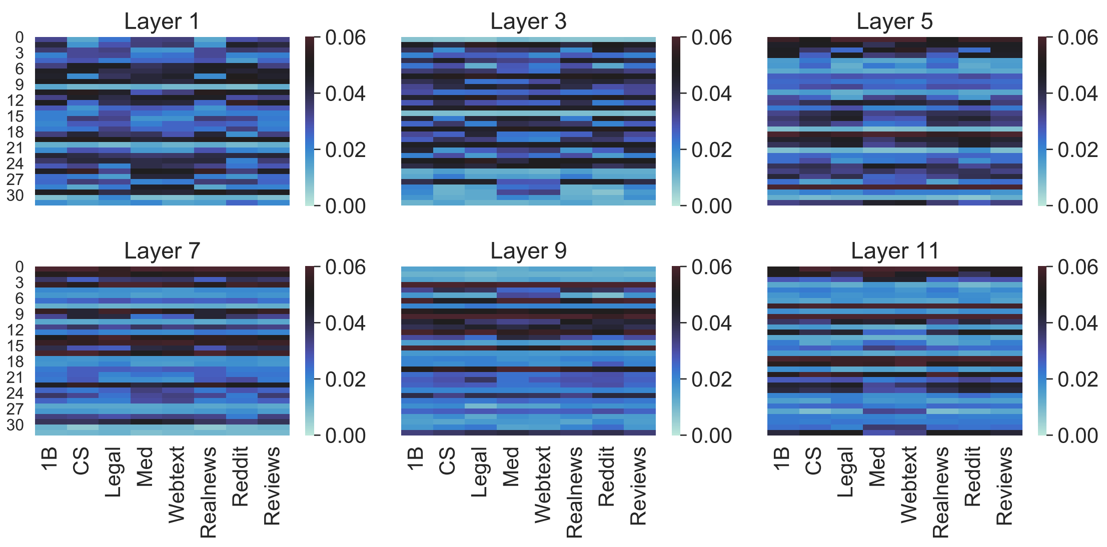


Figure 4: Average gating probabilities across domains (x-axis) for each expert (y-axis) in the expert layers of a GSHARD LM with 125M parameters per GPU. We observe high entropy of gating probabilities across experts and domains in each expert layer, with similar results in larger models.

Overall, DEMIX LMs are substantially simpler and more efficient for training and evaluation, and even outperform GSHARD (especially out of domain) despite being substantially smaller, suggesting the importance of domain modularity as an alternative mechanism to model scaling for improving generalization in LMs.

A.6 Interleaving Experiments

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. 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.

A.7 Hyperparameter Assignments

We display hyperparameter assignments for LM pretraining in Tables 13, 14,15, and 16. We set the total number of training steps based on this allocated runtime, set 8% of these steps to be warmup, 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. These settings are inspired by Lewis et al. (2021).

A.8 Per-Domain Results

We display the rest of the per-domain test results in the spreadsheets at the following link: `anonymous.com`

A.9 Domain Posterior Calculations

We track calculated domain posteriors over sequences of development data in Figure 5 (training domains) and Figure 6 (novel domains). The domain posteriors are noisier for earlier sequences, stabilizing usually after around 50 sequences. For all experiments, we conservatively use 100 sequences 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.10 Domain Posterior Heatmaps

In this section, we display heatmaps of the computed domain posteriors in Figure 7, for 100 sequences of data in the training and novel domains.

A.11 Illustration of Expert Mixing

See Figure 8 for an illustration of expert mixing.

A.12 Illustration of DEMIX-DAPT

See Figure 9 for an illustration of DEMIX-DAPT.

