TRAINING-FREE DYNAMIC UPCYCLING OF EXPERT LANGUAGE MODELS

Eros Fanì
Gensyn

Oğuzhan Ersoy
Gensyn

ABSTRACT

Large Language Models (LLMs) have achieved remarkable performance on a wide range of specialized tasks, exhibiting strong problem-solving capabilities. However, training these models is prohibitively expensive, and they often lack domain-specific expertise because they rely on general knowledge datasets. Expertise finetuning can address this issue; however, it often leads to overspecialization, and developing a single multi-domain expert remains difficult due to diverging objectives. Furthermore, multitask training is challenging due to interference and catastrophic forgetting. Existing work proposes combining the expertise of dense models within a Mixture of Experts (MoE) architecture, although this approach still requires multitask finetuning. To address these issues, we introduce Dynamic Upcycling MoE (DUME), a novel approach that reuses dense experts trained on different domains to construct a unified MoE model. Our method builds a single multitask model that preserves the capabilities of the original dense experts without requiring additional training. DUME is both cost-efficient and scalable: by leveraging the closed-form solution of ridge regression, it eliminates the need for further optimization and enables experts to be added dynamically while maintaining the model's original performance. We demonstrate that DUME consistently outperforms baseline approaches in both causal language modeling and reasoning settings. Finally, we also show that the DUME model can be fine-tuned to further improve performance. We show that, in the causal language modeling setting, DUME can retain up to 97.6% of a dense expert model specialized in one particular domain, and that it can also surpass it in the reasoning setting, where it can achieve 102.1% of the dense expert performance. Our code is available at: github.com/gensyn-ai/dume.

1 INTRODUCTION

Since the introduction of the transformer architecture (Vaswani et al., 2017), Large Language Models (LLMs) have demonstrated remarkable capabilities across a wide range of tasks (Radford et al., 2019), including specialized applications such as code generation (Chen, 2021), mathematical problem-solving (Cobbe et al., 2021), and instruction following (Zhou et al., 2023). Their adaptability and scalability have established them as powerful tools in both research and practical applications. In principle, training a single expert model that effectively generalizes across multiple domains would be desirable; however, in practice, this can be challenging due to optimization difficulties (Yu et al., 2020), like negative transfer among heterogeneous tasks, catastrophic forgetting (McCloskey & Cohen, 1989), or because of the prohibitive computational cost of scaling such models (Hoffmann et al., 2022), and the communication costs or potential privacy challenges (Zhang et al., 2020) associated with aggregating large, multi-domain datasets.

Recent works on model merging (He et al., 2025) seek to construct unified models with multi-domain and multi-task capabilities by combining multiple specialized experts (that are trained independently in parallel). These experts share the same architecture but differ in their fine-tuned parameters, and are merged into a single model typically through linear operations in parameter space (Ilharco et al., 2022; Matena & Raffel, 2022; Wang et al., 2024; Yadav et al., 2023), such as simple weight averaging or task arithmetic. This approach is appealing because it avoids retraining from scratch, instead leveraging existing specialized models to produce a multitask expert. However, naïve aggregation can lead to destructive interference (Yadav et al., 2023), as experts trained on different domains may correspond to distinct loss landscapes; consequently, their parameters

Corresponding author: Eros Fanì, email: eros@gensyn.ai

may be optimized for incompatible objectives, resulting in merged models that are suboptimal or detrimental across all tasks.

To mitigate interference, another approach is to ensemble these domain experts rather than merge them into a single model; see Branch-Train-Merge (BTM) (Li et al., 2022). BTM ensembling demonstrates strong performance; however, the inference cost increases linearly with the number of experts relative to a single dense model. Branch-Train-Mix (BTX) (Sukhbaatar et al., 2024) addresses this issue by merging the experts into a single Mixture of Experts (MoE) with sparse routing. MoE architectures (Shazeer et al., 2017; Jacobs et al., 1991; Fedus et al., 2022; Dai et al., 2024; Jiang et al., 2024; Cai et al., 2025) allow sparse parameter activation, enabling conditional computation by activating only a subset of model parameters for each input, jointly reducing the effective computational cost during inference. In BTX, the expert models are merged as follows: the non-MLP parameters of the expert models are averaged, whereas the MLP parameters are used as separate MoE experts, which are sparsely routed by a gating network (or router).

After constructing the combined MoE model, BTX requires finetuning it to calibrate the randomly initialized router. This could constitute a limitation in certain settings where such finetuning is unfeasible, for instance, because it requires to aggregate domain data into a single centralized node to train the final MoE model, which could raise concerns about privacy, or simply because of computational costs. To address issues about additional computational cost and domain data privacy, we propose Dynamic Upcycling MoE (DUME). Similar to BTX, DUME combines multiple dense experts into a single MoE architecture; however, it differs fundamentally in how the routing mechanism is initialized. Specifically, inspired by the applications of ridge regression in the horizontal federated learning (Afonin & Karimireddy, 2021; Fanì et al., 2025; 2024) and vertical federated learning (Cai et al., 2022; Huang et al., 2022) literature, DUME initializes the gating networks of the MoE blocks using the closed-form solution of the ridge regression problem (Stigler, 1981). With this initialization, we demonstrate that no further training is necessary to approach, or even surpass, the performance of the best dense experts for each domain. Moreover, even if router fine-tuning is possible, our method remains superior because it provides a substantially better router initialization.

Our main contributions can be summarized as follows:

- We propose DUME, an embarrassingly parallel method that aggregates existing dense experts into a single, multi-domain MoE model, requiring no further training. Our method also allows the addition of more experts later in an incremental learning setting without performance degradation.
- We demonstrate, with experiments in various domains and with 115M and 3B LLama models (Touvron et al., 2023), that DUME outperforms the baselines in both the causal language modeling and the reasoning scenarios.
- We demonstrate that, when a unified multi-domain dataset is available, our model can be further trained to outperform the baselines by a larger margin.

