Multimodal Contrastive Learning with LIMoE: the Language-Image Mixture of Experts

Basil Mustafa*, **Carlos Riquelme***, **Joan Puigcerver***, **Rodolphe Jenatton**, **Neil Houlsby**
Google Brain
{basilm, rikel, jpuigcerver, rjenatton, neilhoulsby}@google.com

Abstract

Large sparsely-activated models have obtained excellent performance in multiple domains. However, such models are typically trained on a single modality at a time. We present the Language-Image MoE, LIMoE, a sparse mixture of experts model capable of multimodal learning. LIMoE accepts both images and text simultaneously, while being trained using a contrastive loss. MoEs are a natural fit for a multimodal backbone, since expert layers can learn an appropriate partitioning of modalities. However, new challenges arise; in particular, training stability and balanced expert utilization, for which we propose an entropy-based regularization scheme. Across multiple scales, we demonstrate remarkable performance improvement over dense models of equivalent computational cost. LIMoE-L/16 trained comparably to CLIP-L/14 achieves 78.6% zero-shot ImageNet accuracy (vs. 76.2%), and when further scaled to H/14 (with additional data) it achieves 84.1%, comparable to state-of-the-art methods which use larger custom per-modality backbones and pre-training schemes. We analyse the quantitative and qualitative behavior of LIMoE, and demonstrate phenomena such as differing treatment of the modalities and the organic emergence of modality-specific experts.

1 Introduction

Sparsely activated mixture of expert (MoE) models have recently been used with great effect to scale up both vision [1, 2] and text models [3, 4]. The primary motivation for using MoEs is to scale model parameters while keeping compute costs under control. These models however have other benefits; for example, the sparsity protects against catastrophic forgetting in continual learning [5] and can improve performance for multitask learning [6] by offering a convenient inductive bias.

Given success in each individual domain, and the intuition that sparse models may better handle distinct tasks, we explore the application of MoEs to multimodal modelling. We take the first step in this direction, and study models that process both images and text. In particular, we train a single multimodal architecture that aligns image and text representations via contrastive learning [7].

When using a setup proposed in prior unimodal models [8, 1], we find that feeding multiple modalities to a single architecture leads to new failure modes unique to MoEs. To overcome these, we present a set of *entropy based regularisers* which stabilise training and improve performance. We call the resulting model LIMoE (Language-Image MoE).

We train a range of LIMoE models which significantly outperform compute-matched dense baselines. We scale this up to a large 5.6B parameter LIMoE-H/14, which applies 675M parameters per token. When evaluated zero-shot [7] on ImageNet-2012 [9] it achieves an accuracy of 84.1%, competitive with two-tower models that make use of modality-specific pre-training and feature extractors, and apply 3-4x more parameters per token.

*Authors contributed equally.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

In summary, our contributions are as follows.

- We propose LIMoE, the first large-scale multimodal mixture of experts models.
- We demonstrate in detail how prior approaches to regularising mixture of experts models fall short for multimodal learning, and propose a new entropy-based regularisation scheme to stabilise training.
- We show that LIMoE generalises across architecture scales, with relative improvements in zero-shot ImageNet accuracy ranging from 7% to 13% over equivalent dense models. Scaled further, LIMoE-H/14 achieves 84.1% zero-shot ImageNet accuracy, comparable to SOTA contrastive models with per-modality backbones and pre-training.
- Lastly, we present ablations and analysis to understand the model's behavior and our design decisions.

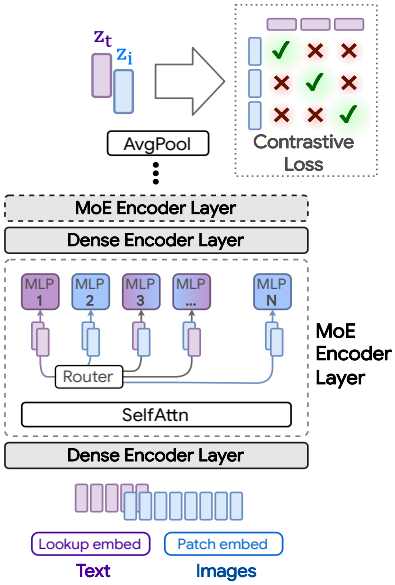


Figure 1: LIMoE, a sparsely activated multimodal model, processes both images and texts, utilising conditional computation to allocate computations in a modality-agnostic fashion.

2 Multimodal Mixture of Experts

Multimodal contrastive learning typically works with *independent* per-modality encodings [7, 10]. That is, separate models f_m are trained to provide a final representation for every input from the corresponding modality, m. In the case of some image and text inputs, $\mathbf{i}$ and $\mathbf{t}$, we have $\mathbf{z_i} = f_{\text{image}}(\mathbf{i})$ and $\mathbf{z_t} = f_{\text{text}}(\mathbf{t})$. For contrastive learning with images and text, this approach results in a "two-tower" architecture, one for each modality. We study a one-tower setup instead, where a *single* model is shared for all modalities, as shown in Figure 1. The one-tower design offers increased generality and scalability, and the potential for cross-modal and cross-task knowledge transfer. We next describe the LIMoE architecture and training routine.

2.1 Multimodal contrastive learning

Given n pairs of images and text captions $\{(\mathbf{i}_j, \mathbf{t}_j)\}_{j=1}^n$, the model learns representations $\mathcal{Z}_n = \{(\mathbf{z_i}_j, \mathbf{z_t}_j)\}_{j=1}^n$ such that those corresponding to paired inputs are closer in feature space than those of unpaired inputs. The contrastive training objective [7, 11], with learned temperature T, is:

$$\mathcal{L}_j(\mathcal{Z}_n) = \underbrace{-\frac{1}{2}\log\frac{e^{\langle \mathbf{z_i}_j, \mathbf{z_t}_j\rangle/T}}{\sum_{k=1}^n e^{\langle \mathbf{z_i}_j, \mathbf{z_t}_k\rangle/T}}}_{\text{image-to-text loss}} \underbrace{-\frac{1}{2}\log\frac{e^{\langle \mathbf{z_i}_j, \mathbf{z_t}_j\rangle/T}}{\sum_{k=1}^n e^{\langle \mathbf{z_i}_k, \mathbf{z_t}_j\rangle/T}}}_{\text{text-to-image loss}}. \tag{1}$$

2.2 The LIMoE Architecture

We use a single Transformer-based architecture for both image and text modalities. The model uses a linear layer per modality to project the intrinsic data dimension to the desired width: for text, a standard one-hot sentencepiece encoding and learned vocabulary [12], and for images, ViT-style patch-based embeddings [13]. Then all tokens are processed by a shared transformer encoder, which is not explicitly conditioned on modality. The token representations from the final layer are average-pooled to produce a single representation vector $\mathbf{z}_m$ for each modality. To compute the training loss in (1), the paired image and text representations are then linearly projected using per-modality weight matrices $\mathbf{W}_m$'s and $\mathcal{L}_j$ is applied to $\{(\mathbf{W}_{\text{image}}\,\mathbf{z_i}_k, \mathbf{W}_{\text{text}}\,\mathbf{z_t}_k)\}_{k=1}^n$.

This one-tower setup can be implemented with a standard dense Transformer (and we train many such models as baselines). Next, we describe how we introduce MoEs to this setup for LIMoE.

Sparse MoE backbone: Sparse MoE layers are introduced following the architectural design of [1, 3]. The *experts*—parts of the model activated in an input-dependent fashion—are MLPs. LIMoE contains multiple MoE layers. In those layers, each token $\mathbf{x} \in \mathbb{R}^D$ is processed sparsely by K out of E available experts. To choose which K, a lightweight router predicts the gating weights *per token*:

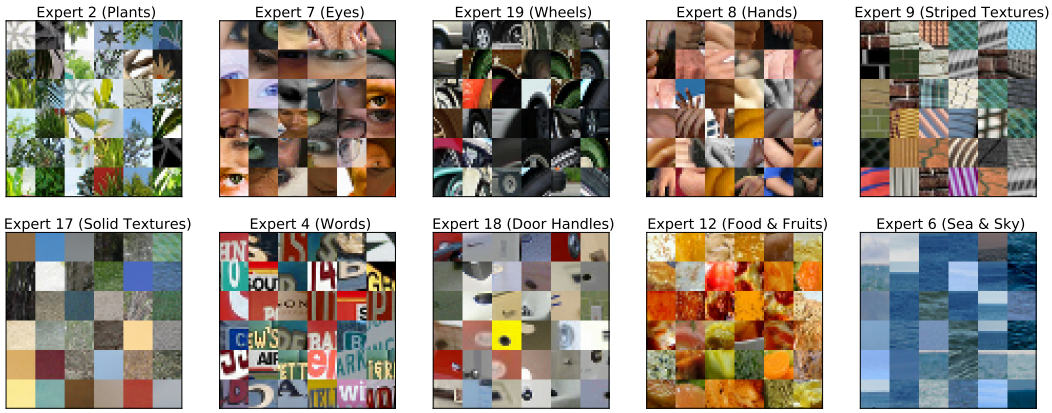


Figure 2: **Token routing examples for Coco.** Image examples of how patches are routed at the MoE layer placed in the 18-th encoder block –i.e. middle of the network– for the LIMoE-H/14 model.