A.13 GPT-3 *Da-Vinci* Comparison

We conduct an experiment comparing our largest DEMIX LM with GPT-3 *Da-Vinci* from Brown

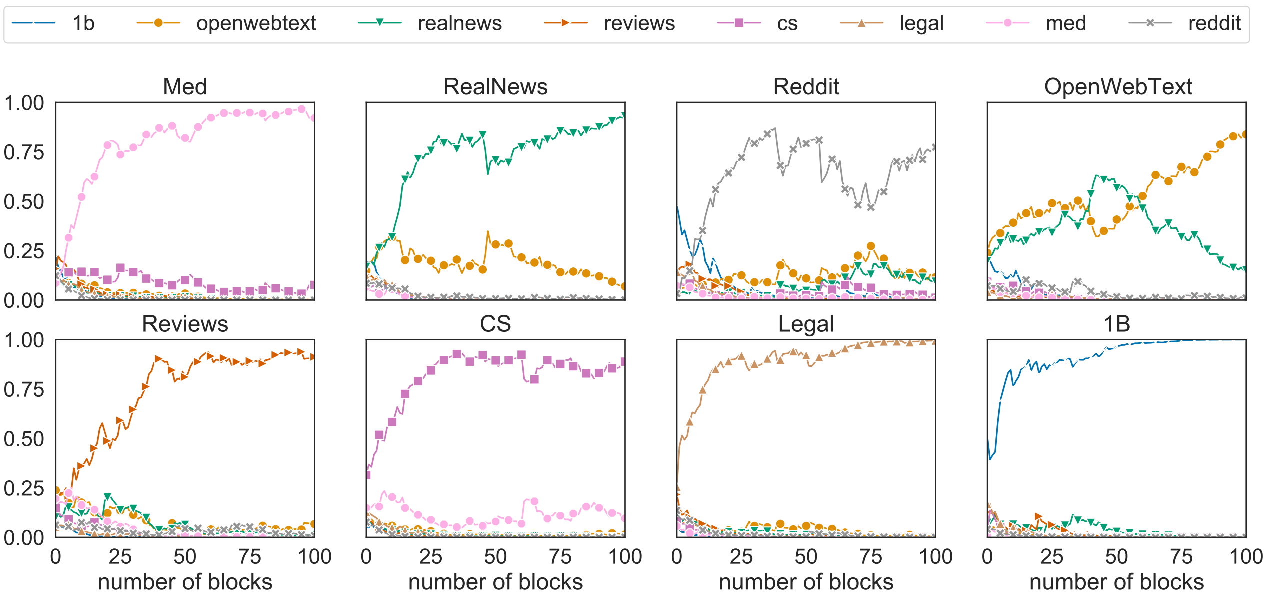


Figure 5: Calculated domain posteriors for 8 training domains.

