DIRICHLET-PRIOR SHAPING: GUIDING EXPERT SPECIALIZATION IN UPCYCLED MOES

Leyla Mirvakhabova[*]
Qualcomm AI Research[†]

Babak Ehteshami Bejnordi[*]
Qualcomm AI Research

Gaurav Kumar
Qualcomm Technologies

Hanxue Liang[‡]
University of Cambridge

Wanru Zhao
University of Cambridge

Paul Whatmough
Qualcomm AI Research

ABSTRACT

Upcycling pre-trained dense models into sparse Mixture-of-Experts (MoEs) efficiently increases model capacity during post-training, but often suffers from poor expert specialization due to naive weight replication. We introduce Dirichlet-Prior Shaping Loss (DPSL), a novel router regularization technique that directly shapes routing probability distributions by matching expert assignments to a target Dirichlet prior resulting in enhanced expert specialization. DPSL enables encoding of inductive biases such as encouraging experts to focus on specific modalities or tasks, without requiring manual intervention. DPSL is a general tool applicable to any module that outputs categorical probability distributions, extending its utility beyond MoE training. Experiments on upcycled MoE vision-language models show that DPSL consistently outperforms upcycling strategies and regularization techniques across standard vision-language benchmarks, addressing the critical issue of poor specialization and fostering higher-performing models.

Recent advances in large language models (LLMs) and multimodal LLMs (MLLMs) have transformed natural language and vision-language tasks. Model scaling drives this success (Kaplan et al., 2020; Hoffmann et al., 2022), enhancing accuracy and unlocking new capabilities, albeit with significant increases in training and inference costs. As pre-training becomes increasingly expensive, a growing fraction of model improvement is now driven by post-training stages such as instruction-tuning and multimodal fine-tuning, where compute efficiency and stability are critical. Sparse Mixture-of-Experts (MoE) architectures offer a promising solution by increasing model capacity while maintaining computational efficiency, activating only a subset of parameters ("experts") for each input token (Jacobs et al., 1991; Eigen et al., 2013).

Concurrently, sparse upcycling has emerged as an efficient training strategy by initializing an MoE from a pre-trained dense model, enabling rapid adaptation while leveraging existing representations (Komatsuzaki et al., 2023). This approach is particularly attractive in instruction-tuning and multimodal post-training, where budgets often preclude training large sparse models from scratch. The combination of MoE architectures and upcycling is especially well-suited for advancing MLLMs, enabling more capable multimodal systems without prohibitive computational overhead. Recent efforts such as LLaVA-MoE demonstrate the potential of this direction for improving MLLM efficiency during fine-tuning (Lin et al., 2024).

However, sparse upcycling introduces specific challenges in expert specialization during post-training. Naively initializing all MoE experts by replicating the dense model's feedforward network (FFN) weights (Komatsuzaki et al., 2023) leads to weight homogeneity, impeding the router's ability to differentiate experts and fully utilize model capacity, resulting in suboptimal performance (Nakamura et al., 2025). Drop-Upcycling (Nakamura et al., 2025) partially addresses this issue by re-initializing a subset of parameters within each expert to promote diversity, but its benefits typically emerge only after extensive training, often exceeding practical instruction-tuning budgets.

[*]Equal contribution
[†]Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc.
[‡]Work done during the internship at Qualcomm AI Research

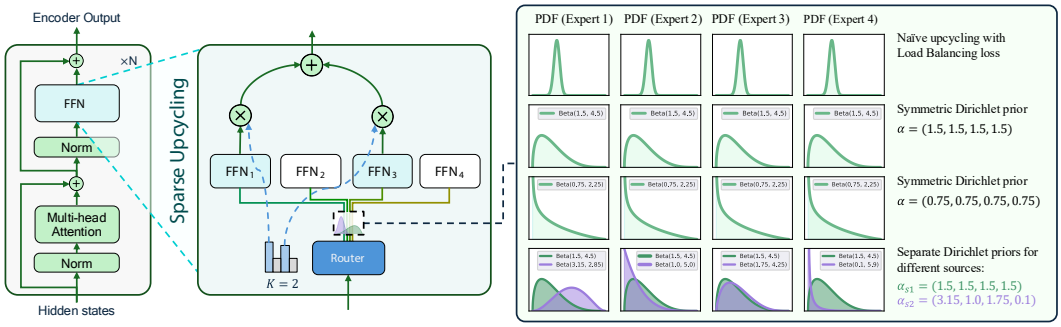


Figure 1: Sparse upcycling (left) initializes identical experts, yielding homogeneous routing probabilities and limited specialization. (right) Our proposed Dirichlet-Prior Shaping Loss guides routing towards desired distributions fostering balanced and confident selection (via symmetric priors) or targeted, modality-/task-aware specialization (via asymmetric priors).

Specialized upcycling methods such as Branch-Train-MiX (BTX) (Sukhbaatar et al., 2024) fine-tune separate model copies on different datasets to create diverse experts, which are then merged into an MoE and further fine-tuned with a learned router. However, BTX may yield experts specialized in suboptimal ways for MoE routing and can miss positive knowledge transfer, leading to inefficiencies and suboptimal convergence. In addition, standard MoE regularization, such as load-balancing loss (Shazeer et al., 2017; Fedus et al., 2022) and z-loss (Zoph et al., 2022) aim to stabilize training and prevent expert collapse, but do not directly induce specialization from identically initialized experts. Hence, they are unable to overcome the specialization challenges in upcycled MoEs.

To address the specialization challenges in upcycled MoEs, we first analyze routing behavior and find that, even with conventional regularization, upcycled MoEs exhibit low-confidence, weakly differentiated routing distributions. Expert assignment probabilities remain sharply peaked near $1/N$ (where N is the number of experts), indicating a persistent lack of specialization throughout training.

To overcome this, we propose Dirichlet-Prior Shaping Loss (DPSL), a principled router regularization technique that directly shapes the distribution of routing probabilities using Dirichlet priors (see Figure 1). DPSL generalizes the Batch Shaping Loss (Bejnordi et al., 2020) by matching the empirical distribution of expert assignments to a target Dirichlet prior enabling fine-grained control over both expert balance and specialization. Symmetric priors promote balanced expert utilization, while asymmetric priors allow targeted specialization. In this work, we focus on Vision-Language Models (VLMs), which present novel opportunities for expert specialization in MoEs. In VLMs, the coexistence of distinct modalities and data sources naturally exposes domain structure that MoE routers can harness, creating opportunities for experts to specialize along meaningful axes such as modality, dataset provenance, or task family. By doing so, our framework paves the way for more adaptive and efficient vision-language models. Our main contributions are:

- We show that naive upcycling restricts routing probability ranges, causing standard router regularization to fail in inducing effective expert specialization.
- We propose Dirichlet-Prior Shaping Loss (DPSL), a flexible tool to instill a wide array of desired statistical properties into the learning process of any module that outputs categorical probability distributions.
- We apply DPSL to upcycled MoEs to enable fine-grained control over expert behavior, using symmetric priors for robust balancing and asymmetric priors for targeted modality specialization.
- Extensive experiments on Qwen2 and Llama3.2 show that DPSL consistently outperforms state-of-the-art upcycling and regularization techniques across standard vision-language benchmarks.

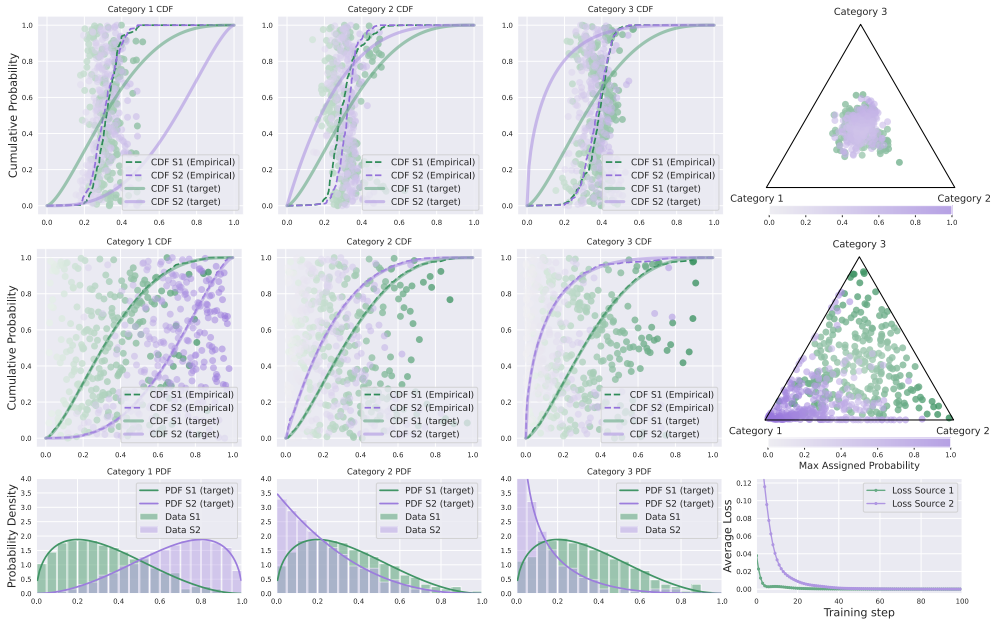


Figure 2: Dirichlet-Prior Shaping Loss (DPSL) shapes categorical probability distributions from two data sources (S1, S2). Top and middle rows show the empirical (dashed) vs. target (solid) CDFs for each category, at initialization and after convergence, respectively, along with simplex of assignment probabilities. Bottom row presents data histograms of assignment probabilities overlaid with target Beta PDFs, and learning curves showing DPSL minimization during training.