$g(\mathbf{x}) = \texttt{softmax}(\mathbf{W}_g\mathbf{x}) \in \mathbb{R}^E$ with learned $\mathbf{W}_g \in \mathbb{R}^{D\times E}$. The outputs of the K activated experts are linearly combined according to the gating weights: $\texttt{MoE}(\mathbf{x}) = \sum_{e=1}^{K} g(\mathbf{x})_e \cdot \texttt{MLP}_e(\mathbf{x})$.

Note that, for computational efficiency and implementation constraints, experts have a *fixed buffer capacity*. The number of tokens each expert can process is fixed in advance, and typically assumes that tokens are roughly balanced across experts. If capacity is exceeded, some tokens are "dropped"; they are not processed by the expert, and the expert output is all zeros for those tokens. The rate at which tokens are successfully processed (that is, not dropped) is referred to as the "success rate". It is an important indicator of healthy and balanced routing and often indicative of training stability.

We discovered that routing with tokens from multiple modalities introduces new failure modes; in the next sections we demonstrate this phenomenon, and describe our techniques to address it.

2.2.1 Challenges for multimodal MoEs

As mentioned, experts have a fixed buffer capacity. Without intervention, Top-K MoEs tend to "collapse", thus using only one expert. This causes most tokens to be dropped and leads to poor performance [14]. Prior works therefore use auxiliary losses to encourage balanced routing [1, 3, 8].

In multimodal settings, new challenges arise; one is modality misbalance. In realistic setups, there will likely be more of one data type than another. Accordingly, we do not assume or enforce balanced data across modalities, and our experiments have $3 - 17\times$ more image tokens than text tokens.

Modality-specific experts tend to emerge naturally. In this imbalanced context, this leads to a scenario where all of the tokens from the minority modality get assigned to a single expert, which runs out of capacity. On a global level, routing still appears balanced: tokens from the majority modality are nicely distributed across experts, thereby satisfying modality-agnostic auxiliary losses. For example, in our standard B/16 setup, the router can optimize the importance loss [14] to within 0.5% of its minimum value by perfectly balancing image tokens but dropping all text tokens. This however leads to unstable training and unperforming models.

2.2.2 Auxiliary losses

We refer to auxiliary losses used in V-MoE [1] as the *classic* auxiliary losses. We find that they do not yield stable and performant multimodal MoE models. Therefore, we introduce two new losses: the *local entropy loss* and the *global entropy loss*, which are applied on a per-modality basis. We combine these losses with the classic losses; see Appendix B for a summary of all auxiliary losses.

Definition. In each MoE layer, for each modality m, the router computes a gating matrix $\mathbf{G}_m \in \mathbb{R}^{n_m\times E}$. Each row of $\mathbf{G}_m$ represents the probability distribution over E experts for one of the n_m tokens of that modality in the batch. For a token $\mathbf{x}$ that corresponding row is $p_m(\texttt{experts}|\mathbf{x}) \in \mathbb{R}^E$;

this later dictates which experts process $\mathbf{x}$. The local and global entropy losses are defined by:

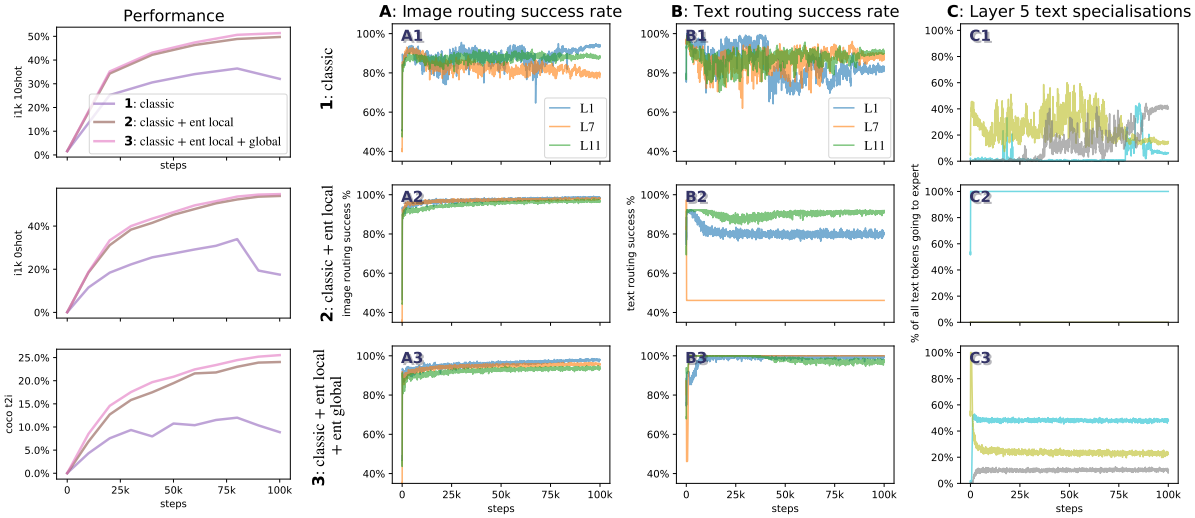
$$\Omega_{\text{local}}(\mathbf{G}_m) := \frac{1}{n_m}\sum_{i=1}^{n_m}\mathcal{H}(p_m(\texttt{experts}|\mathbf{x}_i)) \;\text{ and }\; \Omega_{\text{global}}(\mathbf{G}_m) := -\mathcal{H}(\tilde{p}_m(\texttt{experts}))\,, \tag{2}$$

where $\tilde{p}_m(\texttt{experts}) = \frac{1}{n_m}\sum_{i=1}^{n_m} p_m(\texttt{experts}|\mathbf{x}_i)$ is the expert probability distribution averaged over the tokens and $\mathcal{H}(p) = -\sum_{e=1}^{E} p_e \log(p_e)$ denotes the entropy. Note that $\tilde{p}_m(\texttt{experts}) \approx p_m(\texttt{experts})$ since we approximate the true marginal from the tokens in the batch. We use the terminology *local* vs. *global* to emphasise the fact that Ω_{local} applies the entropy *locally* for each token while Ω_{global} applies the entropy *globally* after having marginalized out the tokens.

Effects of the losses. Figure 3 shows why these losses are necessary. With the default losses, modality-specific experts naturally emerge, but the router often changes its preference. This results in unstable training and poor success rate, particularly for the text modality. The local entropy loss encourages concentrated router weights ($p_{\text{text}}(\texttt{experts}|\mathbf{x}_i)$'s have low entropy), but at the expense of the *diversity* of the text experts: the same expert is used for all text tokens (the marginal $\tilde{p}_{\text{text}}(\texttt{experts})$ also has low entropy), leading to dropping. In this setup, many layers have poor text success rates.

To address this, Ω_{global} encourages maximization of the marginal entropy, thus pushing $\tilde{p}_{\text{text}}(\texttt{experts})$ towards a more uniform expert distribution. The result is diverse expert usage, stable and confident routing, and high success rates. These are consequently the most performant models.

Intuitively, it is desirable for text tokens to use multiple experts, but not all of them. In order to allow flexibility, we threshold the global entropy loss as $\Omega^{\tau}_{\text{global}}(\mathbf{G}_m) = \max\{0, \tau + \Omega^{\text{global}}(\mathbf{G}_m)\}$, such that the model is encouraged to have a certain minimum entropy, but after exceeding that, the loss is not applied. This avoids distributional collapse but does not apply overly restrictive priors on the routing distribution, as there are many optimal solutions. This can be thought of as a "soft minimum" S. With $\tau = \log(S)$, the model must use at least S experts to minimize the loss (either a uniform distribution across S experts -with entropy $\log(S)$-, or a non-uniform distribution using more than S). Figure 3b shows the latter occurs; the empirical effect of these thresholds is analysed in Section 4.1.



(a) Performance w.r.t. aux losses.

(b) Analysing routing behaviour of the auxiliary losses. *First column*: Average success rate of image routing in layers 1/7/11. *Second column*: Same, for text. *Third column*: In some experts of layer 5, what fraction of all text tokens go to those experts

Figure 3: **What necessitates entropy losses?** *Classic* refers to the standard formulation (importance + load losses [1]). We add the local entropy loss to text tokens (middle row), followed by the global entropy loss (bottom row). **Left:** The "classic" setting is low-performing and unstable. **Right:** Analyzing the entropies shows us why: Without the local loss, the model is prone to unstable changes in expert preferences (C1), and routing success rates are low (A1, B1). The local loss fixes this but causes distributional collapse for one modality (C2), with all text tokens going to one expert (expert 11); this causes even poorer text success rates (B2). This is addressed by the global loss, which has stable expert allocations (C3) and consistently high success rates (A3, B3).