2 METHOD

In this section we present our method. First, in Section 2.1, we briefly recall what ridge regression is and how can it be used for classification purposes, as our DUME relies on the ridge regression solution to correctly route tokens to the best domain experts. Then, in Section 2.2, we explain the concept of experts *MoErging*, a procedure introduced previously and used in our method. Finally, Section 2.3 describes how DUME initializes the router parameters exploiting the closed-form solution of ridge regression.

2.1 BACKGROUND ON RIDGE REGRESSION

Let $f(x; W) = W^\top x$ be a linear predictor parameterized by $W \in \mathbb{R}^{H\times C}$, which maps inputs $x \in \mathbb{R}^H$ to targets $y \in \mathbb{R}^C$. For a given dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$, the following ridge regression problem (Stigler, 1981):

$$\underset{W\in\mathbb{R}^{H\times C}}{\arg\min} \sum_{i=1}^{n} \|f(x_i; W) - y_i\|^2 + \lambda \|W\|^2, \tag{1}$$

controlled by a Tikhonov hyperparameter $\lambda \in \mathbb{R}^+$, admits the closed-form solution:

$$W^* = (X^\top X + \lambda I_H)^{-1} X^\top Y, \tag{2}$$

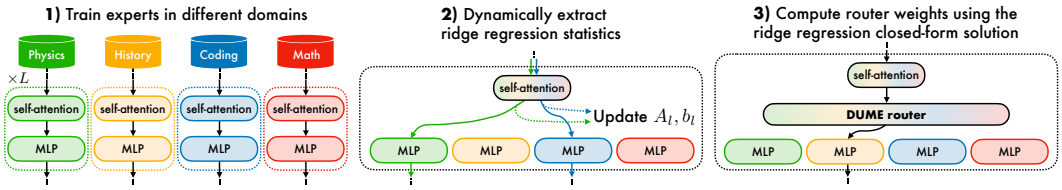

Figure 1: Overview of the DUME method. For simplicity and illustration purposes, we only show a single transformer block l among the total L blocks. 1) Dense experts are trained on different domains. 2) MoErging phase and ridge regression statistics extraction: the parameters of all the layers but the MLP layers, including the self-attention layers, are averaged. Each input token from all the domains is forwarded to the expert corresponding to its domain label. The feature maps extracted from the layers preceding the MLP blocks are used to update the A_l and b_l matrices. 3) The parameters W_l^* are computed via Equation 2 using the matrices computed at the previous step.

where $X \in \mathbb{R}^{n \times H}$ and $Y \in \mathbb{R}^{n \times C}$ are constructed by stacking the input and target vectors, respectively, and I_H is the identity matrix of size H. The term $\lambda \|W\|^2$ makes the optimization problem strictly convex, guaranteeing a unique solution (Shawe-Taylor & Cristianini, 2004). If the targets are one-hot encoding vectors, the problem in Equation 1 can be effectively purposed to solve classification tasks (Rifkin et al., 2003). Importantly, thanks to Equation 2, it is possible to avoid costly, stochastic optimization algorithms to solve Equation 1.

If the data is sequentially available in B batches of any size, it is possible to sequentially update the $X^\top X$ and $X^\top Y$ matrices as more batches become available, and finally compute the optimal solution in Equation 2 (Björck, 1996; Fanì et al., 2025):

$$W^* = \left(\sum_{i=1}^{B} X_i^\top X_i + \lambda I_H \right)^{-1} \sum_{i=1}^{B} X_i^\top Y_i, \tag{3}$$

where $X_i \in \mathbb{R}^{n_i \times H}$ and $Y_i \in \mathbb{R}^{n_i \times C}$ are constructed by stacking the input and target vectors within batch i, respectively, with $\sum_{i=1}^{B} n_i = n$. The matrices $X^\top X \in \mathbb{R}^{H \times H}$ and $X^\top Y \in \mathbb{R}^{H \times C}$ are independent of the number of samples per batch, allowing for batches of different sizes.

2.2 MoErging

Both our method and some of the baselines in this work share a common merging step, which we call *MoErging*. Consider a set of *dense experts* $\{f_d\}_{d=1}^{D}$, sharing a common transformer architecture constituted by an embedding layer, a sequence of L transformer blocks, and the output head. Each transformer block is composed of an attention block and an MLP block. We refer to the l-th MLP block of the dense expert f_d as the *MoE expert* $\mathcal{E}_{dl}$. We also assume that each dense expert f_d has been trained to solve different language tasks using the local domain dataset $\mathcal{D}_d$ of size n_d.

To construct a unified multi-domain expert MoE model F that leverages the capabilities of pre-trained dense experts f_d while circumventing the need for expensive multi-task re-training, the MoErging approach averages the parameters of all layers of the dense experts except the MLP layers. The rationale for excluding MLP layers from averaging is that they are less specialized than the MLP layers, as noted by (Sukhbaatar et al., 2024). The MLP layers are, instead, combined into MoE blocks as follows:

$$\text{MoE}_l(x) = \sum_{d=1}^{D} [g(x; W_l)]_d \, \mathcal{E}_{dl}(x), \tag{4}$$

where MoE_l is the l-th MoE block of F, W_l are the parameters of the router of the l-th MoE block, and g is a routing function which we define as $g(x; W_l) = \text{Top-k}(\text{Softmax}(W_l^\top x))$, $W_l \in \mathbb{R}^{H \times D}$, where H is the hidden dimension of the dense experts. To have the same forward costs as the dense experts, we set $\text{k} = 1$ (see Appendix A for further discussion on the top-k routing of DUME).

2.3 Dynamic Upcycling MoE (DUME)

We propose Dynamic Upcycling MoE (DUME), which constructs a unified multi-domain expert MoE model F by exploiting the capabilities of the already-trained dense experts f_d while avoiding costly multi-task re-training. Figure 1 presents an overview of our method.