1 METHOD

1.1 DIRICHLET PRIORS FOR CATEGORICAL DISTRIBUTIONS

Let a model component output a probability vector $\mathbf{p} = [p_1, p_2, \ldots, p_K]$ over K distinct categories, where $\sum_{k=1}^{K} p_k = 1$ and $p_k \geq 0$. As the conjugate prior for categorical distributions, the Dirichlet distribution is the natural choice to model beliefs over such probability vectors. We model $\mathbf{p}$ as drawn from a Dirichlet distribution, $\mathbf{p} \sim \text{Dir}(\boldsymbol{\alpha})$, where $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_K]$ are positive concentration parameters that define the prior. The joint probability density function (PDF) of the Dirichlet distribution is:

$$f(\mathbf{p}; \boldsymbol{\alpha}) = \frac{1}{B(\boldsymbol{\alpha})} \prod_{k=1}^{K} p_k^{\alpha_k - 1}, \tag{1}$$

where the multivariate Beta function is $B(\boldsymbol{\alpha}) = \frac{\prod_{k=1}^{K} \Gamma(\alpha_k)}{\Gamma\left(\sum_{k=1}^{K} \alpha_k\right)}$ and $\Gamma(\cdot)$ is the Gamma function. A key property of the Dirichlet distribution is that each marginal p_k follows a Beta distribution (see Appendix A for derivation): $p_k \sim \text{Beta}(\alpha_k, A - \alpha_k)$, where $A = \sum_{k=1}^{K} \alpha_k$. The Beta distribution with parameters (α, β) has the following probability density and cumulative distribution functions:

$$f_{\text{Beta}}(x; \alpha, \beta) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha, \beta)}, \quad F_{\text{Beta}}(x; \alpha, \beta) = \int_0^x \frac{t^{\alpha-1}(1-t)^{\beta-1}}{B(\alpha, \beta)}\, dt \tag{2}$$

where $B(\alpha, \beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}$. The shape of each p_k's distribution depends on both α_k and the total A, reflecting dependencies among categories. For symmetric Dirichlet distribution, where all of the elements of the concentration parameter have the same value, larger α_k concentrates p_k near its mean; smaller values yield more dispersed or even U-shaped distributions (see Appendix A.3 for visualizations). The $\boldsymbol{\alpha}$ concentration parameter can flexibly control the expected distribution over categories: setting all $\alpha_k = 1$ yields a uniform prior, while asymmetric choices (e.g., $\boldsymbol{\alpha} = (\alpha_{\text{high}}, \alpha_{\text{low}}, \ldots)$) bias the distribution toward specific categories. This enables fine-grained control over categorical outputs, as detailed in the next section.

1.2 DIRICHLET-PRIOR SHAPING LOSS

To align the empirical distribution of categorical probabilities with a target Dirichlet prior, we adapt the Batch Shaping Loss from Bejnordi et al. (2020), based on the Cramér–von Mises criterion (Anderson, 1962). This criterion measures the squared difference between the empirical cumulative distribution function (CDF), $F_N(x)$, and the target theoretical CDF, $F(x)$:

$$\omega^2 = \int_{-\infty}^{\infty} [F_N(x) - F(x)]^2 dF(x). \tag{3}$$

For each of the K categories, we match the empirical distribution of assigned probabilities p_k (over a batch of samples) to the theoretical Beta distribution, Beta$(\alpha_k, A - \alpha_k)$, defined by Dirichlet prior.

Let $p_k^{(b)}$ denote the probability assigned to category k for the b-th sample in a batch of B total samples. The empirical CDF for the probabilities of category k, denoted as $F_N^{(k)}(x)$, is constructed from these probability values. The Dirichlet-Prior Shaping Loss (DPSL), $\mathcal{L}_{\text{DPS}}$, is then computed as the sum of squared differences between the empirical CDF and the target Beta CDF for each category:

$$\mathcal{L}_{\text{DPS}} = \lambda \sum_{k=1}^{K} \frac{1}{B} \sum_{j=1}^{B} \left[F_N^{(k)}\big(p_{k,(j)}\big) \; - F_{\text{Beta}}\big(p_{k,(j)}; \alpha_k, A - \alpha_k\big) \right]^2 \tag{4}$$

where $p_{k,(j)}$ denotes the j-th value in the sorted list of probabilities, $p_{k,(1)} \leq p_{k,(2)} \leq \cdots \leq p_{k,(B)}$, assigned to category k across the B samples in the batch. $F_{\text{Beta}}(p; \alpha_k, A - \alpha_k)$ is the theoretical CDF of the Beta distribution with parameters $(\alpha_k, A - \alpha_k)$, and $F_N^{(k)}(p_{k,(j)}) = j/B$ is the value of the empirical CDF at $p_{k,(j)}$. The hyperparameter λ controls the strength of this regularization.

Figure 2 illustrates DPSL in practice. For two data sources (S1 in green, S2 in purple) and three output categories, independent Dirichlet priors shape the output distributions. The first two rows show, for each category, empirical CDFs (dashed) and target Beta CDFs (solid) for both sources, at initialization (top row) and convergence (middle row), respectively; DPSL minimizes the distance between the empirical and target CDFs, thereby encouraging the model's output probabilities for each source to conform to the desired statistical profile. The rightmost bottom plot tracks DPSL convergence during training. The remaining bottom plots show, for each source, the empirical probability histograms per category, overlaid with the target Beta PDFs. For S1, a symmetric Dirichlet prior, $\boldsymbol{\alpha} = (1.5, 1.5, 1.5)$, yields balanced probabilities. In contrast, an asymmetric Dirichlet prior for S2, $\boldsymbol{\alpha} = (3, 1, 0.5)$, induces specialization predominately toward category one. We provide the training details, along with an additional example in Appendix B.1.

In essence, as demonstrated by the example in Figure 2, our Dirichlet-Prior Shaping method offers a powerful and flexible tool to instill a wide array of desired statistical properties and behaviors into the learning process of any module that outputs categorical probability distributions.

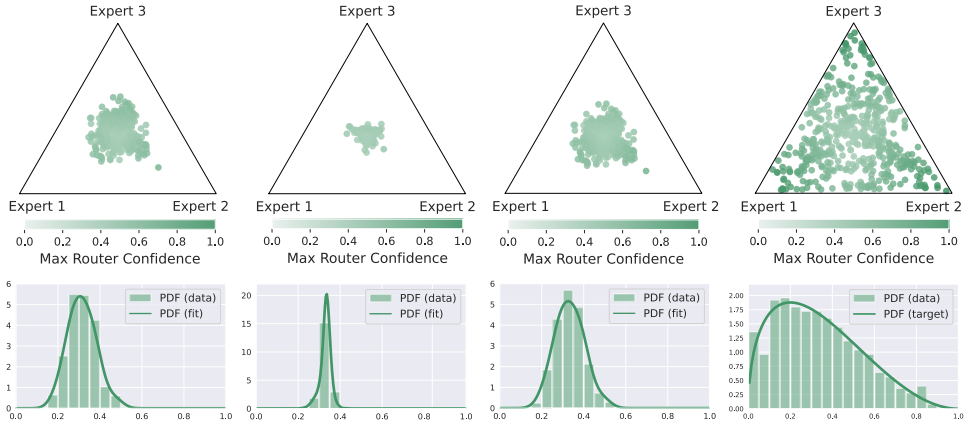
1.3 DPSL FOR UPCYCLED MOES

To motivate the need for a more nuanced control over router learning, especially in the context of upcycled MoE models, we first briefly review MoE fundamentals and then present an empirical study of router output distributions under various common regularization schemes.

1.3.1 MIXTURE-OF-EXPERTS BACKGROUND

MoEs increase model capacity and efficiency by activating only a subset of specialized subnetworks, or "experts", for each input token. Each MoE layer replaces a standard FFN with N expert FFNs $(E_1, E_2, \ldots, E_N)$ and a router module that assigns tokens to experts (see Figure 1 for an example with 4 experts and top-2 routing).

Given a token representation x, the router (with weights $\mathbf{W}_g$) computes logits $\mathbf{x}\mathbf{W}_g$, which are converted to routing probabilities $\mathbf{g}(\mathbf{x}) = \text{softmax}(\mathbf{x}\mathbf{W}_g)$. Sparse MoEs typically employ top-K gating: only the K experts with the highest routing probabilities $\mathbf{g}_i(\mathbf{x})$ are selected to process the



a) Without regularization b) z-loss c) load-balancing d) Dirichlet-Prior Shaping

Figure 3: Router output distributions for three experts in an upcycled MoE with top-1 routing. Each panel shows the simplex of routing probabilities under (a) no regularization, (b) z-loss, (c) load-balancing loss, and (d) Dirichlet-Prior Shaping Loss (symmetric prior).

token. Let $\mathcal{T}_k(\mathbf{x})$ be the set of indices corresponding to these top-K experts for token $\mathbf{x}$. The MoE output is:

$$\mathbf{y}(\mathbf{x}) = \sum_{i \in \mathcal{T}_k(\mathbf{x})} \mathbf{g}_i(\mathbf{x}) \cdot E_i(\mathbf{x}). \tag{5}$$

Recent MoE designs introduce "shared expert" (Dai et al., 2024) in addition to routed experts. This expert processes all input tokens, akin to a standard FFN. Throughout this paper, we employ MoE architectures with shared experts.