Connection with mutual information. The sum $\Omega_{\text{local}}(\mathbf{G}_m) + \Omega_{\text{global}}(\mathbf{G}_m)$ corresponds to the (negative) mutual information [15] between experts and tokens, conditioned on the modality m, which we write $-\text{MI}_m(\texttt{experts}; \texttt{x})$. For each modality taken separately, we are effectively encouraging the knowledge of the token representation to reduce the uncertainty about the experts selection. We also tried other variants of the losses which exploit this connection, such as the mutual information between the experts and modalities, $-\text{MI}(\texttt{experts}; m)$, obtained by first marginalizing the tokens.

2.2.3 Priority routing

With Top-K routing, some token dropping is virtually inevitable. Batch Priority Routing (BPR) [1] actively decides which tokens to skip based on their routing weights. It assumes that tokens with a large routing weight are likely to be informative, and should be favored. BPR was mostly used at inference time in [1], allowing for smaller expert capacity buffers. In this setup, one must take care not to systematically favor one modality over the other, for instance, by determining which token to drop based on their rank in the batch, which are usually grouped according to the token modality. BPR provides an essential stabilisation effect during training (Figure 6); we show that it does not trivially rank one modality over another, and it cannot be replaced by other methods of re-ordering the batch. In the appendix we further show how routing priorities compare across text and images.

3 Experiments

We study LIMoE in the context of multimodal contrastive learning. We first perform a controlled comparison of LIMoE to an equivalent "standard" dense Transformer, across a range of model sizes. We then show that when scaled up LIMoE can reach a high level of performance. Finally, we ablate the various design decisions leading to LIMoE in Section 4.

Training data. By default, all models are trained on paired image-text data used in [16], consisting of 3.6B images and alt-texts scraped from the web. For large LIMoE-H/14 experiment, we also co-train with JFT-4B [17]. We construct artificial text captions from JFT by comma-delimited concatenation of the class names [18]. Appendix A contains full details of our training setup.

Evaluation. Our main evaluation is "zero-shot": the model uses its text representations of the classes to make predictions on a new task without extra training data [19, 7]. We focus on image classification accuracy on ImageNet [9] and cross-modal retrieval on MS-COCO [20], following the protocol in [16]. We also evaluate LIMoE's image representations via a linear adaptation protocol [13], and report 10-shot accuracy on ImageNet accuracy accordingly. Where ranges are given, they report 95% confidence intervals across three trials.

3.1 Controlled study across scales

We train a range of LIMoE models at batch size 16k for 781k steps. This matches the number of training examples used for CLIP [7]. Due to use of different training data and additional tricks, a direct comparison is difficult; we therefore train dense one-tower models as baselines. All models activate $k = 1$ experts per token, similar to Switch Transformer [8].

Figure 4 shows the performance of each model (dense and sparse) against forward-pass FLOPs (for step times and further discussion on compute costs, see Appendix D.2.). The cost-performance Pareto frontier for LIMoE dominates the dense models by a wide margin, indicating that LIMoE offers strong improvements across all scales from S/32 , up to L/16. The effect is particularly large on zero-shot and 10-shot ImageNet classification, with absolute performance improvements of **10.1%** and **12.2%** on average. For text-to-image retrieval on COCO, LIMoE offers a strong boost at small scales, while at larger scales the gains are more modest but still significant.

3.2 Scaling up LIMoE

We increase the architecture size, training duration, and data size to assess the performance of LIMoE in the large-scale regime. In particular, we train a 32-layer LIMoE-H/14 with 12 expert layers; these are non-uniformly distributed, with 32 experts per layer, and $K = 1$ activated per token. It was trained at a batch size of 21k, introducing 25% JFT-4B images [17] into each batch (with class names as texts). We average checkpoints towards the end of training [21]; refer to Appendix A.3 for details.

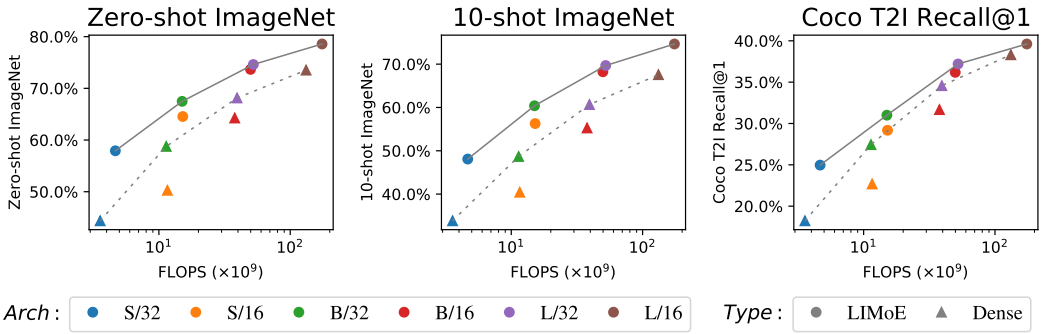


Figure 4: LIMoE scales well to large models, with consistent performance improvements.

The model contains 5.6B parameters in total, but only applies 675M parameters per token. All routers combined account for less than 0.5M parameters. Table 1 shows its performance alongside current state-of-the-art contrastive models. LIMoE achieves 84.1% zero-shot ImageNet classification accuracy with a comparably modest architecture size and training counts. LIMoE is fully trained from scratch, without any pre-trained components, and is the first competitive model with a shared backbone.

In light of its modality agnostic approach, this result is surprisingly strong. Large models handling dozens of distinct tasks are increasingly popular [22], but do not yet approach the state-of-the-art in these tasks. We believe the ability to build a generalist model with specialist components, which can decide how different modalities or tasks should interact, will be key to creating truly multimodal multitask models which excel at everything they do. LIMoE is a promising first step in that direction.

Table 1: Comparing state of the art zero-shot classification models. At a relatively modest scale, LIMoE-H/14 is comparable with the best two-tower models, and it is the first performant one-tower model at this scale. T-x refers to a Transformer [23] with the equivalent parameters of ViT-x [13].
Key: ** Pretrained* *^{PT} Examples seen during pretraining* *† Uses FixRes [24]* *§ Other non-contrastive training objective*

	Architecture		Batch	Examples seen	Parameters	ImageNet top-1 %			
	Image	Text	size		per token	Test	V2	R	A
COCA§ [25]	ViT-g	T-g	65k	32.8B	1.1B	86.3	80.7	96.5	90.2
BASIC [18]	CoAtNet-7*	T-H*	65k	19.7B^{PT} +32.8B	1.5B	85.7	80.6	95.7	85.6
LIT [16]	ViT-g*	T-g	32k	25.8B^{PT} + 18.2B	1.1B	84.5	78.7	93.9	79.4
ALIGN [10]	EffNet-L2	T-L*	16k	19.8B	$\sim$ 410M	76.4	70.1	92.2	75.8
CLIP [7]	ViT-L/14†	T-B	32k	12.8B	$\sim$ 200M	76.2	70.1	88.9	77.2
LIMoE	H/14		21k	23.3B	675M	84.1	77.7	94.9	78.7

4 Ablations

We use a smaller setup to study various aspects of LIMoE. We train B/16 models at batch size 8096 for 100,000 steps (see Appendix A.2 for further details). Table 2 shows the average over three trials of this setting alongside dense one-tower and two-tower baselines. LIMoE greatly outperforms both dense models on ImageNet 0- and 10-shot, while confidence intervals overlap for retrieval with two towers. The two-tower model is twice as large and expensive, and still falls behind the sparse one.

4.1 Routing and auxiliary losses

Choice of auxiliary losses. With the introduction of the entropy based losses in addition to classic ones, there are 7 possible auxiliary losses. We aimed to find the simplest combination of these which obtains good performance. To study this, we performed a large sweep of auxiliary losses: for $N \in [2, \ldots, 5]$, we considered all $\binom{7}{N}$ possible loss combinations. Table 3 shows, for each loss, the highest performing model with and without that loss. Some conclusions stand out: Both entropy losses are important for text, but for images, the global loss is not impactful and the local

Table 2: Baselines for ablations: B/16 with batch size 8096 trained for for 100,000 steps. 0shot and 10shot columns show accuracy (%), t2i and i2t show recall@1 (%).

Model	i1k 0shot	i1k 10shot	coco t2i	coco i2t
dense one-tower	49.8 $^{50.4}_{49.2}$	43.8 $^{44.3}_{43.3}$	23.7 $^{24.0}_{23.4}$	36.7 $^{38.9}_{34.6}$
dense two-tower	54.7 $^{55.2}_{54.1}$	47.1 $^{47.6}_{46.7}$	26.6 $^{27.1}_{26.2}$	41.3 $^{42.0}_{40.6}$
LIMoE	56.9 $^{57.1}_{56.7}$	50.5 $^{50.8}_{50.2}$	25.6 $^{27.3}_{23.9}$	39.7 $^{42.2}_{37.1}$

Table 3: Across 121 combinations, each row shows the best accuracy (%) of all combinations that *included* the auxiliary loss (✓) vs. those that did not (✗). Bold auxiliary losses indicate they are in LIMoE. Validation accuracy is the average contrastive accuracy in a minibatch of size 1024.