After the MoErging phase, DUME uses the closed-form ridge regression solution in Equation 2 along with the training data's domain labels to initialize the router parameters. Consider the first MoE block, MoE_1. We initialize two matrices $A_{1,0} = 0_{H\times H}$ and $b_{1,0} = 0_{H\times D}$, where $0_{a\times c}$ is a matrix of zeros of size $a \times c$. The purpose of these two matrices is to iteratively reconstruct the final $X^\top X$ and $X^\top Y$ matrices of Equation 2 (using the notation in Section 2.1). Given an input text x_i from domain d, and the matrix $y_i \in \mathbb{R}^{T_i\times D}$ which is the one-hot encoding matrix associated with the input x_i, whose column corresponding to the domain indices is a vector of ones, and all the other values are zeros, *i.e.,* $[y_i]_{jk} = 1$ if $k = d$, 0 if $k \neq d$, the output of all the layers preceding MoE_1, which we call $F_{<1}(x_i)$, has shape $T_i \times H$, where T_i is the number of tokens in x_i. To compute the optimal ridge regression weights of the first router, we accumulate the contribution of each new token of an input text x_i, in any order, into the A_1 and b_1 matrices:

$$\begin{cases} A_{1,t} = A_{1,t-1} + F_{<1}(x_i)^\top F_{<1}(x_i) \\ b_{1,t} = b_{1,t-1} + F_{<1}(x_i)^\top y_i, \end{cases} \tag{5}$$

where the index t indicates the number of recursive updates performed so far. We can continue accumulating the contribution of each sample of text in each domain until all the datasets have been forwarded through $F_{<1}$, and use the final A_1 and b_1 matrices to compute the optimal ridge regression parameters for the first router:

$$W_1^* = (A_1 + \lambda I_H)^{-1} b_1. \tag{6}$$

If the input text is processed in batches X_i of size B, the feature map $F_{<1}(X_i)$ and the stacked one-hot tensor Y_i would both have shape $B \times T_i \times H$[1]. In this case, the matrices A_1 and b_1 can analogously be accumulated by reshaping $F_{<1}(X_i)$ and Y_i to the dimension $BT_i \times H$. Importantly, thanks to the sequential property of ridge regression in Equation 3 (Björck, 1996), the update rule in 5 guarantees that the final parameters are exactly the same optimal parameters as if we computed the ridge regression solution using Equation 2. Finally, to address possible domain unbalanced distributions (Fanì et al., 2024; 2025; Legate et al., 2023), we normalize W_1^* by dividing each column by its norm: $[W_1^*]_{:,c} = [W_1^*]_{:,c} / \|[W_1^*]_{:,c}\|$.

In general, all the MoE routers' parameters and the final optimal weights can be computed analogously. However, a layer l would require all the MoE routers $l > 1$ to be initialized beforehand to compute $F_{<l}(x_i)$, *i.e.,* the output of all the layers preceding MoE_l. A possible solution is to compute the optimal weights for the routers sequentially, from router 1 to L. However, this strategy would require increasing the number of forward passes by a factor of L, since one would need to forward the entire datasets from all domains every time for each new router. To avoid this issue, we instead forward the text from all the domains datasets only once, update all the matrices $A_{l,t}$ and $b_{l,t}$ simultaneously and, instead of using the gating layer to forward each feature map to the MoE experts, we instead deterministically forward each feature map to the MoE expert corresponding to the domain of the input text. The general update rule of the matrices $A_{l,t}$ and $b_{l,t}$ is:

$$\begin{cases} A_{l,t} = A_{l,t-1} + F_{d,<l}(x_i)^\top F_{d,<l}(x_i) \\ b_{l,t} = b_{l,t-1} + F_{d,<l}(x_i)^\top y_i, \end{cases} \implies W_l^* = (A_l + \lambda I_H)^{-1} b_l, \tag{7}$$

where $A_{l,0} = 0_{H\times H}$, $b_{l,0} = 0_{H\times D}$, $F_{d,<l}(x_i)$ indicates the feature maps extracted with this strategy for domain d. In particular, note that, for $d = 1$ and $\forall l$, $F_{d,<l}(x_i) = F_{<l}(x_i)$. Finally, the final weights of each router are normalized: $[W_l^*]_{:,c} = [W_l^*]_{:,c} / \|[W_l^*]_{:,c}\|$.

For the MoE routers $l > 1$, the computed parameters are not optimal in the sense of ridge regression, because the input space used to compute the ridge regression solution depends on the domain of the input text. In other words, given a router $l > 1$, each input of the domain dataset $\mathcal{D}_d$ is forwarded to the MoE experts $\mathcal{E}_{dl'}$ for all the preceding MoE blocks $l' < l$, which constitutes a different model architecture from the case of domain $d' \neq d$, for which the MoE experts choice is $\mathcal{E}_{d'l'}$ for all $l' < l$. However, we argue that the effects of having different input spaces on the final solution are negligible, because *a)* the latent spaces would still have the same dimensionalities, and *b)* the self-attention layers, which are the same for all the domains as their parameters are averaged from the original dense experts, are structurally well-suited to encode relational and syntax knowledge which prevails over the domain knowledge in shaping the distribution of the latent feature maps. Indeed, in Section 3, we show that our approximate ridge regression solution already outperforms existing methods. We present the complete pseudo-code in Algorithm 1.

[1]The number of tokens T_i for each input text within a batch can be made the same by methods like padding.