1.3.2 Analyzing Router Output Distributions in Upcycled MoEs

Upcycling a pre-trained dense model into an MoE creates challenges for the router: all experts start as identical FFNs, while the router must learn to differentiate token assignments to foster expert specialization. We analyzed router output distributions in an upcycled MoE with three experts and top-1 routing, comparing the effects of common regularization techniques. Figure 3 visualizes the router's output probability distribution for a specific MoE layer, plotted on a simplex where each point represents the probabilities (p_1, p_2, p_3) for a given input. Without regularization (a), router outputs cluster near the center, reflecting low confidence and limited expert differentiation. Applying z-loss (b) (Zoph et al., 2022) further compacts the distribution, stabilizing training but reducing the range and specialization of expert assignments. Load-balancing loss (c) (Fedus et al., 2022) distributes load more evenly but neither improves routing confidence nor encourages a wider probability dynamic range; notably, imbalanced load is often less critical in upcycled MoE training.

In contrast, our proposed Dirichlet-Prior Shaping Loss, illustrated in (d) with a symmetric prior, explicitly shapes the router's output distribution, allowing confident and diverse expert assignments while utilizing the full probability range. From a Bayesian perspective, this imposes a Dirichlet prior on expert selection, where α encodes our inductive bias. While, like any Bayesian approach, this introduces a choice of prior, we empirically find that a simple symmetric unit prior is effective across diverse settings, preventing the low-confidence collapse of conventional methods without requiring complex tuning.

2 Experiments and Results

This section evaluates our proposed Dirichlet-Prior Shaping Loss for training upcycled VLM MoEs. We base our study on the LLaVA framework (Liu et al., 2024b). For the language modeling backbone, we selected Qwen2-1.5B (Bai et al., 2023)) and Llama3.2-1B (Dubey et al., 2024). Following the setup outlined in LLaVA (Liu et al., 2024b) and MoE-LLaVA (Lin et al., 2024), we utilize CLIP Large (Radford et al., 2021) as the visual encoder.

2.1 Training Stages and Implementation Details

We upcycle pre-trained LLMs within the LLaVA framework into MoE architectures, while keeping the vision encoder intact. We investigate two primary MoE configurations: (1) a standard MoE,

where each expert is a full FFN replica, and (2) a granular MoE, where each expert is partitioned into multiple smaller ones, allowing more granular experts per token while maintaining constant active parameters (He et al., 2024; Dai et al., 2024; Ludziejewski et al., 2024). The standard configuration corresponds to a granularity of 1, resulting in a 4-expert setup with top-2 routing (2in4). In contrast, the granular MoE configuration uses a granularity of 4, yielding 16 experts (each ¼ the size of a full FFN) with top-8 routing (8in16). Despite the increased number of experts, the total and activated parameter count remains constant across configurations. We further discuss the details of the implementation of the upcycling of FFNs into granular experts in Appendix B.2.

Training stages. The training consists of three stages. Initially, we train the MLP projector to map visual tokens into the LLM's embedding space. The subsequent *warm-up stage* aims to bolster the model's general visual-language understanding using a large corpus, predominantly captioning datasets. This stage comprises two phases: first, the dense model with the aligned projector is fine-tuned; second, we introduce the MoE experts and fine-tune the complete MoE architecture, including the experts, router, and other existing parameters. The final *finetuning stage*, involves training on diverse task-specific datasets. This stage aims to refine the experts' capabilities, enabling them to learn the nuances and intricacies of specific tasks. The detailed breakdown of the datasets can be found in Appendix B.3 along with implementation details in Appendix B.4. We maintain the same training pipeline for all baselines and our method.

DPSL for Upcycled MoE and Computational Overhead. DPSL is computed at the token level across the entire batch, resulting in an effective sample size of $B = S \times T$, where S denotes the number of sequences and T represents the average sequence length. We apply DPSL and other router regularization baselines only in the second phase of *warm-up stage*. The computational overhead of DPSL is minimal. When using DeepSpeed with ZeRO offloading, we observe 1–5% overhead during warm-up (Appendix B.5). Since DPSL is active only during warm-up, the total increase in end-to-end training time is only 0.3–1.6%. Importantly, the cost is incurred primarily in the forward pass; because the derivative of the CDF (the PDF) is already computed during the forward pass, there is no additional backward pass cost.

2.2 Baselines and Downstream Evaluations

We categorize our baselines into two groups. The first comprises upcycling methods without explicit regularizers: Sparse Upcycling (Komatsuzaki et al., 2023), which directly copies dense model weights to intialize experts, and Drop-Upcycling (Nakamura et al., 2025), which introduces partial weight re-initialization with random noise. The second group includes methods with additional router regularizations: load-balancing loss (Shazeer et al., 2017; Fedus et al., 2022); z-loss (Zoph et al., 2022), and the loss-free DeepSeek balancing procedure (Wang et al., 2024; Liu et al., 2024a). We describe the hyperparameters of these methods in Appendix B.4. For a fair and rigorous comparison, we fine-tuned these baselines for their strongest possible performance, as detailed in Appendix B.4. Finally, we subjected our dense baselines to the exact same enhanced training protocol as our MoE models which resulted in significantly stronger reference accuracies beyond standard practices used in LLaVA (Liu et al., 2024b) and MoE-LLaVA (Lin et al., 2024).

We evaluate our method across six benchmarks. For VQA-style tasks, models are tested on GQA (Hudson & Manning, 2019), TextVQA (Singh et al., 2019), and VizWiz (Gurari et al., 2018). Instruction-following capabilities are assessed using MME (Fu et al., 2023) (consisting of MME-Perception and MME-Cognition), MM-Vet (Yu et al., 2024) and MMBench (Liu et al., 2025). Due to the constraint on the number of submissions for VizWiz evaluation and our large number of baselines and models, we have evaluated all models on the Test-Dev2024 split.

2.3 Downstream Task Evaluation Results

Table 1 summarizes the downstream evaluation results across all evaluated models and upcycling methods. Standard sparse upcycling without regularization shows minimal performance gains, and in some cases, performs worse than the dense baseline, underscoring the challenge of effective expert specialization in naive upcycling. Our Dirichlet-Prior Shaping approach consistently achieves the highest average performance across all models and MoE configurations, including both the standard 2in4 and granular 8in16 expert settings, while the second-best method is a moving target. This consistency demonstrates the effectiveness of our method, regardless of backbone or architecture making DPSL a more reliable and generalizable choice. Among the baselines, DeepSeek balancing

Table 1: Downstream task performance comparison of upcycled VLM MoEs methods across various backbone LLMs and MoE configurations. Highest accuracy is marked in bold, 2nd best is underlined. We also report the average accuracy (unweighted mean across seven metrics) for MME-P and MME-C, the scores were normalized over the maximal possible scores (2000 and 800, respectively).

		Setup	TextVQA	GQA	MM-Vet	MME-P	MME-C	VizWiz	MMB	Avg
Qwen2-1.5B		Dense	**54.28**	61.43	34.3	1442.67	266.07	38.80	**66.31**	36.60
	2in4	Sparse Upcycling	53.14	61.65	32.9	1418.53	296.07	39.38	65.07	36.17
		Drop-Upcycling	53.23	**62.10**	34.9	1389.21	287.86	46.01	65.70	37.57
		Load-balancing	53.66	61.42	33.0	1412.83	**298.57**	41.22	64.96	36.48
		Z-loss	53.80	61.81	**36.3**	1417.29	265.00	39.04	65.86	36.84
		DeepSeek balancing	53.30	61.68	33.2	1420.25	293.92	41.87	65.92	36.72
		DPSL	53.01	62.01	35.3	**1459.06**	289.71	**49.55**	66.26	**38.17**
	8in16	Sparse Upcycling	53.74	62.01	33.8	1393.42	270.00	40.74	65.75	36.72
		Drop-Upcycling	54.16	61.80	34.1	1435.91	280.35	41.30	**66.36**	36.97
		Load-balancing	53.93	**62.10**	**34.6**	1418.28	266.78	38.16	65.80	36.52
		Z-loss	53.95	61.36	29.2	1394.94	266.79	39.48	65.86	35.84
		DeepSeek balancing	**54.49**	61.97	32.3	**1444.88**	**310.71**	**43.70**	65.74	37.04
		DPSL	53.32	61.86	34.0	1421.90	265.00	43.54	65.98	**37.25**
Llama3.2-1B		Dense	51.19	60.18	30.5	1295.99	**253.93**	35.81	61.71	34.19
	2in4	Sparse Upcycling	51.20	**61.00**	30.0	1309.71	251.43	40.81	60.31	34.85
		Drop-Upcycling	50.50	60.43	29.8	1293.02	236.78	**43.00**	62.61	35.33
		Load-balancing	49.49	59.79	31.3	1331.86	247.14	40.54	59.30	34.48
		Z-loss (2in4)	51.00	60.67	30.7	1318.65	246.07	39.62	61.49	34.92
		DeepSeek balancing	51.50	60.64	29.5	1265.25	220.00	36.65	62.39	34.51
		DPSL	**52.82**	60.98	**31.7**	**1334.78**	253.21	42.19	**62.78**	**35.92**
	8in16	Sparse Upcycling	51.47	60.78	28.5	1285.61	223.57	38.66	61.88	34.92
		Drop-Upcycling	51.75	60.70	32.0	**1352.30**	**267.40**	39.50	**63.30**	35.47
		Load-balancing	49.80	59.97	28.1	1312.68	227.50	**45.53**	61.32	35.09
		Z-loss (8in16)	52.06	60.92	**33.5**	1340.57	246.43	38.74	62.84	35.5
		DeepSeek balancing	**52.89**	**61.32**	32.2	1321.10	228.21	38.74	63.17	35.61
		DPSL	52.10	61.09	29.7	1294.50	247.86	44.03	63.23	**35.87**