	Validation		0shot		10shot	
Auxiliary loss	✗	✓	✗	✓	✗	✓
Importance	70.5	70.6	55.4	56.2	51.1	51.3
Load	70.3	70.6	56.2	55.7	51.3	51.1
Z-Loss	70.3	70.6	55.8	56.2	50.5	51.3
Global Ent Image	70.6	70.5	56.0	56.2	50.8	51.3
Global Ent Text	69.1	70.6	54.3	56.2	51.1	51.3
Local Ent Image	70.6	68.7	56.2	53.5	51.3	47.5
Local Ent Text	67.2	70.6	53.3	56.2	47.5	51.3

loss is harmful. The final combination of losses was chosen based on validation accuracy alongside qualitative observations around training stability and routing success rate.

Threshold for global entropy losses. In Section 2.2.2, we introduced a threshold τ to encourage balanced expert distributions without forcing all modalities to use all experts. To understand the importance of this threshold, we sweep over it for both the image and text global entropy losses. Appendix B.2 contains a full analysis; the most important conclusions are:

- τ_{image} did not affect the number of experts used for images, as global entropy was always high. Aside from these threshold experiments with very high τ_{image}, this loss is usually inactive. It was used in our main experiments, but can likely be removed in future work.
- The threshold τ_{text} behaved exactly as a soft minimum for text experts: Sweeping τ_{text}, we typically observed approximately $S = e^{\tau_{\text{text}}}$ text experts.
- Performance is robust to different values of τ_{text}, provided it is not too low. A low τ_{text} can be useful to limit the number of text experts, for later pruning, see Appendix E.4.

Mutual-information auxiliary loss. In Section 2.2.2, we discussed an alternative loss, namely $-\text{MI}(\texttt{experts}; m)$, based on the mutual information between experts and modalities. While it has the advantage of merging the local and global entropy losses for both the text and image modalities into a single term, without threshold parameters, it leads to slightly worse results: in a comparable setup, it had 1.5% and 0.1% worse zero-shot and 10-shot performance compared to Table 2.

The effect of modality balancing. Our models use a text sequence length of 16, but image sequence lengths from 49 to 400 (for these ablations, 196).

Our ablations reveal that the entropy losses are most important when applied to the text tokens. This leads to a hypothesis that these are only necessary or useful in the imbalanced case. To test this, we vary the modality balance of LIMoE-B/16 by varying the patch size; this enables us to control the number of image tokens, and hence image:text balance, without changing the information content in the data. Fig-

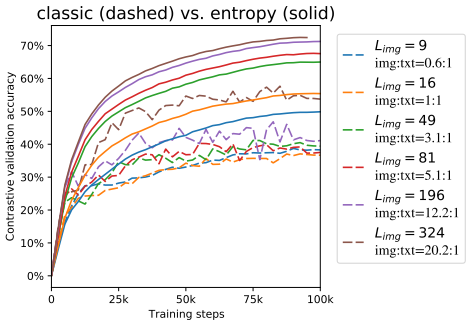


Figure 5: **Entropy losses are not just addressing a modality imbalance**. With different image:text balancing, including completely balanced, the entropy losses substantially improves over the classic setting.

ure 5 shows the results. First, we observe that, with entropy routing, a longer image sequence length is always better. This shows that entropy routing can effectively handle highly imbalanced setups, and mirrors the observation that for classical Vision Transformers: a longer sequence is better. Importantly, entropy routing is always far superior to the classical setup with growing gaps, even when the modalities are balanced 1:1 ($L_{\text{img}} = 16$). This experiment also confirms the robustness of entropy routing to different setups.

Batch priority routing as a training stabilizer. Figure 6 shows the effect of BPR during training. BPR not only ameliorates against token dropping, but also improves training stability. Models with no dispatch order intervention (first-in-first-out) perform extremely poorly, whether we route images first or text first. These routers have low success rate. Randomly shuffling tokens (i.e. deciding which tokens to drop at random when an expert becomes full) partially ameliorates this, but its performance is still much worse than that of models trained with BPR. We further analyse BPR in Appendix F.5 and show that it does not simply rank one modality above another.

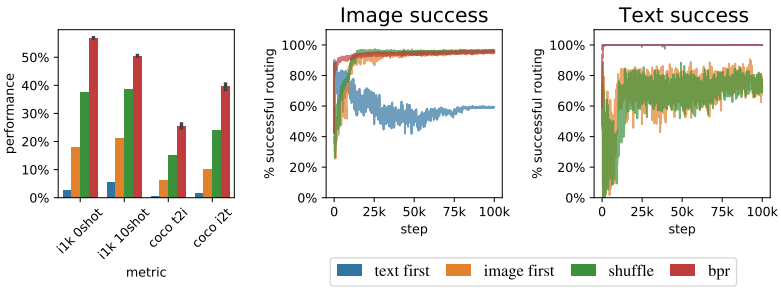


Figure 6: **BPR stabilizies training and enables performant models**; the first figure shows different performance metrics. The last two show *success rates* for the MoE router in Layer 9.

4.2 Other ablations

We summarize our other ablations here due to space constraints; details can be found in Appendix E.

Router structure (*Appendix E.3*). Our router is modality agnostic; we experiment with per-modality routers, and separate pools of per-modality experts. We find they all perform comparably to our generic, modality agnostic setup, but that separate pools of experts by design is more stable and does not require auxiliary losses for regularisation—while harder to scale to many modalities and tasks.

Increasing selected experts per token K (*Appendix E.1*). We propose modifications to BPR and the local auxiliary loss to generalise to $K > 1$; by doing so we can steadily increase performance by increasing K, e.g. from 55.5% zero-shot accuracy with $K = 1$ to 61.0% with $K = 5$.

Total number experts (*Appendix E.2*). We show that increasing the pool of available experts at fixed K improves performance (unlike what was observed for vision-only tasks [1]).

Expert pruning (*Appendix E.4*). We show using simple heuristics we can prune down to modality-specific experts for unimodal forward passes, thus avoiding expert collapse under unimodal batches.

Training on public data (*Appendix E.6*) The majority of `LIMoE` models were trained on proprietary data [16]. We show that LIMoE works similarly well on publically available data, retaining performance improvements against a comparable dense model.

5 Model Analysis

In this section, we explore some of the internal workings of `LIMoE`. We use simple B/32 and B/16 models with 8 experts, and the large H/14 with 32. See Appendix F for further details and experiments.

Multimodal experts arise *(Appendix F.1).* Aside from encouraging diversity, we do not explicitly enforce experts to specialize. Nonetheless, we observe the emergence of both modality-specific experts, and multimodal experts which process both images and texts (per-expert distributions in F.1).

Qualitative analysis *(Appendix F.2).* We analyse some example data and show a clear emergence of semantically meaningful experts. With images for instance, some experts specialize on lower level features (colours, lines) while others on more complex features (faces and text), see Figure 2.

BPR ranking *(Appendix F.5).* The local loss encourages high max-routing weights for text, and BPR ranks according to this. We show however that this does not mean text is always prioritised first: Especially in later layers, the model often prioritises important image patches over text.

6 Related work

Unimodal, task-specific neural networks have long been researched, with increasing convergence towards Transformer-based architectures [23, 26] for both NLP [27] and Computer Vision [13, 28, 29]. *Multimodal models* aim to process multiple types of data using a single neural network.

Many approaches "fuse" modalities [30, 31, 32, 33] to tackle inherently multimodal tasks. `LIMoE` is more similar to approaches which do not do that, and still operate as unimodal feature extractors. Some co-train on distinct tasks [34, 35, 36, 22] without aligning or fusing representations—effectively sharing weights across tasks—whereas others include both unimodal aspects and fused multimodal aspects for functionality in both contexts [37].

We build on deep *Sparse Mixture of Experts models*, which have been studied independently in Computer Vision [1, 2] and NLP [14, 3, 8], typically in the context of transfer learning. These models use a learned gating mechanism whereby only a subset of K experts out of $E \gg K$ are activated for a given input. Many works aim to improve the gating mechanism itself, by making it differentiable [38], reformulating as a linear assignment task [39] or even swapping it out for a simple hashing algorithm [40]. MoE models have also been studied for multitask learning [38], with per-task routers [6] but a shared pool of experts. To our knowledge, sparse models have not been explored for multimodal learning.

A large body of research exists on contrastive learning, usually in self-supervised [41] but also in supervised regimes [42]. *Multimodal contrastive learning* trains on aligned data from multiple modalities. Originally studied for medical images and reports [11], it was recently scaled to noisy web data [7, 10], where strong image-text alignments enabled performant image classification and cross-modal image-text retrieval without finetuning on downstream data. Follow up works improved upon this significantly by scaling up and using pretrained models [18, 16] and multitask training with generative modelling [25] or other vision tasks [43]. These works use unimodal models which *separately* process image and text data; we are not aware of previous research using a single model to process both images and texts for contrastive learning, neither with dense nor with sparse models.

7 Conclusions and Future Work

We have presented `LIMoE`, the first multimodal sparse mixture of experts model. We uncovered new failure modes specific to this setup and proposed entropy based auxiliary losses which stabilises training and results in highly performant models. It works across many model scales, with average improvements over FLOP-matched dense baselines of +10.2% zero-shot accuracy. When scaled to a large H/14 model, we achieve 84.1% accuracy, competitive with current SOTA approaches.