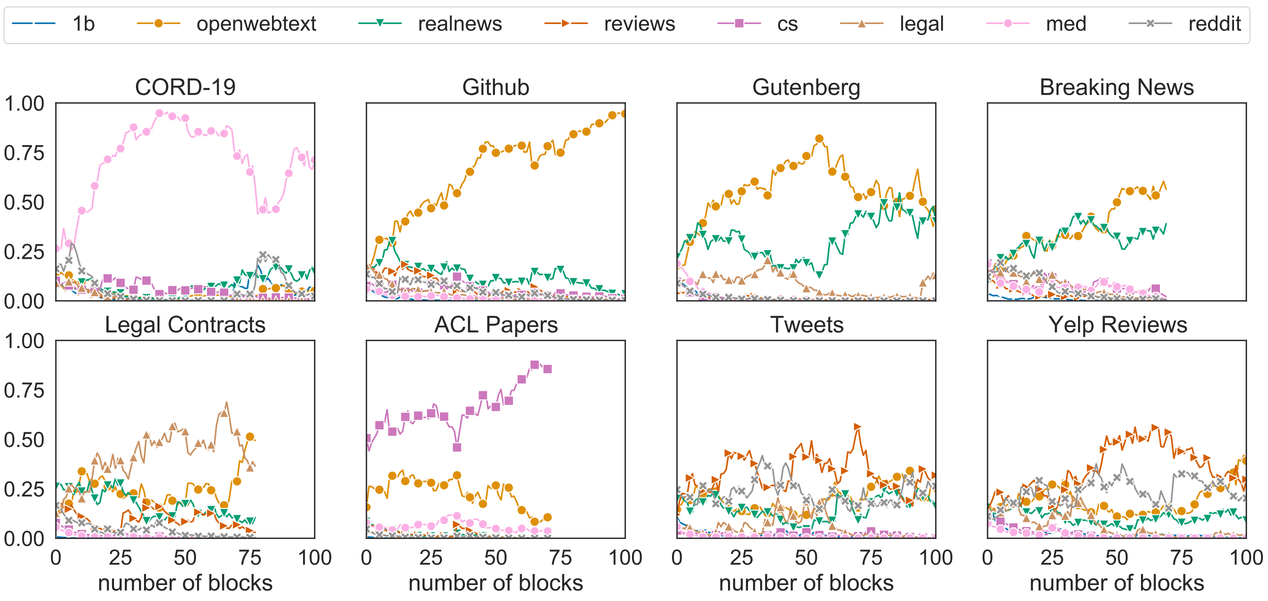


Figure 6: Calculated domain posteriors for 8 novel domains.

	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 10: 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.2 for more details on how these data were collected. All datasets are fair use for research purposes according to their original licenses. Please check `anonymous.com` for more details.

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

et al. (2020), using the zero-shot language modeling evaluation they report: Penn TreeBank (Marcus et al. 1993; Table 17). We observe that the largest DEMIX LM achieves competitive results with the GPT-3 *Da-Vinci* result with a fraction of the computation, and gives large performance boosts on this benchmark over our other DENSE baselines. These results further suggest the importance of domain modularity as a mechanism to improve generalization performance, in addition to model scaling.

A.14 Perplexity changes after DENSE-DAPT

In Table 12, 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.

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

Table 12: 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.

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 13: 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 14: 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

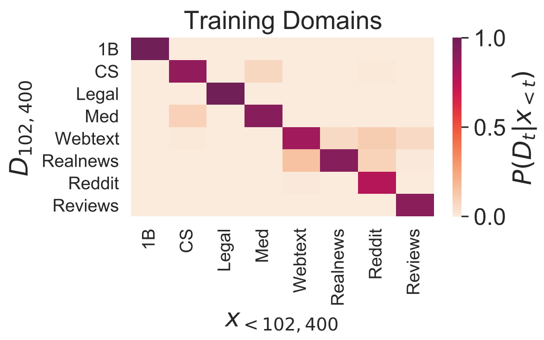
Hyperparameter	**Assignment**
architecture	GPT-3 large
tokens per sample	1024
batch size	2
number of workers	2
learning rate	[5e–4, 3e–4, 1e–4]
clip norm	0.1
gradient acculumation steps	8
number of steps	65,000
save interval updates	2,000
validation interval	1,000
number of warmup steps	5,200
learning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

Table 15: 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 16: Hyperparameters for pretraining the LM with 1.3B parameters per GPU. All hyperparameters are the same for DEMIX and DENSE training.



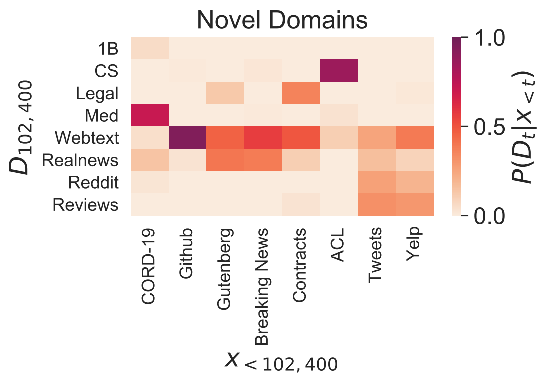


Figure 7: Estimates of posteriors $p(D_t \mid \boldsymbol{x}_{<t})$ with a DEMIX LM (1.3B parameters per GPU), after 100 sequences (i.e., 102,400 tokens) of data in training (top heatmap) and novel domains (bottom heatmap).