and Drop-Upcycling are generally strong performers, but their effectiveness varies by model and architecture. For instance, DeepSeek balancing achieves high scores with Qwen 8in16, but underperforms on Llama 2in4 and Qwen 2in4. Overall, these results establish DPSL as the most consistent and broadly effective upcycling regularization strategy.

2.4 Modality-Specialized Upcycling

We compare a manual modality-specific routing baseline, where experts are hard-assigned to vision or language tokens, against DPSL with modality-aware priors. For this experiment, we use Llama3.2-1B model with 4 experts and top-2 routing. In DPSL, we encourage soft specialization by adding a scalar bias 0.5 to the base concentration of the target experts. As shown in Table 2, manual specialization performs poorly due to rigidity, whereas modality-specific DPSL achieves the best results, even slightly outperforming our symmetric DPSL, highlighting the benefit of flexibly integrated, informed priors for MLLMs, whereas suboptimal manual approaches might prematurely dismiss such strategies.

2.5 Ablation Study

Concentration parameter. We ablate the symmetric Dirichlet concentration α to assess sensitivity to prior sharpness. Smaller α encourages sparse, corner-biased routing, while larger α favors uniform, center-biased assignments ($\alpha = 1$ is the flat prior). Detailed results are in Appendix C.1 (Table 6 and Table 7). The optimal α can be reliably selected using just a small subset of training data (e.g. one-quarter of the warm-up phase). As shown in Appendix C.1, the α yielding the lowest training loss on this subset corresponds to the best performer on the full training set. In general, a symmetric

Table 2: Performance comparison of modality- and task-specific expert specialization strategies on Llama3.2-1B (2in4) performance.

Model	TextVQA	GQA	MM-Vet	MME-P	MME-C	VizWiz	MMB	Avg
Dense	51.19	60.18	30.5	1295.99	253.93	35.81	61.71	34.19
DPSL (symmetric-prior)	**52.82**	60.98	31.7	**1334.78**	253.21	42.19	62.78	35.92
Manual (modality)	51.60	60.82	30.3	1323.09	242.14	37.37	61.49	34.65
DPSL (modality-prior)	51.83	**61.40**	**32.1**	1304.96	**285.00**	42.88	**64.01**	**36.18**

unit prior ($\alpha = 1$) serves as a reliable starting point, performing well for Qwen. We also observe that this larger backbone remains robust across a wide range of α. Llama3.2-1B benefits from a modestly lower concentration ($\alpha = 0.75$). We provide practical guidelines for selecting α in Appendix C.2.

Routing Distributions & Expert Specialization Patterns. We visualize router score distributions for Llama3.2-1B (2in4) in Appendix C.3. While conventional baselines (z-loss, load-balancing, DeepSeek) peak sharply at the uniform probability (i.e., $1/4$ for four experts), reflecting low routing confidence, DPSL yields a significantly broader and more differentiated distribution. We further examine the expert specialization patterns by computing pairwise cosine similarity between expert activations across layers, see Appendix C.4 for a detailed analysis. Our DPSL maintains the lowest average similarity compared to load-balancing loss and z-loss, demonstrating enhanced expert specialization. We additionally analyze the expert utilization by measuring the Coefficient of Variation (CoV) of expert loads at different layers. We present the results in the Appendix C.5 (Table 9). We can observe that even without explicit enforcing of expert balancing, DPSL loss with a symmetric prior intrinsically encourages a balanced load distribution consistently visible across layers.

3 Related Work

Our work builds on sparse upcycling, which enhances model capacity by initializing MoE experts from dense FFN weights (Komatsuzaki et al., 2023; Lin et al., 2024). Naive sparse upcycling typically involves replicating FFN weights, which can lead to initial expert homogeneity and low specialization. To address this, Drop-Upcycling (Nakamura et al., 2025) uses partial re-initialization to promote diversity. Similarly, He et al. (2024) propose "virtual group" initialization for fine-grained MoEs. However, while these methods focus on initialization, our DPSL method offers continuous, fine-grained control over expert specialization throughout training by shaping router output distributions.

Effective MoE training also relies on managing router behavior and expert utilization. Common strategies include load-balancing losses for uniform expert activation (Shazeer et al., 2017) and router z-loss for training stability (Zoph et al., 2022). Entropy-based regularization (Chen et al., 2025) pushes per-token distributions toward high entropy, thereby discouraging confident assignments and typically shrinking all router outputs toward the center of the simplex. Recent auxiliary-loss-free methods, like DeepSeek-V3 Wang et al. (2024); Liu et al. (2024a), employ dynamic bias adjustment. Unlike these methods that target even load distribution or numerical stability, our DPSL directly models and regularizes the entire categorical distribution of routing probabilities. DPSL exposes explicit, interpretable control knobs via the Dirichlet concentration parameters.

4 Conclusion

In this paper, we introduce Dirichlet-Prior Shaping Loss (DPSL), a novel and principled regularization technique that aligns categorical output distributions with a target Dirichlet prior. Applied to upcycled VLM MoEs, DPSL consistently outperforms baselines across three backbones and two MoE granularities. Our analysis confirms that DPSL induces broader routing distributions and stronger expert differentiation while maintaining balanced utilization. Furthermore, we demonstrate the promise of modality-specific priors for multimodal learning, enabling more adaptive and effective expert allocation in MLLMs. Future work will explore extending DPSL to training MoEs from scratch, leveraging its ability to instill desired statistical behaviors directly into the learning process.

5 ETHICS AND REPRODUCIBILITY STATEMENTS

We adhere to the ICLR Code of Ethics. This paper focuses on training methodology to enhance MoE upcycling, however, the model itself incorporates an LLM that may perpetuate biases present in the training data, potentially affecting fairness and reliability. Therefore, we recommend adhering to standard ethical guidelines for the use of LLMs to mitigate these risks.

During the preparation of this manuscript, we utilized large language models (LLMs) to assist with grammar correction and refinement of the writing.

In this paper, we provide all the necessary details to ensure the reproducibility of the presented method. We provide the theoretical justification of the method in Section 1 and Appendix A, implementation details and training protocoles in Section 2.1, Appendix B.2 and Appendix B.4, and data description in Appendix B.3.

REFERENCES

Theodore W Anderson. On the Distribution of the Two-Sample Cramer-von Mises Criterion. *The Annals of Mathematical Statistics*, pp. 1148–1159, 1962.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. Qwen Technical Report. *arXiv preprint arXiv:2309.16609*, 2023.

Babak Ehteshami Bejnordi, Tijmen Blankevoort, and Max Welling. Batch-Shaping for Learning Conditional Channel Gated Networks. In *International Conference on Learning Representations*, 2020.

Guiming Hardy Chen, Shunian Chen, Ruifei Zhang, Junying Chen, Xiangbo Wu, Zhiyi Zhang, Zhihong Chen, Jianquan Li, Xiang Wan, and Benyou Wang. ALLaVA: Harnessing GPT4V-Synthesized Data for a Lite Vision-Language Model. *arXiv preprint arXiv:2402.11684*, 2024.

Xuanze Chen, Jiajun Zhou, Jinsong Chen, Shanqing Yu, and Qi Xuan. Mixture of decoupled message passing experts with entropy constraint for general node classification. *arXiv preprint arXiv:2502.08083*, 2025.

Damai Dai, Chengqi Deng, Chenggang Zhao, R.x. Xu, Huazuo Gao, Deli Chen, Jiashi Li, Wangding Zeng, Xingkai Yu, Y. Wu, Zhenda Xie, Y.k. Li, Panpan Huang, Fuli Luo, Chong Ruan, Zhifang Sui, and Wenfeng Liang. DeepSeekMoE: Towards Ultimate Expert Specialization in Mixture-of-Experts Language Models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1280–1297. Association for Computational Linguistics, 2024.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The Llama 3 Herd of Models. *arXiv preprint arXiv:2407.21783*, 2024.

David Eigen, Marc'Aurelio Ranzato, and Ilya Sutskever. Learning Factored Representations in a Deep Mixture of Experts. *arXiv preprint arXiv:1312.4314*, 2013.

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.