Algorithm 1 - Dynamic Upcycling MoE (DUME)

Inputs: dense experts $\mathcal{E}_d, \forall d = 1, \ldots, D$
$\forall d = 1, \ldots, D$, train the dense expert f_d using the domain dataset $\mathcal{D}_d$.
Perform the MoErging phase discussed in Section 2.2.
Initialize: $A_l = 0_{H \times H}$, $b_l = 0_{H \times D}$ $\forall l = 1, \ldots, L$.
for each $d = 1, \ldots, D$, *in parallel* **do**
  for each $i = 1, \ldots, B$, $l = 1, \ldots, L$ **do**
    $A_l \leftarrow A_l + F_{d,<l}(X_i)^\top F_{d,<l}(X_i)$
    $b_l \leftarrow b_l + F_{d,<l}(X_i)^\top Y_i$
  end for
end for
for each $l = 1, \ldots, L$ **do**
  $W_l^* = (A_l + \lambda I_h)^{-1} b_l$
  $[W_l^*]_{:,c} = [W_l^*]_{:,c} / \|[W_l^*]_{:,c}\|$
end for
Return MoE model F with the parameters of the routers initialized using W_l^*, $\forall l = 1, \ldots, L$

Unlike the existing methods, which require additional training, our method only necessitates forwarding the domain datasets once, thereby dramatically reducing costs compared with further multi-task training. Moreover, in our method, the datasets can be accessed sequentially, as the ridge regression solution is schedule-invariant (Fanì et al., 2024; Wang et al., 2022). In other words, if the domain datasets are made available sequentially, DUME enables continual learning, as it can dynamically update the routers without incurring catastrophic forgetting (McCloskey & Cohen, 1989).

Extensions. Note that, although routers initialized with our DUME are already sufficient to surpass the baselines (as we show in Section 3), they can be further finetuned. We refer to our method with additional router finetuning as DUME+. Finally, in Section 3.3, we show that our method generalizes even when the distribution of the datasets used to update the A_l and b_l matrices differs from the test distribution. We refer to this variation as DUME out-of-distribution (DUME OOD).

Computational costs. The computational costs of Equation 2 are $\mathcal{O}(nH^2)+\mathcal{O}(H^3)+\mathcal{O}(nHC)+\mathcal{O}(H^2C)$, which could be further optimized with techniques like Cholesky decomposition (Cholesky, 2005). When we collect all the updates from Equation 7, the total computational costs are the same as if we had a single batch containing all the inputs. In DUME, the number of domains is negligible, and we need to substitute n with nT. For simplicity, we assume a constant number of tokens T per data point. Moreover, we need to compute the ridge regression solution once for every MoE block. Therefore, the final computational costs for DUME are the forward costs of the inputs plus the cost of computing the ridge regression solutions, which is $\mathcal{O}(nTH^2 + H^3)$. Note that, unlike BTX, our algorithm does not require backpropagation because it does not require additional training. Moreover, in our experiments, we found that the cost of computing ridge regression solutions is negligible relative to the forward and backpropagation costs, as it is expected, given that the costs of computing the ridge regression solutions only involves matrix multiplications and inversion of relatively small matrices (recall that the size of the matrix that has to be inverted is the hidden size of the model, which is fixed). Finally, in Appendix A, we also show that a smaller token window per paragraph is sufficient to retain most of the performance, jointly reducing computational costs.

3 Experiments

3.1 Experimental Setup

Our experiments are conducted in two settings: *causal language modeling (CLM)*, in which the task across all domain datasets is next-token prediction, and *reasoning*, in which each domain has its own distinct reasoning task. We run our experiments on up to 4 NVIDIA H100 GPUs with 80 GB of HBM3 memory. Below, we present our implementation choices and default hyperparameters. All of our default values for the hyperparameters are the result of our hyperparameter tuning. Further details are provided in Appendix A.

Details on the CLM experiments. For CLM experiments, we use the model and the domain datasets from (Ersoy et al., 2025), with the corresponding training hyperparameters. First, we

Table 1: Datasets used for evaluating the reasoning experiments, and their evaluation metrics.

Dataset	Description	Metric
HumanEval (Chen, 2021)	Code synthesis benchmark consisting of programming problems with natural language descriptions and unit tests.	pass@1: proportion of correctly solved problems.
GSM8k (Cobbe et al., 2021)	Grade-school math reasoning dataset composed of linguistically diverse word problems requiring multi-step reasoning.	Accuracy: proportion of problems for which the model predicts the correct final answer.
M_ARC (Lai et al., 2023)	Multilingual ARC challenge question-answering dataset used in Okapi evaluations, focusing on multiple-choice reasoning tasks.	Accuracy: percentage of correct answers.
IFEval (Zhou et al., 2023)	Instruction-Following Evaluation benchmark for large language models with prompts containing explicit, verifiable constraints.	Task compliance rate: percentage of outputs satisfying all instructions.

pre-trained a seed 115M Llama model (Touvron et al., 2023) on the OpenWebText corpus dataset (Gokaslan & Cohen, 2019), using a learning rate $= 6 \cdot 10^{-4}$, with a cosine annealing scheduler with 1000 warmup steps, a batch size $= 16$ paragraphs of 1024 padded tokens, with 4 gradient accumulation steps and a total of 20000 training iterations. We then trained 5 experts, starting from the same pre-trained seed model, on the following M2D2 domains (Reid et al., 2022): coding, mathematics, physics, history and events, and philosophy and thinking. We used a learning rate of $6 \cdot 10^{-5}$, a cosine annealing scheduler with 50 warmup steps, a batch size of 16 paragraphs of 1024 padded tokens, 4 gradient accumulation steps, and a total of 1000 training iterations. For our DUME experiments, our default choice for the hyperparameters is $\lambda = 0.01$ and $k = 1$. For BTX training, we use a learning rate of 10^{-4}, $k = 2$ as the default values for top-k routing, and 1000 training iterations, as we found these values perform best for BTX. For DUME+, we use a learning rate of 10^{-5}, $k = 1$, and 1000 training iterations. Similar to (Sukhbaatar et al., 2024), we also attempted to incorporate load balancing into our BTX and DUME+ training, but its effects were negligible in our settings, so we decided not to use it. Finally, for our DUME+ experiments, we also attempted to modify the routers' softmax temperature, as motivated in (Fanì et al., 2024), but we observed no improvement and ultimately kept the temperature fixed at 1.