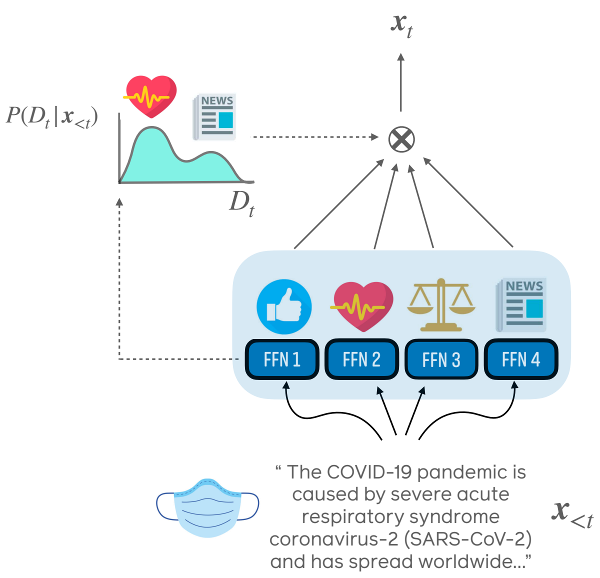

x_t
$P(D_t|x_{<t})$
NEWS
D_t
FFN 1
FFN 2
FFN 3
FFN 4
" The COVID-19 pandemic is caused by severe acute respiratory syndrome coronavirus-2 (SARS-CoV-2) and has spread worldwide..."
$x_{<t}$

Figure 8: 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 output $\boldsymbol{x}_t$.

DENSE (1.3B params per GPU)	29.4
DEMIX (cached; 1.3B params per GPU)	21.8
GPT-3 *Da-Vinci*	**20.5**

Table 17: Zero-shot perplexity on the Penn TreeBank Corpus (Marcus et al., 1993), comparing our largest DENSE and DEMIX baselines with GPT-3 *Da-Vinci*, the largest Brown et al. (2020). Our largest DEMIX LM gains a large boost in performance over DENSE baseline, approaching the performance of GPT-3 *Da-Vinci* with a fraction of the compute budget.

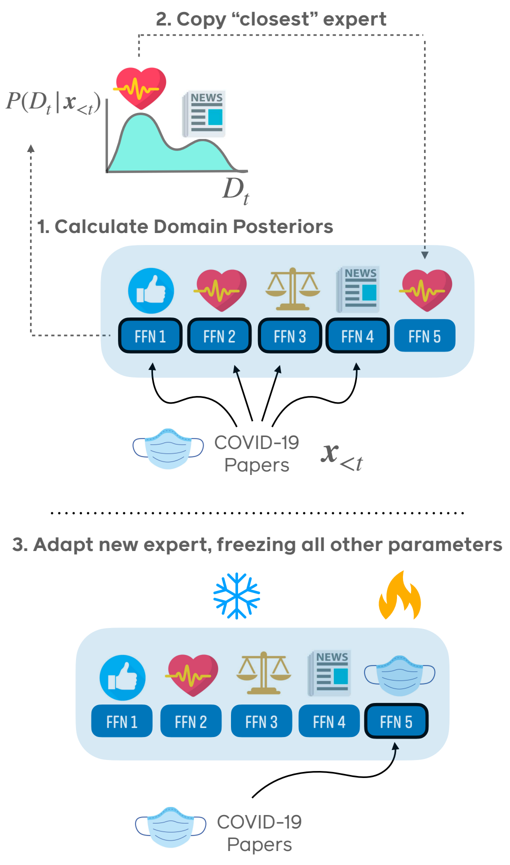


Figure 9: Illustration of DEMIX-DAPT. First, we estimate domain posteriors on a held out sample of the target domain (in this case, CORD-19). We then initialize a new expert with the parameters of the most probable expert under the domain posterior distribution. Finally, we adapt the parameters of the newly initialized expert to the target domain, keeping all other parameters in the LM frozen.