Chaoyou Fu, Peixian Chen, Yunhang Shen, Yulei Qin, Mengdan Zhang, Xu Lin, Jinrui Yang, Xiawu Zheng, Ke Li, Xing Sun, et al. MME: A Comprehensive Evaluation Benchmark for Multimodal Large Language Models. *arXiv preprint arXiv:2306.13394*, 2023.

Danna Gurari, Qing Li, Abigale J Stangl, Anhong Guo, Chi Lin, Kristen Grauman, Jiebo Luo, and Jeffrey P Bigham. VizWiz Grand Challenge: Answering Visual Questions from Blind People. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3608–3617, 2018.

Ethan He, Abhinav Khattar, Ryan Prenger, Vijay Korthikanti, Zijie Yan, Tong Liu, Shiqing Fan, Ashwath Aithal, Mohammad Shoeybi, and Bryan Catanzaro. Upcycling large language models into mixture of experts. *arXiv preprint arXiv:2410.07524*, 2024.

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. An Empirical Analysis of Compute-Optimal Large Language Model Training. *Advances in Neural Information Processing Systems*, 35:30016–30030, 2022.

Drew A Hudson and Christopher D Manning. GQA: A New Dataset for Real-World Visual Reasoning and Compositional Question Answering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6700–6709, 2019.

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.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling Laws for Neural Language Models. *arXiv preprint arXiv:2001.08361*, 2020.

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment Anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 4015–4026, 2023.

Aran Komatsuzaki, Joan Puigcerver, James Lee-Thorp, Carlos Riquelme Ruiz, Basil Mustafa, Joshua Ainslie, Yi Tay, Mostafa Dehghani, and Neil Houlsby. Sparse Upcycling: Training Mixture-of-Experts from Dense Checkpoints. In *The Eleventh International Conference on Learning Representations*, 2023.

Bin Lin, Zhenyu Tang, Yang Ye, Jiaxi Cui, Bin Zhu, Peng Jin, Junwu Zhang, Munan Ning, and Li Yuan. MoE-LLaVA: Mixture of Experts for Large Vision-Language Models. *arXiv preprint arXiv:2401.15947*, 2024.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 Technical Report. *arXiv preprint arXiv:2412.19437*, 2024a.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved Baselines with Visual Instruction Tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 26296–26306, 2024b.

Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, et al. MMBench: Is your Multi-Modal Model an All-Around Player? In *European Conference on Computer Vision*, pp. 216–233. Springer, 2025.

Jan Ludziejewski, Jakub Krajewski, Kamil Adamczewski, Maciej Pióro, Michał Krutul, Szymon Antoniak, Kamil Ciebiera, Krystian Król, Tomasz Odrzygóźdź, Piotr Sankowski, Marek Cygan, and Sebastian Jaszczur. Scaling Laws for Fine-Grained Mixture of Experts. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 33270–33288. PMLR, 2024.

Taishi Nakamura, Takuya Akiba, Kazuki Fujii, Yusuke Oda, Rio Yokota, and Jun Suzuki. Drop-Upcycling: Training Sparse Mixture of Experts with Partial Re-Initialization. In *The Thirteenth International Conference on Learning Representations*, 2025.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning Transferable Visual Models from Natural Language Supervision. In *International Conference on Machine Learning*, pp. 8748–8763. PMLR, 2021.

Babak Saleh and Ahmed Elgammal. Large-Scale Classification of Fine-Art Paintings: Learning the Right Metric on the Right Feature. *arXiv preprint arXiv:1505.00855*, 2015.

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*, 2017.

Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. Towards VQA Models That Can Read. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8317–8326, 2019.

Sainbayar Sukhbaatar, Olga Golovneva, Vasu Sharma, Hu Xu, Xi Victoria Lin, Baptiste Roziere, Jacob Kahn, Shang-Wen Li, Wen tau Yih, Jason E Weston, and Xian Li. Branch-Train-MiX: Mixing Expert LLMs into a Mixture-of-Experts LLM. In *First Conference on Language Modeling*, 2024.

Junke Wang, Lingchen Meng, Zejia Weng, Bo He, Zuxuan Wu, and Yu-Gang Jiang. To See is to Believe: Prompting GPT-4V for Better Visual Instruction Tuning. *arXiv preprint arXiv:2311.07574*, 2023.

Lean Wang, Huazuo Gao, Chenggang Zhao, Xu Sun, and Damai Dai. Auxiliary-Loss-Free Load Balancing Strategy for Mixture-of-Experts. *arXiv preprint arXiv:2408.15664*, 2024.

Weihao Yu, Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Zicheng Liu, Xinchao Wang, and Lijuan Wang. MM-Vet: Evaluating Large Multimodal Models for Integrated Capabilities. *International Conference on Machine Learning*, 2024.

Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. ST-MoE: Designing Stable and Transferable Sparse Expert Models. *arXiv preprint arXiv:2202.08906*, 2022.

APPENDIX

A MARGINALS OF THE DIRICHLET DISTRIBUTION

In this appendix, we provide proof that the marginal distribution of each component p_k of a Dirichlet random vector $\mathbf{p} = (p_1, \ldots, p_K) \sim \mathrm{Dir}(\boldsymbol{\alpha})$ follows a Beta distribution. We first establish the aggregation property of the Dirichlet distribution, then use it to derive the marginal.

A.1 AGGREGATION PROPERTY OF THE DIRICHLET DISTRIBUTION

Statement: If $\mathbf{p} = (p_1, \ldots, p_i, \ldots, p_j, \ldots, p_K) \sim \mathrm{Dir}(\alpha_1, \ldots, \alpha_i, \ldots, \alpha_j, \ldots, \alpha_K)$, then the vector $\mathbf{p}'$ obtained by aggregating components p_i and p_j into a single component $p'_i = p_i + p_j$, i.e., $\mathbf{p}' = (p_1, \ldots, p_i + p_j, \ldots, p_{j-1}, p_{j+1}, \ldots, p_K)$, follows a Dirichlet distribution with parameters $(\alpha_1, \ldots, \alpha_i + \alpha_j, \ldots, \alpha_{j-1}, \alpha_{j+1}, \ldots, \alpha_K)$.

Proof: Without loss of generality, we aggregate the first two components, p_1 and p_2. We want to find the marginal distribution of $\mathbf{p}' = (y, p_3, \ldots, p_K)$, where $y = p_1 + p_2$, by integrating the joint PDF of $(p_1, p_2, p_3, \ldots, p_K)$ over the region defined by $p_1 + p_2 = y$, keeping $p_3, \ldots, p_K$ fixed. We integrate with respect to p_1, while substituting $p_2 = y - p_1$. Based on Equation (1) in Section 1.1, the PDF for $(y, p_3, \ldots, p_K)$ is:

$$f(y, p_3, \ldots, p_K) = \int_0^y \frac{1}{B(\boldsymbol{\alpha})} p_1^{\alpha_1 - 1} (y - p_1)^{\alpha_2 - 1} \left(\prod_{k=3}^{K} p_k^{\alpha_k - 1} \right) dp_1 \tag{6}$$

$$= \frac{1}{B(\boldsymbol{\alpha})} \left(\prod_{k=3}^{K} p_k^{\alpha_k - 1} \right) \int_0^y p_1^{\alpha_1 - 1} (y - p_1)^{\alpha_2 - 1} dp_1 \tag{7}$$

Applying a change of variables $p_1 = yt$, and evaluating the integral:

$$\int_0^y p_1^{\alpha_1 - 1} (y - p_1)^{\alpha_2 - 1} dp_1 = \int_0^1 (yt)^{\alpha_1 - 1} (y - yt)^{\alpha_2 - 1} (y dt) \tag{8}$$

$$= y^{\alpha_1 + \alpha_2 - 1} \int_0^1 t^{\alpha_1 - 1} (1 - t)^{\alpha_2 - 1} dt \tag{9}$$

The remaining integral is the definition of the Beta function $B(\alpha_1, \alpha_2)$. Substituting this back in Equation 7:

$$f(y, p_3, \ldots, p_K) = \frac{B(\alpha_1, \alpha_2)}{B(\boldsymbol{\alpha})} y^{\alpha_1 + \alpha_2 - 1} \prod_{k=3}^{K} p_k^{\alpha_k - 1} \tag{10}$$

The constant term $\frac{B(\alpha_1, \alpha_2)}{B(\boldsymbol{\alpha})}$ is:

$$\frac{B(\alpha_1, \alpha_2)}{B(\boldsymbol{\alpha})} = \frac{\frac{\Gamma(\alpha_1)\Gamma(\alpha_2)}{\Gamma(\alpha_1 + \alpha_2)}}{\frac{\Gamma(\alpha_1)\Gamma(\alpha_2)\Gamma(\alpha_3)\cdots\Gamma(\alpha_K)}{\Gamma(\alpha_1 + \alpha_2 + \alpha_3 + \cdots + \alpha_K)}} = \frac{\Gamma(\alpha_1 + \alpha_2 + \alpha_3 + \cdots + \alpha_K)}{\Gamma(\alpha_1 + \alpha_2)\Gamma(\alpha_3) \cdots \Gamma(\alpha_K)} \tag{11}$$

This is the reciprocal of the multivariate Beta function for parameters $(\alpha_1 + \alpha_2, \alpha_3, \ldots, \alpha_K)$. Let $\boldsymbol{\alpha}' = (\alpha_1 + \alpha_2, \alpha_3, \ldots, \alpha_K)$. Then the constant is $\frac{1}{B(\boldsymbol{\alpha}')}$. So, the PDF becomes:

$$f(y, p_3, \ldots, p_K) = \frac{1}{B(\boldsymbol{\alpha}')} y^{(\alpha_1 + \alpha_2) - 1} \prod_{k=3}^{K} p_k^{\alpha_k - 1} \tag{12}$$

Therefore, the marginal distribution of $\mathbf{p}'$ is exactly a Dirichlet distribution $\mathrm{Dir}(\alpha_1 + \alpha_2, \alpha_3, \ldots, \alpha_K)$. This proves the aggregation property for summing two components. The argument can be extended by induction to summing any number of components.