Details on the reasoning experiments. For the reasoning experiments, we downloaded four domain-specific LLama-3B experts (Touvron et al., 2023) from Hugging Face, uploaded by the authors of (He et al., 2025), for the following domains: coding, mathematics, multilingual understanding, and instruction following. We perform evaluation on the following datasets: HumanEval (Chen, 2021) for coding, GSM8k (Cobbe et al., 2021) for mathematics, M_ARC (Lai et al., 2023) for multilingual understanding, and IFEval (Zhou et al., 2023) for instruction following. We specify the evaluation metrics for these datasets in Table 1. For all these datasets, the metrics range from 0 (worst performance) to 100 (best performance). For the DUME OOD experiment in Section 3.3, we extract the ridge regression statistics from some of the datasets where the experts were originally trained, namely Magicoder (Wei et al., 2023) for coding, DART-Math (Tong et al., 2024) for mathematics, Aya (Singh et al., 2024) for multilingual understanding, and TULU-3 persona IF (Lambert et al., 2024) for instruction following.

Metrics. The CLM experiments have been evaluated using the normalized average perplexity score on all five domains. In particular, let p_d be the perplexity score of a certain experiment on domain d, and let $\hat{p}_d$ be the perplexity score of the dense expert $\mathcal{E}_d$ on domain d, which is expected to achieve the best performance (lowest perplexity) on domain d among all the other models. We define the normalized average perplexity score $\bar{p} \in \mathbb{R}^+$ for a given experiment as $\bar{p} = \frac{100}{D} \sum_{d=1}^{D} \frac{\hat{p}_d}{p_d}$. Similarly, we evaluate the reasoning experiments using the normalized average performance score $\bar{p} \in \mathbb{R}^+$, defined as $\bar{p} = \frac{100}{D} \sum_{d=1}^{D} \frac{p_d}{\hat{p}_d}$, where, in this case, we inverted the fraction in the sum since the best performance corresponds to the highest values for all the metrics of the reasoning datasets. Please note that, with these definitions, it is possible (and desirable) that the value of the metrics for a particular domain exceeds 100, in which case it means that the experiment surpasses the dense expert in the corresponding domain.

In the following sections, we present our CLM and reasoning experiments. We provide the results in the format *mean* $\pm$ *std*, as we run these experiments with three different random seeds.

Table 2: CLM experiments. The best and second best results, excluding the results of the experts in the corresponding domains, for which the score is 100.0 by definition, and the Oracle baseline, which, unlike the other methods, uses the domain label at test time, are highlighted in **bold** and underlined, respectively. Results of the aggregated models that surpass the Oracle results are highlighted in green.

Method	Coding	Mathematics	Physics	History	Philosophy	Average
Coding expert	100.0 ± 0.0	66.8 ± 0.0	78.1 ± 0.0	79.2 ± 0.0	81.1 ± 0.0	81.0 ± 0.0
Math expert	89.9 ± 0.0	100.0 ± 0.0	75.2 ± 0.0	69.4 ± 0.0	71.4 ± 0.0	81.2 ± 0.0
Physics expert	80.2 ± 0.0	58.8 ± 0.0	100.0 ± 0.0	80.0 ± 0.0	81.8 ± 0.0	80.2 ± 0.0
History expert	63.8 ± 0.0	36.8 ± 0.0	64.7 ± 0.0	100.0 ± 0.0	98.0 ± 0.0	72.7 ± 0.0
Philosophy expert	66.9 ± 0.0	40.2 ± 0.0	66.3 ± 0.0	95.3 ± 0.0	100.0 ± 0.0	73.7 ± 0.0
Model averaging	86.8 ± 0.0	63.1 ± 0.0	83.7 ± 0.0	90.4 ± 0.1	93.1 ± 0.0	83.4 ± 0.0
Oracle	95.9 ± 0.0	87.3 ± 0.0	95.1 ± 0.0	97.1 ± 0.0	98.6 ± 0.0	94.8 ± 0.0
Random routing	86.3 ± 2.0	62.8 ± 2.6	82.4 ± 1.0	88.7 ± 0.6	91.5 ± 0.6	82.4 ± 0.9
BTX	91.7 ± 0.1	85.7 ± 0.5	89.9 ± 1.3	95.5 ± 0.2	96.6 ± 0.3	91.9 ± 0.5
DUME (ours)	**94.5 ± 0.0**	82.3 ± 0.0	92.2 ± 0.0	96.9 ± 0.0	98.2 ± 0.0	92.8 ± 0.0
DUME+ (ours)	94.1 ± 0.0	**85.6 ± 0.0**	**94.0 ± 0.0**	**97.6 ± 0.0**	**98.4 ± 0.0**	**93.9 ± 0.0**

3.2 CLM EXPERIMENTS

Table 2 presents the results for the CLM experiments. We compare our DUME and DUME+ methods with the following baselines across five distinct domains (Coding, Mathematics, Physics, History, and Philosophy): (i) *"Domain" experts* that are the dense experts trained on a special "domain" dataset, (ii) *Model averaging* is the model obtained by averaging the dense experts, (iii) *MoErging* models are MoEs constructed from the dense experts. *Oracle* is a MoE model initialized with MoErging. In this approach, the tokens are forwarded to the MoE experts corresponding to the batch domain label. The main limitation of this baseline is the requirement for domain labels at test time, which may not be available if the test dataset comprises multiple distributions or tasks. On the one hand, the rule for directing tokens to MoE experts could be overly rigid and may not effectively differentiate among individual tokens, a hallmark of standard MoEs. On the other hand, this could constitute an advantage in forwarding the tokens to the most appropriate expert, as our experiments also show. *Random routing* is a MoE model initialized with MoErging and, like Oracle, it does not perform additional training. Unlike Oracle, this baseline includes routers, but they are initialized at random. Therefore, each token is randomly forwarded to one of the domain MoE experts. This baseline is included as a lower-bound.

As it is possible to observe, both DUME and DUME+ surpass the baselines, approaching the Oracle, without necessitating domain labels at test time. In particular, DUME is already sufficient to surpass BTX without requiring any additional training. Finally, both DUME and DUME+ retain between 82% and 98% of the corresponding dense expert-domain performance.