Societal impact and limitations: The potential harms of large scale models [44], contrastive models [7] and web-scale multimodal data [45] also carry over here, as `LIMoE` does not explicitly address them. On the other hand, it has been shown that *pruning* models tends to cause low-resource groups to be forgotten [46], causing performance to disproportionally drop for some subgroups. This would be worth considering for our expert-pruning experiments, but by analogue, the ability to scale models with experts that can specialize deeply may result in better performance on underrepresented groups.

Environmentally speaking, training large models is costly, though efforts are made to use efficient datacenters and offset emitted CO_2. Prior works however show that most environmental impact occurs during model inference, and that MoEs are significantly more efficient in that regard [47]; `LIMoE` is naturally a good candidate for efficient, large-scale multimodal foundation models.

Future work: There are many interesting directions from here. The routing interference with multiple modalities still is not fully understood. In general, conclusions from applications of MoEs

to NLP have not carried over perfectly to Vision, and vice-versa, and here we see again different behaviour between images and text. Naturally, extensions to more modalities should be explored; even with only two we see fascinating interactions between different data types and the routing algorithms, and that will only get more difficult, and interesting, with more modalities.

There are always more modalities to learn, and larger models to build: sparse models provide a very natural way to scale up while juggling very different tasks and data, and we look forward to seeing more research in this area.

8 Acknowledgements

We first thank Andreas Steiner, Xiao Wang and Xiaohua Zhai, who led early explorations into dense single-tower models for contrastive multimodal learning, and also were instrumental in providing data access. We also thank Andreas Steiner, and Douglas Eck, for early feedback on the paper. We thank André Susano Pinto, Maxim Neumann, Barret Zoph, Liam Fedus, Wei Han and Josip Djolonga for useful discussions, and Erica Moreira and Victor Gomes for help scaling up to `LIMoE`-H/14.

References

[1] Carlos Riquelme, Joan Puigcerver, Basil Mustafa, Maxim Neumann, Rodolphe Jenatton, André Susano Pinto, Daniel Keysers, and Neil Houlsby. Scaling vision with sparse mixture of experts. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2021.

[2] Yuxuan Lou, Fuzhao Xue, Zangwei Zheng, and Yang You. Cross-token modeling with conditional computation, 2022.

[3] Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[4] 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, 2022.

[5] Mark Collier, Efi Kokiopoulou, Andrea Gesmundo, and Jesse Berent. Routing networks with co-training for continual learning, 2020.

[6] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H. Chi. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, London, UK, August 19-23, 2018*. ACM, 2018.

[7] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *ICML*, Proceedings of Machine Learning Research. PMLR, 2021.

[8] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *JMLR*, 23(120), 2022.

[9] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*. IEEE Computer Society, 2009.

[10] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc V. Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, Proceedings of Machine Learning Research. PMLR, 2021.

[11] Yuhao Zhang, Hang Jiang, Yasuhide Miura, Christopher D. Manning, and Curtis P. Langlotz. Contrastive learning of medical visual representations from paired images and text. *CoRR*, 2020.

[12] Taku Kudo and John Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018: System Demonstrations, Brussels, Belgium, October 31 - November 4, 2018*. Association for Computational Linguistics, 2018.

[13] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021.

[14] 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. In *5th International Conference on Learning Representations, ICLR 2017*, 2017.

[15] Thomas M Cover. *Elements of information theory*. John Wiley & Sons, 1999.

[16] Xiaohua Zhai, Xiao Wang, Basil Mustafa, Andreas Steiner, Daniel Keysers, Alexander Kolesnikov, and Lucas Beyer. Lit: Zero-shot transfer with locked-image text tuning. *CVPR*, 2021.

[17] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. *CVPR*, 2021.

[18] Hieu Pham, Zihang Dai, Golnaz Ghiasi, Kenji Kawaguchi, Hanxiao Liu, Adams Wei Yu, Jiahui Yu, Yi-Ting Chen, Minh-Thang Luong, Yonghui Wu, Mingxing Tan, and Quoc V. Le. Combined scaling for open-vocabulary image classification, 2022.

[19] Richard Socher, Milind Ganjoo, Christopher D Manning, and Andrew Ng. Zero-shot learning through cross-modal transfer. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2013.

[20] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Doll'ar, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. In *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V*, Lecture Notes in Computer Science. Springer, 2014.

[21] Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. *CoRR*, 2022.

[22] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, Tom Eccles, Jake Bruce, Ali Razavi, Ashley Edwards, Nicolas Heess, Yutian Chen, Raia Hadsell, Oriol Vinyals, Mahyar Bordbar, and Nando de Freitas. A generalist agent, 2022.

[23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2017.

[24] Hugo Touvron, Andrea Vedaldi, Matthijs Douze, and Herv'e J'egou. Fixing the train-test resolution discrepancy. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[25] Jiahui Yu, Zirui Wang, Vijay Vasudevan, Legg Yeung, Mojtaba Seyedhosseini, and Yonghui Wu. Coca: Contrastive captioners are image-text foundation models, 2022.

[26] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[27] Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey. *ACM Comput. Surv.*, 2022.

[28] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[29] Kishaan Jeeveswaran, Senthilkumar Kathiresan, Arnav Varma, Omar Magdy, Bahram Zonooz, and Elahe Arani. A comprehensive study of vision transformers on dense prediction tasks. In *Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, VISIGRAPP 2022, Volume 4: VISAPP, Online Streaming, February 6-8, 2022*. SCITEPRESS, 2022.

[30] Hao Tan and Mohit Bansal. Lxmert: Learning cross-modality encoder representations from transformers. In *EMNLP*, 2019.

[31] Weijie Su, Xizhou Zhu, Yue Cao, Bin Li, Lewei Lu, Furu Wei, and Jifeng Dai. Vl-bert: Pre-training of generic visual-linguistic representations. In *ICLR*, 2020.

[32] Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In *NeurIPS*. Curran Associates, Inc., 2019.

[33] Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. Visualbert: A simple and performant baseline for vision and language. *CoRR*, 2019.

[34] Valerii Likhosherstov, Anurag Arnab, Krzysztof Choromanski, Mario Lucic, Yi Tay, Adrian Weller, and Mostafa Dehghani. Polyvit: Co-training vision transformers on images, videos and audio. *CoRR*, 2021.

[35] Hassan Akbari, Liangzhe Yuan, Rui Qian, Wei-Hong Chuang, Shih-Fu Chang, Yin Cui, and Boqing Gong. Vatt: Transformers for multimodal self-supervised learning from raw video, audio and text. *NeurIPS*, 2021.

[36] Qing Li, Boqing Gong, Yin Cui, Dan Kondratyuk, Xianzhi Du, Ming-Hsuan Yang, and Matthew Brown. Towards a unified foundation model: Jointly pre-training transformers on unpaired images and text. *arXiv preprint arXiv:2112.07074*, 2021.

[37] Jianfeng Wang, Xiaowei Hu, Zhe Gan, Zhengyuan Yang, Xiyang Dai, Zicheng Liu, Yumao Lu, and Lijuan Wang. UFO: A unified transformer for vision-language representation learning. *CoRR*, 2021.

[38] Hussein Hazimeh, Zhe Zhao, Aakanksha Chowdhery, Maheswaran Sathiamoorthy, Yihua Chen, Rahul Mazumder, Lichan Hong, and Ed H. Chi. Dselect-k: Differentiable selection in the mixture of experts with applications to multi-task learning. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, 2021.

[39] Mike Lewis, Shruti Bhosale, Tim Dettmers, Naman Goyal, and Luke Zettlemoyer. BASE layers: Simplifying training of large, sparse models. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, Proceedings of Machine Learning Research. PMLR, 2021.

[40] Stephen Roller, Sainbayar Sukhbaatar, Arthur Szlam, and Jason Weston. Hash layers for large sparse models. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, 2021.

[41] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey E. Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, Proceedings of Machine Learning Research. PMLR, 2020.

[42] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[43] Lu Yuan, Dongdong Chen, Yi-Ling Chen, Noel Codella, Xiyang Dai, Jianfeng Gao, Houdong Hu, Xuedong Huang, Boxin Li, Chunyuan Li, Ce Liu, Mengchen Liu, Zicheng Liu, Yumao Lu, Yu Shi, Lijuan Wang, Jianfeng Wang, Bin Xiao, Zhen Xiao, Jianwei Yang, Michael Zeng, Luowei Zhou, and Pengchuan Zhang. Florence: A new foundation model for computer vision. *CoRR*, 2021.

[44] Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri S. Chatterji, Annie S. Chen, Kathleen Creel, Jared Quincy Davis, Dorottya Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah D. Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark S. Krass, Ranjay Krishna, Rohith Kuditipudi, and et al. On the opportunities and risks of foundation models. *CoRR*, 2021.

[45] Abeba Birhane, Vinay Uday Prabhu, and Emmanuel Kahembwe. Multimodal datasets: misogyny, pornography, and malignant stereotypes. *CoRR*, 2021.