A.2 Marginals of the Dirichlet Distribution are Beta Distributions

Using the aggregation property proven above, we can derive the marginal distribution of a single component p_i. Aggregate all components except p_i into a single component:

$$p_{-i} = 1 - p_i = \sum_{k \neq i} p_k. \tag{13}$$

By the aggregation property, we have:

$$(p_i, p_{-i}) \sim \text{Dir}(\alpha_i, A - \alpha_i), \tag{14}$$

where $A = \sum_{k=1}^{K} \alpha_k$. Since, the 2-dimensional Dirichlet distribution is equivalent to a Beta distribution, it follows that:

$$p_i \sim \text{Beta}(\alpha_i, A - \alpha_i). \tag{15}$$

This proves that the marginals of a Dirichlet distribution are Beta distributed, as stated in Section 1.1.

A.3 Visualization of the Marginal Beta Distributions

Figure 4 visualizes the marginal Beta distributions for each component of a Dirichlet distribution. For a symmetric Dirichlet distribution, where all of the elements of the concentration parameter have the same value, larger α_k concentrates p_k near its mean (e.g. $\text{Dir}(5.0, 5.0, 5.0)$); smaller values yield more dispersed or even U-shaped distributions (e.g. $\text{Dir}(0.2, 0.2, 0.2)$); while an $\alpha = 1$ known as the flat Dirichlet distribution corresponds to a uniform distribution over the simplex ($\text{Dir}(1.0, 1.0, 1.0)$). Finally, we present the marginal beta distributions when an asymmetric concentration parameter is used ($\text{Dir}(0.75, 0.1, 1.25)$) in which the last component has the biggest value placing more mass at this component.

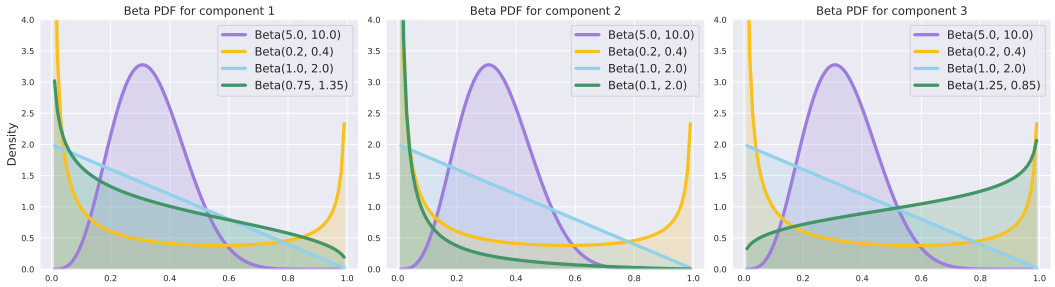


Figure 4: Visualization of the marginal Beta distributions for the following Dirichlet distributions: —$\text{Dir}(5.0, 5.0, 5.0)$, —$\text{Dir}(0.2, 0.2, 0.2)$, —$\text{Dir}(1.0, 1.0, 1.0)$, and —$\text{Dir}(0.75, 0.1, 1.25)$.

B Training and Implementation Details

B.1 Training details for the experiment in Section 2.2

This appendix provides the training details with an additional illustrative example for applying the Dirichlet-Prior Shaping Loss (DPSL), as referenced in Section 1.2. The objective is to guide a set of learnable probability distributions over three categories to match target Dirichlet priors.

In the example shown in Figure 2 in the paper and Figure 5 in this section, we consider data points representing probability distributions derived from two distinct sources. Independent Dirichlet priors are applied to shape the distributions for each source: For example in Figure 5, Source one has a target prior of $\text{Dir}(5, 5, 5)$ and Source two has a target prior of $\text{Dir}(0.2, 0.2, 0.2)$.

For training, we initialize the data points as learnable parameters. These parameters are optimized using the Adam optimizer with a learning rate of 0.1 for 100 training steps. The optimization minimizes the Dirichlet-Prior Shaping Loss (defined in Equation 4), which quantifies the difference

between the empirical CDF of the learned probabilities (for each category) and the theoretical Beta CDF derived from the respective target Dirichlet prior. The learning curve, shown in the bottom right panel of Figure 5, tracks the minimization of this loss during training.

As illustrated in Figure 5, minimizing the CDF divergence ensures that the empirical distribution of the learnable probability vectors for each source converges effectively to its specified target Dirichlet prior (top row). The choice of concentration parameters (α_k) significantly influences the characteristics of the learned distributions. For source one, the larger $\alpha_k = 5$ values steer the probability distributions towards the mean of the simplex. For source two, the smaller $\alpha_k = 0.2$ values promote sparse probability distributions. This results in distributions heavily concentrated at the corners of the simplex, where one category is assigned a high probability, and the others are assigned probabilities near zero, indicating a strong preference for a single category.

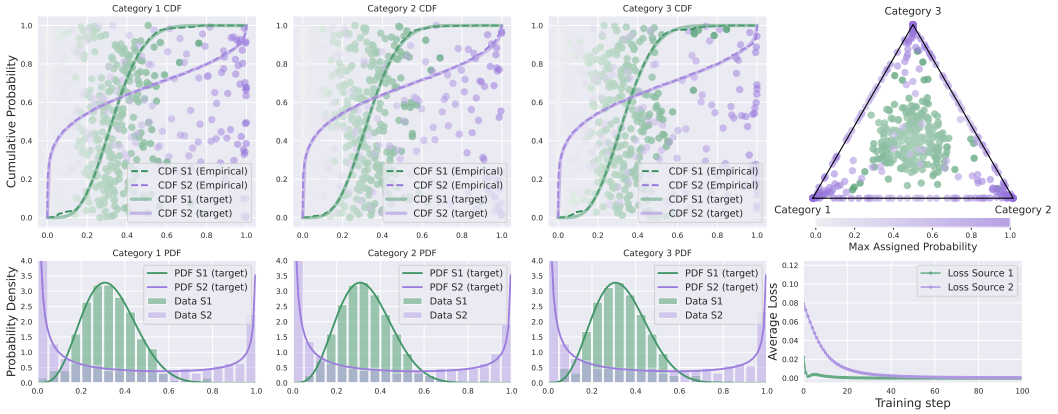


Figure 5: Dirichlet-Prior Shaping Loss (DPSL) shapes categorical probability distributions from two data sources (S1, S2). Top row shows the empirical (dashed) vs. target (solid) CDFs for each category after convergence, along with simplex of assignment probabilities. Bottom row presents data histograms of assignment probabilities overlaid with target Beta PDFs, and learning curves showing DPSL minimization during training.

B.2 Implementation Details for Upcycling FFNs into Granular Experts

This section provides implementation details for upcycling Feed-Forward Networks (FFNs) into granular experts, as referenced in Section 2.1. Granularity, in this context, refers to the ratio of the original FFN's hidden dimension ($d_{\texttt{ffn}}$) to the hidden dimension of an MoE expert ($d_{\texttt{exp}}$), expressed as $G = d_{\texttt{ffn}}/d_{\texttt{exp}}$. Creating smaller, more granular experts allows tokens to be routed to a larger number of experts, which has shown promising accuracy results (Ludziejewski et al., 2024; He et al., 2024) for granular expert upcycling. We closely follow the approach detailed in He et al. (2024).

We experimented with both standard upcycling and fine-grained upcycling, as follows: For standard upcycling, we duplicate the original FFN blocks to create experts. We add noise to the weights at initialization with a small magnitude, $\epsilon \sim \mathcal{N}(0, 0.01)$. For fine-grained upcycling, we follow the approach proposed by He et al. (2024), partitioning each FFN weight tensor into G shards. In our experiments, upcycling with granularity 1 (standard upcycling) corresponds to a setup with 4 experts and top-2 routing. Fine-grained upcycling corresponds to a setup with 16 experts and top-8 routing.

Notably, we implemented weight scaling for expert initialization (He et al., 2024), but found that it resulted in decreased accuracy in our experiments. Therefore, we did not use it in the final experimental setup.

B.3 Datasets

Table 3 provides a detailed breakdown of the datasets used for training in every stage (stage I: *projector-training*, stage II: *warm-up*, stage III: *finetuning*). We maintain the same training pipeline and stages for all baselines and our Dirichlet-Prior Shaped models.

Table 3: Datasets used in training stages. On the first stage, we are training the adapter network. On the second stage, we train the whole network on a larger dataset including 30% of the data used for the Stage III.