3.3 REASONING EXPERIMENTS

Table 3 shows the results of the reasoning experiments. The baselines are defined as in Section 3.2, with the addition of the DUME OOD. Here, we omit baselines that require fine-tuning due to a mismatch between the complex reasoning tasks and the fine-tuning objectives, as well as because some datasets were too small (especially for coding and instruction-following tasks). However, for the sake of a complete comparison, here are the BTX results across the 4 tasks (in the order shown in the table): 6.0%, 95.8%, 51.8%, and 51.9%. Future work will explore alternative router training, such as multitask reinforcement learning.

Results show that our method performs well even when the distribution of the datasets from which DUME collects ridge regression statistics differs from the test distribution. Indeed, both DUME and DUME OOD outperform the Oracle in some domains and even the dense experts in their respective domains, with DUME performing best among all baselines. Moreover, DUME OOD approaches the Oracle average performance, demonstrating that our method can be applied to OOD settings.

4 RELATED WORK

This section complements the related works proposed in Section 1.

Table 3: Reasoning experiments. The best and second best results, excluding the results of the experts in the corresponding domains, for which the score is 100.0 by definition, and the Oracle baseline, which, unlike the other methods, uses the domain label at test time, are highlighted in **bold** and underlined, respectively. Results of the aggregated models that surpass the Oracle results are highlighted in green.

Method	Mathematics	Multilingual	Coding	Instr. following	Average
Math expert	100.0 ± 0.0	95.9 ± 0.0	86.0 ± 9.0	46.5 ± 3.2	82.1 ± 2.4
Multilingual expert	6.1 ± 0.8	100.0 ± 0.0	77.9 ± 8.5	21.1 ± 0.8	51.3 ± 1.8
Coding expert	12.0 ± 0.2	99.3 ± 0.0	100.0 ± 0.0	51.1 ± 5.1	65.6 ± 1.3
Instr. following expert	28.2 ± 1.2	99.8 ± 0.0	79.5 ± 4.6	100.0 ± 0.0	76.9 ± 1.4
Model averaging	44.7 ± 3.1	100.1 ± 0.0	97.6 ± 15.4	59.0 ± 5.6	75.3 ± 4.7
Oracle	94.3 ± 7.0	100.2 ± 0.0	94.0 ± 19.7	85.3 ± 3.4	93.4 ± 4.9
Random routing	39.4 ± 2.6	99.8 ± 0.0	87.0 ± 15.2	57.0 ± 3.8	70.8 ± 3.7
DUME (ours)	**91.1** ± 2.2	**100.4** ± 0.0	**102.1** ± 5.9	**87.2** ± 1.0	**95.2** ± 1.7
DUME OOD (ours)	79.1 ± 2.0	**100.4** ± 0.0	100.9 ± 1.3	83.1 ± 1.0	90.9 ± 0.1

Sparse and modular architectures. Sparse architectures, such as MoEs (Shazeer et al., 2017; Jacobs et al., 1991), have emerged as a key technique for scaling LLMs with efficient compute. MoE models activate only a subset of experts per token, enabling extremely large parameter counts while controlling inference cost. Instruction-tuned MoE models (Shen et al., 2023; Zhu et al., 2025) have been shown to outperform dense counterparts with reduced FLOPs, highlighting the potential of sparse conditional computation in LLMs. Closely related to our work, DEMix (Gururangan et al., 2022) selectively mixes the feature maps of the experts based on domain labels. However, unlike our method, all experts are activated at inference time and mixed according to weights estimated from a validation set. Similarly, (Pfeiffer et al., 2022; Kudugunta et al., 2021) propose multilingual and multi-task expert architectures with language-specific and task-conditioned routing.

Embarrassingly parallel training. Efficient training of large models across heterogeneous corpora has motivated embarrassingly parallel frameworks that minimize costly synchronization. Branch-Train-Merge (BTM) (Li et al., 2022) decomposes LLM training into independent expert language models (ELMs), each trained on a domain subset. By branching from a seed model and ensembling, BTM improves in- and out-of-domain perplexity with minimal inter-worker communication. (Ersoy et al., 2025) builds on BTM by exploring the effects of introducing heterogeneity in the number of training iterations and parameter count of the experts, finding that heterogeneous ensembles almost always achieve lower perplexities than homogeneous baselines. Branch-Train-MiX (BTX) (Sukhbaatar et al., 2024) integrates the BTM pipeline with MoE architectures: after expert training, BTX constructs a single MoE by integrating the experts' feedforward parameters as MoE experts and averaging the remaining parameters, then fine-tunes the MoE to learn routing. This approach retains embarrassingly parallel scalability during expert training while yielding unified sparse models with effective expert selection at inference. Finally, Branch-Train-Stitch (BTS) (Zhang et al., 2025) builds on parallel expert training by stitching domain experts into a generalist model via adding stitch layers that require further training steps.

5 Conclusion

In this work, we presented DUME, a new embarrassingly parallel method that upcycles dense experts trained on different domains to construct a unified Mixture-of-Experts model. Contrary to existing methods, DUME does not require any additional multitask or multidomain training, as it initializes all final MoE parameters at minimal cost by leveraging the sequential properties of ridge regression and simple model averaging. In addition to surpassing existing baselines, in the post training scenario, DUME even outperforms the Oracle method, which has access to the test-domain labels. We show that the final MoE model can be trained further with DUME+.

A possible limitation of DUME and DUME+ is that, although they do not require domain labels at inference time, they require them in the training dataset, which may be unavailable if different training nodes have access to domain-uniform dataset partitions. Future works may explore the applicability of DUME and DUME+ in scenarios with more domains and with larger models.

REFERENCES

Andrei Afonin and Sai Praneeth Karimireddy. Towards model agnostic federated learning using knowledge distillation. *arXiv preprint arXiv:2110.15210*, 2021.

Åke Björck. *Numerical methods for least squares problems*. SIAM, 1996.

Jianping Cai, Ximeng Liu, Zhiyong Yu, Kun Guo, and Jiayin Li. Efficient vertical federated learning method for ridge regression of large-scale samples. *IEEE Transactions on Emerging Topics in Computing*, 11(2):511–526, 2022.