[46] Sara Hooker, Nyalleng Moorosi, Gregory Clark, Samy Bengio, and Emily L. Denton. Characterising bias in compressed models. *ArXiv*, abs/2010.03058, 2020.

[47] David A. Patterson, Joseph Gonzalez, Quoc V. Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David R. So, Maud Texier, and Jeff Dean. Carbon emissions and large neural network training. *CoRR*, 2021.

[48] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 2020.

[49] Christoph Schuhmann, Richard Vencu, Romain Beaumont, Robert Kaczmarczyk, Clayton Mullis, Aarush Katta, Theo Coombes, Jenia Jitsev, and Aran Komatsuzaki. LAION-400M: open dataset of clip-filtered 400 million image-text pairs. *CoRR*, abs/2111.02114, 2021.

[50] Steven Bird, Edward Loper, and Ewan Klein. NLTK: the natural language toolkit. In *ACL 2006, 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*. The Association for Computer Linguistics, 2006.

A Training details

All models were trained with adafactor, using the same modifications used for ViT-G [17]. Unless otherwise specified, we use learning rate 1×10^{-3} and decoupled weight decay of magnitude 1×10^{-5}. We use a cosine learning rate decay schedule, with a linear warmup (40k steps for longer scaling study models, 10k steps for ablations). Models were trained on a mixture of Cloud TPU-v2, v3 and v4 pods.

Models were trained with 32 experts, with experts placed every 2 layers – except where explicitly stated. Otherwise, architecture parameters (e.g. hidden size, number of layers) follow those of ViT [13]. All models except for `LIMoE`-H/14 use dimensionality 512 for the final output representation; this final representation is cast to bfloat16 precision for reduced all-to-all costs and increased memory efficiency. The learned contrastive temperature parameter is initialised at 10. Text sequences are tokenized to a sequence length of 16 using the T5 SentencePiece vocabulary [48]. Images were linearly renormalized to a value range of `[-1, 1]`.

A.1 Scaling study

We train models at batch size 16,384 for 781,250 steps at resolution 224. This trains for the same number of examples as CLIP [7]; they however use a larger batch size (32768), increase resolution in the final epoch, and use a larger dimensionality for the final contrastive feature representation, all of which improve performance.

A.2 Ablations

These are B/16 models trained for 100,000 steps at batch size 8192. The threshold used for the text global entropy loss is $\tau_T = \log(9)$ – that is, we incentivize the use of at least 9 experts (uniformly) or more (not necessarily in a uniform way). For images, $\tau_T = \log(20)$, but with this threshold, the loss is not applied at all and it can be ignored.

A.3 `LIMoE`-H/14

The largest scale model is trained at batch size 21502, with resolution 288 and text sequence length 16. The global entropy loss thresholds are $\tau_{\text{text}} = \log(4)$ and $\tau_{\text{text}} = \log(25)$ for text and image respectively. There are MoE layers in 12 encoder blocks, namely, in 3, 7, 11, 15, 18, 21, 24, 26, 28, 30, 31, 32. The default training data is mixed with data from JFT-4B with a ratio of 3:1. Text strings are generated from JFT-4B by simply concatenating the class names. JFT-4B was also deduplicated using the same method as previous works [16].

Checkpoint souping. We adapt the methodology developed for finetuning [21], but instead combine checkpoints from the same run. We used a reverse-sqrt schedule [48], which has a linear cooldown at the end. To generate diversity for the model soup, we launched multiple cooldowns, and greedily selected checkpoints to maximize zero-shot accuracy on the ImageNet validation set, using the smaller subset of prompts from CLIP [7]. Checkpoints could be reused multiple times.

The model was trained for 700k steps pre-cooldown. There was one cooldown of length 125k steps from the final step, and 3 of length 40k steps starting from step 650k. Two of the cooldowns had no changes to the original setup described above. To generate diversity for the soup, we also trained one 40k cooldown with only JFT data, and one with no JFT data at all.

Figure 7 shows the zero-shot accuracy evaluated at 12.5k step intervals during training, for all the different cooldowns, and the end of training. The final model soup consisted of 8 checkpoints in total.

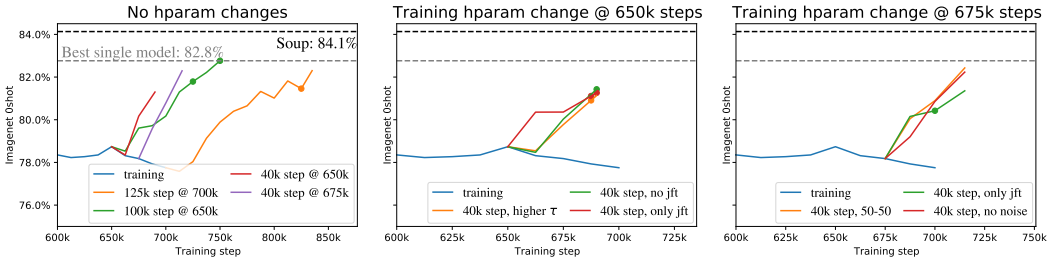


Figure 7: **Souping procedure for** `LIMoE`**-H/14**. Dots show checkpoints in the final model soup.

B Auxiliary losses

B.1 Definitions of all the auxiliary losses

In Section 4.1, we study multiple combinations of auxiliary losses. For completeness, we recall below all their definitions. Given a token $\mathbf{x} \in \mathbb{R}^D$, we denote by $g(\mathbf{x}) = \texttt{softmax}(\mathbf{W}\mathbf{x}) \in \mathbb{R}^E$ the gating weights across the E experts, with $\mathbf{W} \in \mathbb{R}^{E \times D}$ being the routing parameters. When we deal with a batch of multiple tokens $\{\mathbf{x}_i\}_{i=1}^n$, we use the notation $\mathbf{X} \in \mathbb{R}^{n \times D}$.

Importance loss. We consider the definition from [1], inspired by the original proposal of [14]. The importance loss Ω_{imp} enforces a balanced profile of the gating weights across the experts. More formally, for any expert $e \in \{1, \dots, E\}$, we consider

$$\text{imp}_e(\mathbf{X}) = \sum_{\mathbf{x} \in \mathbf{X}} g(\mathbf{x})_e$$

and define the loss Ω_{imp} via the squared coefficient of variation for $\text{imp}(\mathbf{X}) = \{\text{imp}_e(\mathbf{X})\}_{e=1}^E$, namely

$$\Omega_{\text{imp}}(\mathbf{X}) = \left(\frac{\texttt{std}(\text{imp}(\mathbf{X}))}{\texttt{mean}(\text{imp}(\mathbf{X}))} \right)^2 .$$

Load loss. Like previously, we follow [1] whose definition is inspired by the original proposal of [14]. We assume throughout that paragraph that the gating weights $g_{\text{noisy}}(\mathbf{x})$ are obtained by a noisy version of the routing, i.e., $g_{\text{noisy}}(\mathbf{x}) = \texttt{softmax}(\mathbf{W}\mathbf{x} + \varepsilon)$ with $\varepsilon \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})$ and $\sigma = 1/E$ (see details in [1]). We introduce η_K the K-th largest entry of $\mathbf{W}\mathbf{x} + \varepsilon$.

The load loss Ω_{load} complements the importance loss Ω_{imp} by trying to balance the *number of assignments* across the experts. To circumvent the fact that the assignments are discrete, Ω_{imp} focuses instead on the probability of selecting the expert. For any $e \in \{1, \dots, E\}$, the probability is understood as the probability of having the expert e still being among the Top-K while resampling only the noise of that expert. More formally, this corresponds to

$$p_e(\mathbf{x}) = 1 - \Phi\Big(\frac{\eta_K - (\mathbf{W}\mathbf{x})_e}{\sigma} \Big)$$

with Φ the cumulative distribution function of a Gaussian distribution.

The load loss Ω_{load} is eventually defined by

$$\Omega_{\text{load}}(\mathbf{X}) = \left(\frac{\texttt{std}(\text{load}(\mathbf{X}))}{\texttt{mean}(\text{load}(\mathbf{X}))} \right)^2 \text{ with } \text{load}(\mathbf{X}) = \{\text{load}_e(\mathbf{X})\}_{e=1}^E \text{ and } \text{load}_e(\mathbf{X}) = \sum_{\mathbf{x} \in \mathbf{X}} p_e(\mathbf{x}).$$

Z-loss. The z-loss Ω_{zloss} introduced in [4] aims at controlling the maximum magnitude of the router activations $\mathbf{A} = \{\mathbf{W}\mathbf{x}_i\}_{i=1}^n \in \mathbb{R}^{n \times E}$ with entries $a_{i,e} = (\mathbf{W}\mathbf{x}_i)_e$. The loss is defined by

$$\Omega_{\text{zloss}}(\mathbf{X}) = \frac{1}{n} \sum_{i=1}^n \left(\log \left(\sum_{e=1}^E \exp\left(a_{i,e}\right) \right) \right)^2 .$$

The mutual-information loss $-\text{MI}(\texttt{experts}; m)$. In Section 2.2.2, we allude to a variant of the local and global entropy losses in the form of the mutual information between the experts and the modalities (as a reminder, the sum of the local and global entropy losses corresponds instead to the (negative) mutual information between the experts and tokens, conditioned on the modality). Let us assume we have a total of M modalities. Formally, and reusing the notation from Section 2.2.2, we define $-\text{MI}(\texttt{experts}; m)$ as

$$-\text{MI}(\texttt{experts}; m) = \frac{1}{M} \sum_{m'=1}^M \mathcal{H}(\tilde{p}_{m'}(\texttt{experts})) - \mathcal{H}\left(\frac{1}{M} \sum_{m'=1}^M \tilde{p}_{m'}(\texttt{experts}) \right)$$

where, for each modality m', we have computed the approximate marginal probability over the $n_{m'}$ tokens of that modality

$$\tilde{p}_{m'}(\texttt{experts}) = \frac{1}{n_{m'}} \sum_{i=1}^{n_{m'}} p_{m'}(\texttt{experts} | \mathbf{x}_i)$$

and $\mathcal{H}$ denotes the entropy.