	Datasets	**Size**
Stage I	LLaVA 1.5-558k (Liu et al., 2024b)	558k
Stage II	LLaVA 1.5-mix-665k (30%) (Liu et al., 2024b) SAM (30%) (Kirillov et al., 2023) Wikiart (30%) (Saleh & Elgammal, 2015) LVIS (Wang et al., 2023) ALLaVA (Chen et al., 2024) TextVQA (Singh et al., 2019)	1,206k
Stage III	LLaVA 1.5-mix-665k (Liu et al., 2024b) SAM (Kirillov et al., 2023) Wikiart (Saleh & Elgammal, 2015)	750k

B.4 Hyperparameters, Implementation, and Training Details

This appendix outlines the hyperparameters, implementation specifics, and training procedures employed for the experiments discussed in Section 2.2.

All models were trained on a distributed setup utilizing either 4 or 8 NVIDIA A100 GPUs. A consistent total batch size of 128 was maintained across all experiments. When using 4 GPUs, the per-device batch size was set to 8, complemented by 4 gradient accumulation steps. In the 8-GPU configuration, the per-device batch size remained 8, but with 2 gradient accumulation steps. For efficient distributed training, we leveraged DeepSpeed with ZeRO-2 offloading.

The models were optimized using the AdamW optimizer, configured with $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The learning rate was varied across training stages: set to 1×10^{-3} during Stage I, and reduced to 2×10^{-5} for both Stage II and Stage III in all experiments. A cosine learning rate scheduler was used, with a warmup ratio of 0.03.

For our proposed method, we set the DPSL coefficient to $\lambda = 0.01$, adopting the standard value used for load balancing losses without further tuning. The baseline methods were implemented following the descriptions provided in their respective original publications, and we generally adopted the hyperparameters recommended by their authors. Specifically, the weight for the standard load-balancing loss (Shazeer et al., 2017; Fedus et al., 2022) was set to 0.01, and the weight for the z-loss (Zoph et al., 2022) regularizer was 0.001. Following the DeepSeek-V3 Technical Report (Wang et al., 2024; Liu et al., 2024a), we evaluated two update rates ($u = 0.001$ and $u = 0.0001$) for the auxiliary-loss-free DeepSeek strategy and selected the one that yielded the highest final accuracy, even though it produced less balanced routing than all other baselines as can be seen in Table 9. For the Drop-Upcycling (Nakamura et al., 2025) baseline, we encountered instabilities and training freezes with the initially recommended 50% drop rate settings. Consequently, we adjusted the ratios of re-initialized parameters. We found that a re-initialization ratio of 0.5 worked best for the 4-expert setup, but this value led to instabilities in the granular setup with 16-experts. Thus, for the 16-expert setup, we used a smaller re-initialization ratio of 0.2 to ensure stable training, and reported highest accuracies.

B.5 Computational Overhead of DPSL

Computing the DPSL loss introduces an additional minimal overhead during training. There are several factors help mitigate this overhead. First, the gradient computation for DPSL is efficient, as the derivative of the CDF used in the loss calculation is simply the PDF, which is already available from the forward pass. Second, empirical observations indicate that the router distributions converge to the target shape early in training and remain stable thereafter. Consequently, DPSL loss was applied only during the warm-up phase and relaxed during final fine-tuning, thereby minimizing its impact on overall training time.

To quantify this overhead precisely, we profiled the forward pass on a Qwen2.5-0.5B MoE model across varying expert counts (4, 8, and 16) using a Pytorch implementation. As shown in Table 4, the standard implementation introduces a modest overhead of 7–12% depending on the number of experts, primarily due to CPU-GPU transfers for SciPy-based Beta CDF computation. However, when using DeepSpeed with ZeRO offloading (Table 5), this overhead drops significantly to just 1–5%. For larger models, such as Qwen2.5-1.5B, the relative cost becomes smaller as the constant-time CDF operation is dwarfed by the model's forward/backward pass.

Finally, DPSL is active only during part of the warm-up phase (Stage II), which accounts for ≈32% of our training pipeline. Using DeepSpeed, a 1-5% overhead in this stage translates to a trivial 0.3-1.6% increase in total wall-clock time.

Table 4: Qwen2.5-0.5B MoE forward pass overhead (avg. of 5 runs)

Setting	w/o DPSL (ms)	w/ DPSL (ms)	Overhead
4 experts	189	203	14ms (+7%)
8 experts	272	299	27ms (+10%)
16 experts	434	487	52ms (+12%)

Table 5: Qwen2.5-0.5B MoE forward pass overhead using DeepSpeed ZeRO-3 (avg. of 5 runs)

Setting	w/o DPSL (ms)	w/ DPSL (ms)	Overhead
4 experts	502	507	5ms (+1%)
8 experts	797	825	28ms (+3.5%)
16 experts	1358	1436	78ms (+5%)

C Ablation Studies

C.1 Concentration parameter

For this ablation, we utilized the upcycled Llama and Qwen models with 4 experts and top-2 routing. We performed a study over $\alpha \in \{0.75, 1.0, 1.25, 1.5\}$. The results in Table 6 show optimal performance at $\alpha = 0.75$, suggesting a benefit from priors encouraging slightly sparser routing than uniform. Ablation results for Qwen2-1.5B are shown in Table 7. We can observe that this larger model is robust across a wider range of α values.

Table 6: The impact of the Dirichlet prior parameter $\boldsymbol{\alpha}$ on Llama3.2-1B (2in4) performance.

Prior	TextVQA	GQA	MM-Vet	MME-P	MME-C	VizWiz	MMB	Avg
$\alpha = 0.75$	**52.82**	**60.98**	**31.7**	**1334.78**	253.21	**62.78**	**42.19**	**35.92**
$\alpha = 1.0$	51.68	60.84	30.2	1310.57	243.93	61.94	38.43	34.86
$\alpha = 1.25$	51.48	60.53	29.2	1236.36	227.86	61.32	41.01	34.92
$\alpha = 1.5$	51.45	60.85	31.4	1294.43	**256.43**	61.94	37.75	34.91

In practice, the optimal value of α can be identified using a reduced-scale experiment on a small subset of the training set (e.g. one-quarter of the warmup data requiring ∼3 hours). The task training loss provides a reliable indicator of the best choice of α. For example, for Llama3.2-1B, training on this subset yields losses of 1.04, 1.08, and 1.06 for $\alpha \in \{0.75, 1.0, 1.25\}$, respectively, pointing at $\alpha = 0.75$ as the optimal point. For reference, we report downstream benchmark results for these intermediate checkpoints in Table 8. While absolute scores are naturally lower than the fully fine-tuned results in Table 8, the relative performance trends across α values mirror those of the full training, confirming the validity of this efficient selection method. Finally, for simplicity and consistency, we apply the same selected α across both standard (2in4) and granular (8in16) MoE configurations: $\alpha = 1$ for Qwen and $\alpha = 0.75$ for Llama.

Table 7: The impact of the Dirichlet prior parameter $\boldsymbol{\alpha}$ on Qwen2-1.5B (8in16) performance.

Prior	TextVQA	GQA	MM-Vet	MME-P	MME-C	VizWiz	MMB	Avg
$\alpha = 0.75$	54.11	**62.08**	32.6	1427.13	**280**	66.20	43.18	37.03
$\alpha = 1.0$	**54.32**	61.86	34.0	1421.90	265	65.98	**43.54**	37.25
$\alpha = 1.25$	53.79	62.05	35.1	**1428.87**	276	**66.48**	41.53	37.14
$\alpha = 1.5$	53.42	62.02	**36.5**	1402.43	270	65.70	42.31	**37.28**

Table 8: Selecting α using a small subset of training data during the warmup stage.

Prior	TextVQA	GQA	MM-Vet	MME-P	MME-C	MMB	Avg
$\alpha = 0.75$	45.91	53.67	29.5	1169.85	242.14	51.28	30.21
$\alpha = 1.0$	44.65	53.47	27.2	1158.67	268.57	51.40	29.60
$\alpha = 1.25$	45.54	53.74	28.4	1180.98	261.78	52.18	30.12

C.2 Practical guidelines for selecting α.

We view the Dirichlet concentration α as a semantic prior for the desired routing profile. We recommend a symmetric unit prior ($\alpha = 1$) as a robust universal default. This setting inherently provides a balanced loading profile for all experts and yields competitive performance across diverse LLM architectures. We observe that slightly smaller values can be used on smaller backbones to encourage marginally stronger specialization. If fine-grained tuning is desired, we show in Appendix C.1 that the optimal α can be reliably identified using a small calibration set (approx. a few hours of training).

For asymmetric priors, where a specific expert or cluster k is preferred (e.g., due to known modality imbalance or domain importance), we recommend adding a moderate scalar bias to that component's concentration. This approach, which we successfully verified in MoE and unsupervised learning experiments, softly biases probability mass toward the target component without enforcing rigid hard constraints.

C.3 Visualization of Routing Score Distributions

As discussed in Section 2.5, we analyze the impact of different upcycling and regularization strategies on the routing score distributions within our upcycled VLM MoEs. Figure 6 provides a visualization of the routing score distributions for 4 experts at the 12th intermediate layer of a Llama3.2-1B model configured with 4 experts and top-2 routing. The routing scores presented in this visualization were collected during model evaluation on the MM-Vet benchmark (Yu et al., 2024).