Weilin Cai, Juyong Jiang, Fan Wang, Jing Tang, Sunghun Kim, and Jiayi Huang. A survey on mixture of experts in large language models. *IEEE Transactions on Knowledge and Data Engineering*, 2025.

Mark Chen. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

André-Louis Cholesky. Sur la résolution numérique des systèmes d'équations linéaires. *Bulletin de la Sabix*, 39:81–95, 2005. Published posthumously.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Damai Dai, Chengqi Deng, Chenggang Zhao, RX Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Yu Wu, et al. Deepseekmoe: Towards ultimate expert specialization in mixture-of-experts language models. *arXiv preprint arXiv:2401.06066*, 2024.

Oguzhan Ersoy, Jari Kolehmainen, and Gabriel Passamani Andrade. HDEE: Heterogeneous domain expert ensemble. In *ICLR 2025 Workshop on Modularity for Collaborative, Decentralized, and Continual Deep Learning*, 2025. URL https://openreview.net/forum?id=5ukL6nPcYe.

Eros Fanì, Raffaello Camoriano, Barbara Caputo, and Marco Ciccone. Accelerating heterogeneous federated learning with closed-form classifiers. *Forty-first International Conference on Machine Learning (ICML)*, 2024.

Eros Fanì, Raffaello Camoriano, Barbara Caputo, and Marco Ciccone. Resource-efficient personalization in federated learning with closed-form classifiers. *IEEE Access*, 2025.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

Aaron Gokaslan and Vanya Cohen. Openwebtext corpus. http://Skylion007.github.io/OpenWebTextCorpus, 2019.

Suchin Gururangan, Mike Lewis, Ari Holtzman, Noah A Smith, and Luke Zettlemoyer. Demix layers: Disentangling domains for modular language modeling. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 5557–5576, 2022.

Yifei He, Siqi Zeng, Yuzheng Hu, Rui Yang, Tong Zhang, and Han Zhao. Mergebench: A benchmark for merging domain-specialized LLMs. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025. URL https://openreview.net/forum?id=rw50iUoyLu.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Lingxiao Huang, Zhize Li, Jialin Sun, and Haoyu Zhao. Coresets for vertical federated learning: Regularized linear regression and k-means clustering. *Advances in Neural Information Processing Systems*, 35:29566–29581, 2022.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*, 2022.

Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991. doi: 10.1162/neco.1991.3.1.79.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

Sneha Kudugunta, Yanping Huang, Ankur Bapna, Maxim Krikun, Dmitry Lepikhin, Minh-Thang Luong, and Orhan Firat. Beyond distillation: Task-level mixture-of-experts for efficient inference. *arXiv preprint arXiv:2110.03742*, 2021.

Viet Lai, Chien Nguyen, Nghia Ngo, Thuat Nguyen, Franck Dernoncourt, Ryan Rossi, and Thien Nguyen. Okapi: Instruction-tuned large language models in multiple languages with reinforcement learning from human feedback. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 318–327, 2023.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.

Gwen Legate, Nicolas Bernier, Lucas Page-Caccia, Edouard Oyallon, and Eugene Belilovsky. Guiding the last layer in federated learning with pre-trained models. *Advances in Neural Information Processing Systems*, 36:69832–69848, 2023.

Margaret Li, Suchin Gururangan, Tim Dettmers, Mike Lewis, Tim Althoff, Noah A. Smith, and Luke Zettlemoyer. Branch-train-merge: Embarrassingly parallel training of expert language models. In *First Workshop on Interpolation Regularizers and Beyond at NeurIPS 2022*, 2022.

Michael S Matena and Colin A Raffel. Merging models with fisher-weighted averaging. *Advances in Neural Information Processing Systems*, 35:17703–17716, 2022.

Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In Gordon H. Bower (ed.), *Psychology of Learning and Motivation*, volume 24, pp. 109–165. Academic Press, 1989. doi: https://doi.org/10.1016/S0079-7421(08)60536-8.

Jonas Pfeiffer, Naman Goyal, Xi Lin, Xian Li, James Cross, Sebastian Riedel, and Mikel Artetxe. Lifting the curse of multilinguality by pre-training modular transformers. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 3479–3495, 2022.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Machel Reid, Victor Zhong, Suchin Gururangan, and Luke Zettlemoyer. M2d2: A massively multi-domain language modeling dataset. *arXiv preprint arXiv:2210.07370*, 2022.

Ryan Rifkin, Gene Yeo, Tomaso Poggio, et al. Regularized least-squares classification. *Nato Science Series Sub Series III Computer and Systems Sciences*, 190:131–154, 2003.

John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *International Conference on Learning Representations (ICLR)*, 2017.

Sheng Shen, Le Hou, Yanqi Zhou, Nan Du, Shayne Longpre, Jason Wei, Hyung Won Chung, Barret Zoph, William Fedus, Xinyun Chen, et al. Mixture-of-experts meets instruction tuning: A winning combination for large language models. *arXiv preprint arXiv:2305.14705*, 2023.

Shivalika Singh, Freddie Vargus, Daniel D'souza, Börje F Karlsson, Abinaya Mahendiran, Wei-Yin Ko, Herumb Shandilya, Jay Patel, Deividas Mataciunas, Laura O'Mahony, et al. Aya dataset: An open-access collection for multilingual instruction tuning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11521–11567, 2024.

Stephen M Stigler. Gauss and the invention of least squares. *the Annals of Statistics*, pp. 465–474, 1981.

Sainbayar Sukhbaatar, Olga Golovneva, Vasu Sharma, Hu Xu, Xi Victoria Lin, Baptiste Rozière, Jacob Kahn, Daniel Li, Wen-tau Yih, Jason Weston, et al. Branch-train-mix: Mixing expert llms into a mixture-of-experts llm. *First Conference on Language Modeling (COLM)*, 2024.