Final aggregated auxiliary loss. When considering the combination of several auxiliary losses, the final auxiliary loss is computed as the average over all the losses. The average is weighted by a single regularization parameter that is a hyperparameter of our approach. After some preliminary tuning phase, we have set its value to 0.04 in all our experiments and found this choice to be robust.

B.2 In-depth analysis of global entropy threshold

Note again that we can view a threshold τ as a soft minimum, as the minimum number of experts which must be used by a modality to satisfy the loss is $S = e^{\tau}$. We find it more intuitive to think in terms of this soft minimum threshold S_T.

Performance. Figure 8 shows the effect of the threshold on performance.

There are three phenomenon of note:

1. When the text threshold is too low, models are unstable and performance is poor.
2. Past some limit however, performance of models w.r.t. text threshold is fairly consistent.
3. Outside (and probably inside) the unstable region, the image threshold makes no systematic difference.

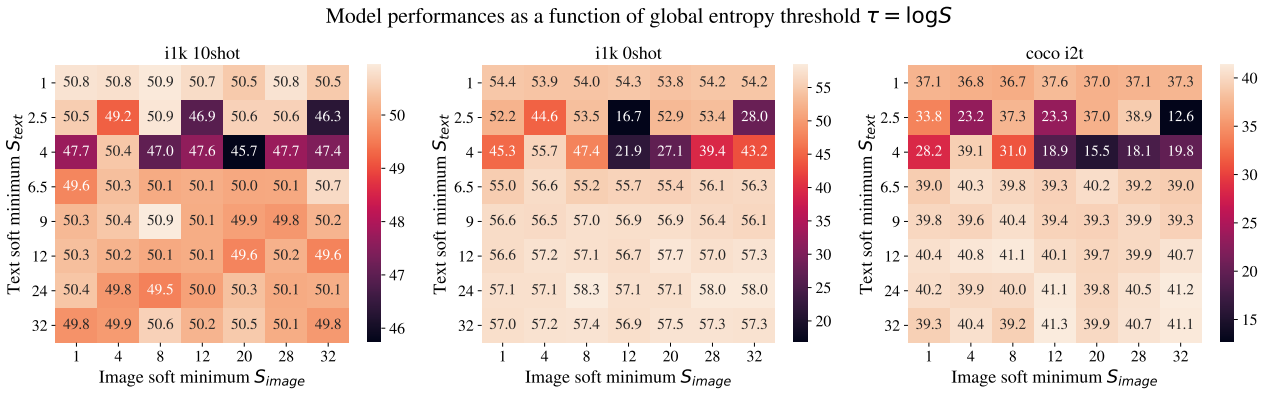


Figure 8: **A high enough text threshold encourages stability**, but otherwise performance is somewhat invariant to the thresholds used. Note the plotted quantity is the *soft minimum* $S_T = e^{\tau}$.

Actual global entropies. Looking at the actual entropies of model routing helps at least explain why the image threshold is unimportant. Figure 9 shows the empirical entropy. The image entropy is always large; note that when it is higher than the threshold τ, the loss is not applied; ergo, for most of the settings, the global entropy loss is *not applied to images*. This also applies to almost all models trained for this paper. On the other hand, analysing text entropies, it is clear that the model closely tracks the threshold τ_{text}. As a side effect, image entropy tends to reduce as τ_{text} increases.

Expert specialization. As discussed, the threshold can be viewed as setting an implicit soft minimum $S_T = e^{\tau}$. The number of experts actually used for each modality is shown in Figure 10. The text threshold exactly behaves as a soft minimum; as it is increased, the model has more text experts and less image experts.

Overall. For text, the entropy loss behaves as expected; as it is increased, there are more text experts. A few questions remain: why does it not impact performance? Why does text behave differently than images - is it due to the imbalance between them during training, or is it simply a fundamental difference in routing behavior for the two modalities?

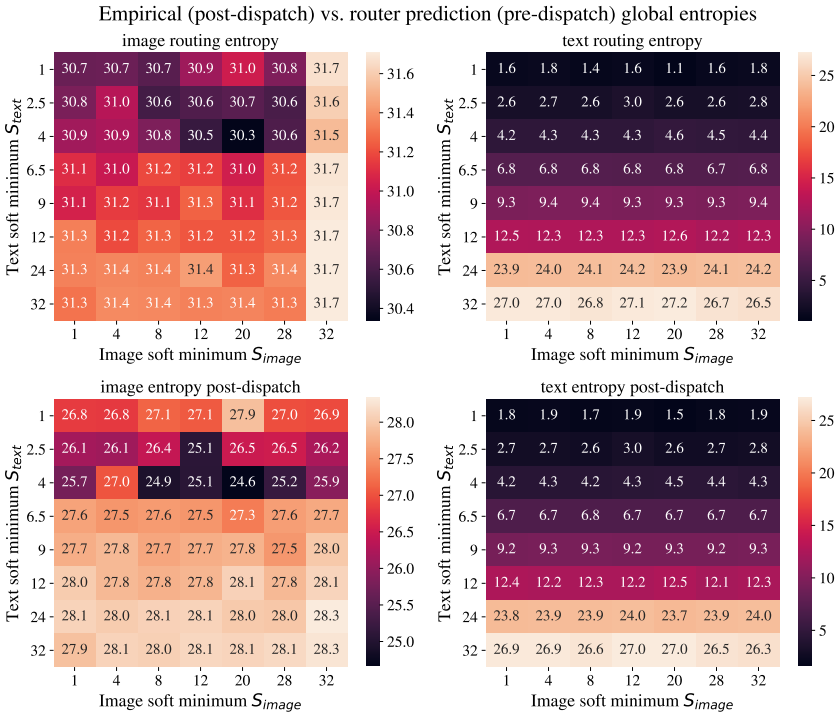


Figure 9: **Image entropy is always high, but text entropy closely tracks the target threshold**. *Top*: The routing entropy is the global entropy of the predictions of the router, which is what is actually regularised. *Bottom*: The post-dispatch entropy is the entropy of the distribution after top-K selection and capacity limits (token dropping) have interfered. For text tokens pre- and post-dispatch entropies pretty much coincide as their routing probabilities are high and BPR favors them –so little dropping happens. The story is a bit different for image tokens; some are dropped, and the pre- and post-dispatch entropies are not completely equivalent.

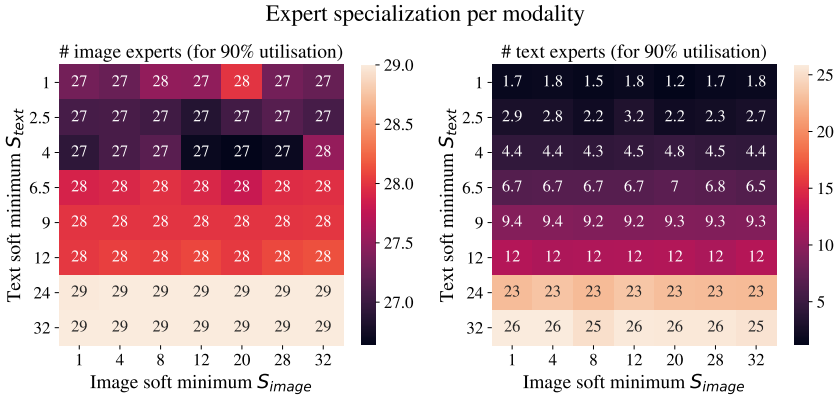


Figure 10: **Almost all experts process images, but the number of text experts closely follows the threshold**. Some experts process both modalities, so it is non trivial to classify an expert as being of a given modality. Our proxy for the number of modality specific experts is the number of experts needed to reach some routing success rate: if only 3 experts are needed for 90% text success rate, then text routing is not highly distributed.

C Tabular results

C.1 Scaling comparison, and architecture definitions.

All results and parameters from Figure 4 are shown in Table 4, alongside the results of LIMoE-H/14.

Table 4: The results from Figure 4 and LIMoE-H/14.