The figure compares several training approaches: Standard sparse upcycling (Komatsuzaki et al., 2023), load-balancing (Shazeer et al., 2017; Fedus et al., 2022), auxiliary-loss-free DeepSeek balancing (Wang et al., 2024; Liu et al., 2024a), and z-loss (Zoph et al., 2022). Notably, these approaches tend to produce similar routing score distributions across the experts. Each distribution exhibits a prominent peak around a score of 0.25, corresponding to a uniform probability distribution if the router were to assign equal preference to each of the four available experts. This suggests a lack of strong differentiation or specialization among them.

In contrast, our DPSL, when applied with a symmetric prior for all experts, results in visibly different routing score distributions. The distributions generated by DPSL are more dispersed and cover a wider range of score values. This indicates that DPSL encourages the router to make more varied and potentially more confident assignments, fostering a greater degree of specialization or differentiation in how tokens are directed to the experts.

C.4 Expert Specialization Patterns

We analyze expert activation similarity patterns for Llama3.2-1B (2in4) across layers 3, 6, 9, and 12 using cosine similarity between expert outputs on 50 MMVet randomly selected sequences. Low

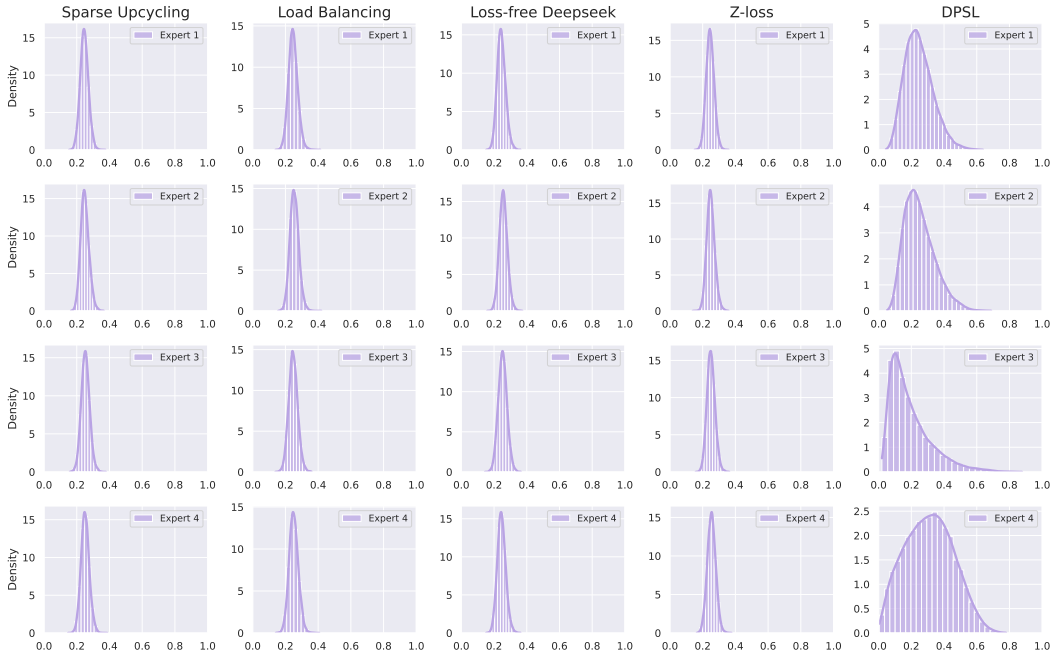


Figure 6: Routing score distributions at layer 12 of an upcycled Llama3.2-1B model (4-expert, top-2 routing). Each column represents a different upcycling/regularization method, and each row displays the distribution for one of the four experts under that method.

similarity values (< 0.4, red) indicates distinct expert specialization, while higher similarity values ($0.4 - 0.8$, yellow-green) suggest overlapping expert behaviors and reduced differentiation. As shown in Figure 7, DPSL demonstrates superior expert differentiation (average similarity: 0.39). In contrast, Load Balancing loss and Router z-loss shows progressive expert convergence from early to deep layers (average similarity: 0.57 and 0.59, respectively). These results clearly indicate that our method effectively prevents expert redundancy and maintains expert specialization across all layers where traditional auxiliary losses struggle to maintain diversity.

C.5 Expert Utilization Analysis

To analyze the expert utilization load, we measure Coefficient of Variation (CoV). As the results show, DPSL achieves low CoV scores, competitive with explicit load-balancing techniques. Although DPSL does not contain an explicit load-balancing term, a symmetric Dirichlet prior intrinsically encourages a balanced load distribution, which we observe across layers.

In all of our experiments, DPSL successfully prevented expert collapse and significant utilization imbalance. We note, however, that none of the baseline methods exhibited severe imbalance issues in our experiments.

Table 9: Coefficient of Variation of expert loads for Llama3.2-1B (2in4) model. Lower values indicate more balanced utilization.

Layer	Sparse Upcycling	Load Balancing	z-loss	DeepSeek	DPSL (Ours)
Layer 4	0.071	0.070	0.088	0.440	0.035
Layer 8	0.075	0.126	0.064	0.191	0.069
Layer 12	0.072	0.091	0.031	0.397	0.057
Layer 16	0.071	0.053	0.103	0.229	0.110

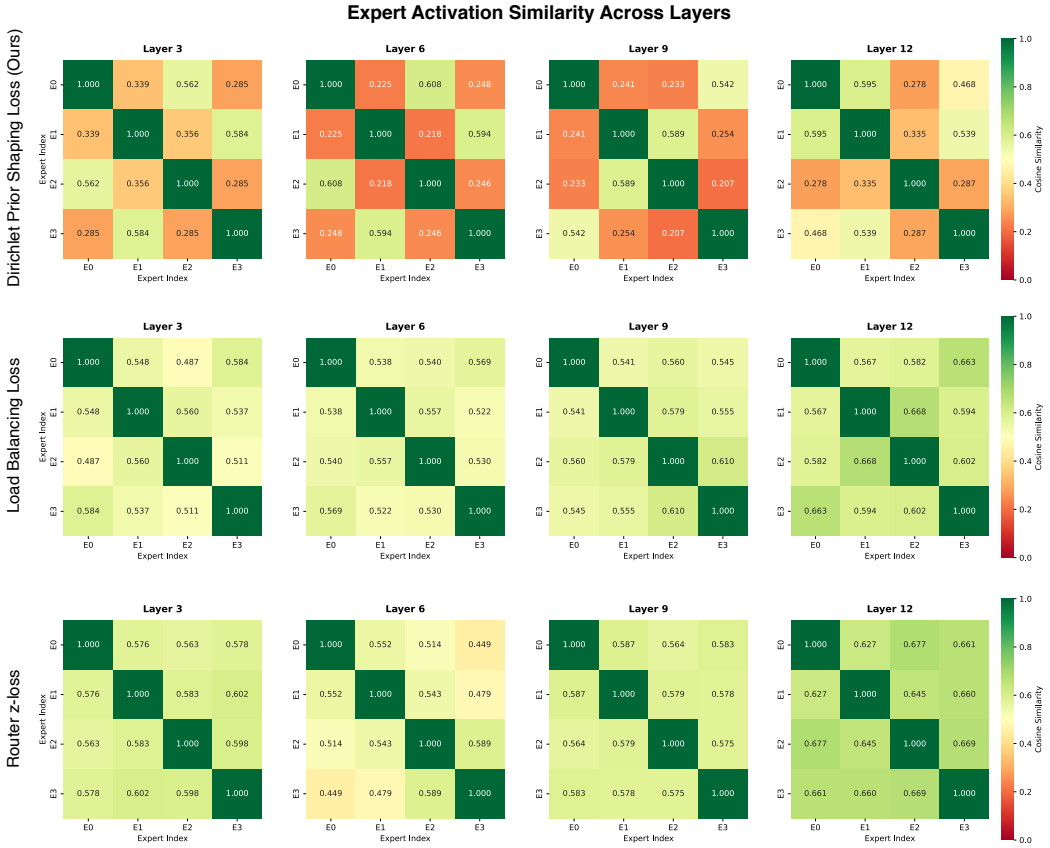


Figure 7: Expert activation similarity scores across layers 3, 6, 9, and 12. Our DPSL (top row) achieves the lowest average similarity (0.39) for these layers compared to Load Balancing loss (0.57) and Router z-loss (0.59), demonstrating superior expert specialization.

C.6 Impact of regularization weight.

For our proposed method, we set the DPSL coefficient to $\lambda = 0.01$ across all experiments, adopting the standard value used for load balancing losses without further tuning. Yet, we provide an ablation on the effect of λ regularization weight in Table 10.

Table 10: The impact of the loss regularization weight parameter $\boldsymbol{\lambda}$ on Llama3.2-1B (2in4) performance.

λ value	TextVQA	GQA	MM-Vet	MME-P	MME-C	VizWiz	MMB	Avg
$\lambda = 0.001$	51.3	60.9	28.7	1286	236.8	60.4	**44.0**	35.2
$\lambda = 0.01$	**52.8**	**61.0**	**31.7**	**1335**	**253.2**	**62.8**	42.2	**35.9**
$\lambda = 0.1$	50.9	60.6	30.5	1261	241.4	61.2	39.6	34.8