Yuxuan Tong, Xiwen Zhang, Rui Wang, Ruidong Wu, and Junxian He. Dart-math: Difficulty-aware rejection tuning for mathematical problem-solving. *Advances in Neural Information Processing Systems*, 37:7821–7846, 2024.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Ke Wang, Nikolaos Dimitriadis, Guillermo Ortiz-Jimenez, François Fleuret, and Pascal Frossard. Localizing task information for improved model merging and compression. *International Conference on Machine Learning (ICML)*, 2024.

Ruohan Wang, Marco Ciccone, Giulia Luise, Massimiliano Pontil, Andrew Yapp, and Carlo Ciliberto. Schedule-robust online continual learning. *arXiv preprint arXiv:2210.05561*, 2022.

Yuxiang Wei, Zhe Wang, Jiawei Liu, Yifeng Ding, and Lingming Zhang. Magicoder: Empowering code generation with oss-instruct. *arXiv preprint arXiv:2312.02120*, 2023.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin A Raffel, and Mohit Bansal. Ties-merging: Resolving interference when merging models. *Advances in Neural Information Processing Systems*, 36:7093–7115, 2023.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *Advances in neural information processing systems*, 33: 5824–5836, 2020.

Chen Zhang, Xiongwei Hu, Yu Xie, Maoguo Gong, and Bin Yu. A privacy-preserving multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition. *Frontiers in Neurorobotics*, Volume 13 - 2019, 2020. ISSN 1662-5218. doi: 10.3389/fnbot.2019.00112.

Qizhen Zhang, Prajjwal Bhargava, Chloe Bi, Chris X Cai, Jakob Nicolaus Foerster, Jeremy Fu, Punit Singh Koura, Ruan Silva, Sheng Shen, Emily Dinan, et al. Bts: Harmonizing specialized experts into a generalist llm. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 6827–6845, 2025.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.

Tong Zhu, Daize Dong, Xiaoye Qu, Jiacheng Ruan, Wenliang Chen, and Yu Cheng. Dynamic data mixing maximizes instruction tuning for mixture-of-experts. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 1663–1677, 2025.

A On the hyperparameter choice

Figure 2 shows how the performance varies with the number of tokens used to extract the ridge regression statistics for each paragraph, k, and λ, in both the CLM and reasoning settings, for DUME, DUME+, and DUME OOD. In all experiments, we used the default hyperparameter values (except for the hyperparameter shown on the x-axis of these plots). The only exception is the DUME CLM curve in Figure 2b, for which we used 2 as the number of tokens per paragraph, as it already achieved a sufficiently good performance.

As shown in Figure 2a, increasing the number of tokens generally yields improved results across all settings and methods. In the CLM setting, two tokens are sufficient to achieve 90% of the dense experts' best performance, while 1024 tokens nearly reach the maximum score. These findings indicate that, within the CLM setting, computational costs can be further reduced since not all tokens in each paragraph are required. Figure 2b shows that DUME and DUME+ are relatively unaffected by the choice of k, while in the reasoning setting, it is better to only activate the top-1 expert every time. This means that the forward and training costs of the final MoE model can be minimized, since using top-1 selection yields the same forward costs as the dense experts, provided we consider the forward through the routers negligible. Finally, Figure 2c shows how $\lambda = 0.01$ provides the best results in the reasoning setting, and that its choice is not important in the CLM setting, thus DUME proves to be very robust to the choice of λ in the CLM experiments.

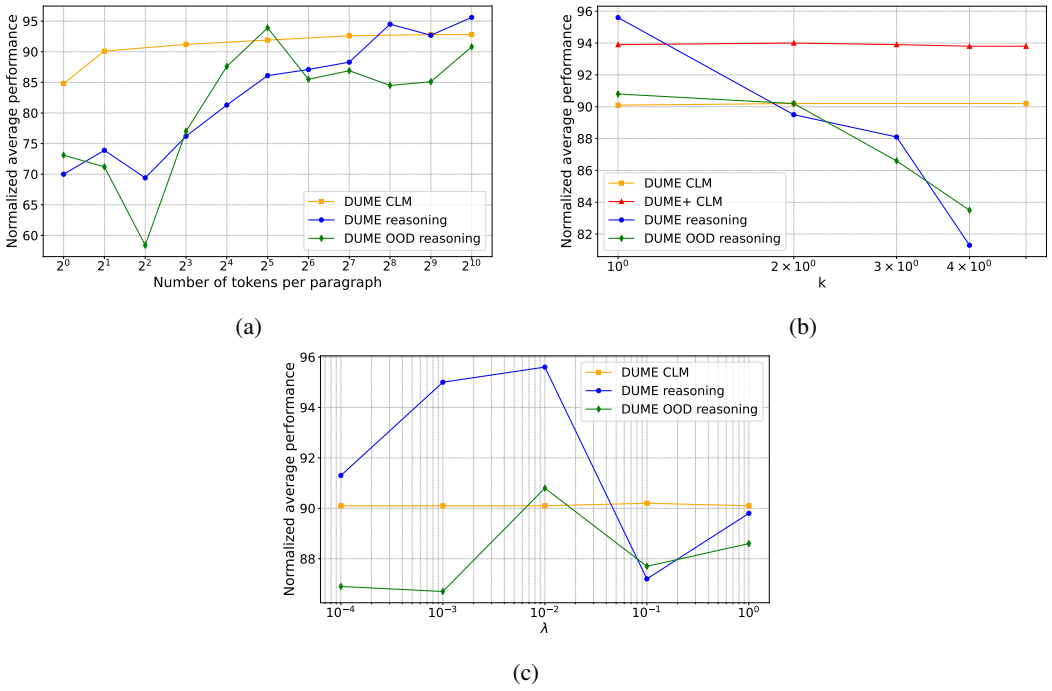

Figure 2: (a) Normalized average performance in both CLM and reasoning settings, with various numbers of tokens used to extract the ridge regression statistics for each paragraph. (b) Normalized average performance in both CLM and reasoning settings, as k of the top-k routing selecting increases. (c) Normalized average performance in both CLM and reasoning settings, with various values for the Tikhonov regularizer.