Model	Patch size	Layers	Heads	Hidden size	MLP size	i1k 0shot	i1k 10shot	coco t2i	coco i2t
Dense S/32	32	8	8	512	2048	44.4	33.8	18.2	30.4
LIMoE S/32						57.9	48.1	25.0	38.9
Dense S/16	16	8	8	512	2048	50.3	40.4	22.7	35.8
LIMoE S/16						64.5	56.3	29.2	43.7
Dense B/32	32	12	12	768	3072	58.8	48.7	27.4	42.5
LIMoE B/32						67.5	60.4	31.0	45.7
Dense B/16	16	12	12	768	3072	64.3	55.3	31.7	46.8
LIMoE B/16						73.7	68.2	36.2	51.3
Dense L/32	32	24	16	1024	4096	68.1	60.7	34.6	51.2
LIMoE L/32						74.6	69.7	37.2	54.5
Dense L/16	16	24	16	1024	4096	73.5	67.6	38.3	54.3
LIMoE L/16						78.6	74.7	39.6	55.7
LIMoE-H/14	14	32	16	1280	5120	84.1	81.4	39.8	51.1

C.2 All tabular results

Table 5: All models trained.

model type	arch	notes	τ_{text}	τ_{image}	batch train	batch eval	data seen	0shot	10shot	coco i2t	coco t2i	val acc
Figure 4: Sweep over scale with CLIP-esque training regime												
dense	S/32		-	-	16384	1024	12.8B	44.4	33.8	30.4	18.2	62.1
LIMoE	S/32		$\log(12)$	$\log(17)$	16384	1024	12.8B	57.9	48.1	38.9	25.0	73.1
dense	S/16		-	-	16384	1024	12.8B	50.3	40.4	35.8	22.7	67.6
LIMoE	S/16		$\log(9)$	$\log(20)$	16384	1024	12.8B	64.5	56.3	43.7	29.2	77.1
dense	B/32		-	-	16384	1024	12.8B	58.8	48.7	42.5	27.4	72.5
LIMoE	B/32		$\log(12)$	$\log(17)$	16384	1024	12.8B	67.5	60.4	45.7	31.0	79.2
dense	B/16		-	-	16384	1024	12.8B	64.3	55.3	46.8	31.7	76.4
LIMoE	B/16		$\log(9)$	$\log(20)$	16384	1024	12.8B	73.7	68.2	51.3	36.2	82.3
dense	L/32		-	-	16384	1024	12.8B	68.1	60.7	51.2	34.6	78.5
LIMoE	L/32		$\log(20)$	$\log(1)$	16384	1024	12.8B	74.6	69.7	54.5	37.2	83.3
dense	L/16		-	-	16384	1024	12.8B	73.5	67.6	54.3	38.3	82.2
LIMoE	L/16		$\log(28)$	$\log(8)$	16384	1024	12.8B	78.6	74.7	55.7	39.6	85.9
Table 2: The baselines for many of the ablation experiments below (1T = 1 Tower, 2T = 2 Towers)												
dense (1T)	B/16	Trial 0	-	-	8192	1024	819.2M	49.9	43.7	37.7	23.7	66.0
dense (1T)	B/16	Trial 1	-	-	8192	1024	819.2M	50.0	44.0	36.6	23.8	66.0
dense (1T)	B/16	Trial 2	-	-	8192	1024	819.2M	49.5	43.6	36.0	23.6	66.0
dense (2T)	B/16	Trial 0	-	-	8192	1024	819.2M	54.8	47.3	41.3	26.6	69.7
dense (2T)	B/16	Trial 1	-	-	8192	1024	819.2M	54.4	47.0	41.0	26.5	69.4
dense (2T)	B/16	Trial 2	-	-	8192	1024	819.2M	54.9	47.1	41.6	26.9	69.5
LIMoE	B/16	Trial 0	$\log(9)$	$\log(20)$	8192	1024	819.2M	56.8	50.5	40.1	25.7	70.8
LIMoE	B/16	Trial 1	$\log(9)$	$\log(20)$	8192	1024	819.2M	57.0	50.4	40.4	26.2	70.8
LIMoE	B/16	Trial 2	$\log(9)$	$\log(20)$	8192	1024	819.2M	56.9	50.6	38.5	24.9	70.7
Table 6: Increasing the number of selected experts, with adjustments to local entropy loss and BPR												
LIMoE	B/16	k=2, target entropy loss, max BPR	$\log(4)$	$\log(25)$	8192	1024	819.2M	55.9	48.1	36.6	25.6	69.1
LIMoE	B/16	k=3, target entropy loss, max BPR	$\log(4)$	$\log(25)$	8192	1024	819.2M	48.2	48.9	27.7	21.1	64.3
LIMoE	B/16	k=5, target entropy loss, max BPR	$\log(4)$	$\log(25)$	8192	1024	819.2M	11.7	36.4	7.1	5.6	23.2
LIMoE	B/16	k=2, merged loss, max BPR	$\log(4)$	$\log(25)$	8192	1024	819.2M	46.4	49.4	28.1	10.7	57.3
LIMoE	B/16	k=3, merged loss, max BPR	$\log(4)$	$\log(25)$	8192	1024	819.2M	52.6	47.9	33.0	23.2	65.5
LIMoE	B/16	k=5, merged loss, max BPR	$\log(4)$	$\log(25)$	8192	1024	819.2M	60.3	53.4	43.3	28.0	73.3
LIMoE	B/16	k=2, top1 loss, max BPR	$\log(4)$	$\log(25)$	8192	1024	819.2M	58.3	51.9	42.0	27.2	71.7
LIMoE	B/16	k=3, top1 loss, max BPR	$\log(4)$	$\log(25)$	8192	1024	819.2M	59.0	53.6	42.7	28.1	72.1
LIMoE	B/16	k=5, top1 loss, max BPR	$\log(4)$	$\log(25)$	8192	1024	819.2M	59.8	54.6	43.0	27.8	72.5
LIMoE	B/16	k=2, none loss, max BPR	$\log(4)$	$\log(25)$	8192	1024	819.2M	46.8	44.3	28.9	14.3	61.0
LIMoE	B/16	k=3, none loss, max BPR	$\log(4)$	$\log(25)$	8192	1024	819.2M	44.6	42.2	27.3	17.5	57.5
LIMoE	B/16	k=5, none loss, max BPR	$\log(4)$	$\log(25)$	8192	1024	819.2M	17.5	35.4	6.1	5.9	22.8
LIMoE	B/16	k=2, target entropy loss, sum BPR	$\log(4)$	$\log(25)$	8192	1024	819.2M	58.2	51.8	42.2	27.7	71.6
LIMoE	B/16	k=3, target entropy loss, sum BPR	$\log(4)$	$\log(25)$	8192	1024	819.2M	59.1	53.2	42.3	27.5	72.5
LIMoE	B/16	k=5, target entropy loss, sum BPR	$\log(4)$	$\log(25)$	8192	1024	819.2M	60.4	53.8	42.1	28.0	73.0
LIMoE	B/16	k=2, merged loss, sum BPR	$\log(4)$	$\log(25)$	8192	1024	819.2M	59.0	52.4	41.1	27.1	72.2
LIMoE	B/16	k=3, merged loss, sum BPR	$\log(4)$	$\log(25)$	8192	1024	819.2M	60.0	52.8	42.4	27.6	73.0
LIMoE	B/16	k=5, merged loss, sum BPR	$\log(4)$	$\log(25)$	8192	1024	819.2M	61.0	53.6	42.7	28.4	73.4
Figure 12: Increasing the total number of available experts with fixed $k = 1$												
LIMoE	B/16	Total # experts = 4	$\log(2.4)$	$\log(0.8)$	8192	1024	819.2M	52.3	46.9	37.8	24.4	67.9
LIMoE	B/16	Total # experts = 8	$\log(4.8)$	$\log(1.6)$	8192	1024	819.2M	54.4	48.2	39.5	25.5	69.4
LIMoE	B/16	Total # experts = 16	$\log(9.6)$	$\log(3.2)$	8192	1024	819.2M	55.7	49.5	38.9	25.5	70.2
LIMoE	B/16	Total # experts = 32	$\log(19.2)$	$\log(6.4)$	8192	1024	819.2M	57.3	50.4	40.1	26.0	70.9
LIMoE	B/16	Total # experts = 64	$\log(38.4)$	$\log(12.8)$	8192	1024	819.2M	58.0	50.7	41.2	26.5	71.3
Figure 16: Varying the group size, trading off compute efficiency and stability												
LIMoE	B/16	Num groups = 1	$\log(9)$	$\log(20)$	8192	1024	819.2M	56.7	49.4	40.3	25.8	70.7
LIMoE	B/16	Num groups = 2	$\log(9)$	$\log(20)$	8192	1024	819.2M	56.6	50.4	40.4	25.4	70.8
LIMoE	B/16	Num groups = 4	$\log(9)$	$\log(20)$	8192	1024	819.2M	56.1	49.5	39.1	24.8	69.8
LIMoE	B/16	Num groups = 8	$\log(9)$	$\log(20)$	8192	1024	819.2M	47.4	44.0	29.5	19.6	62.4
LIMoE	B/16	Num groups = 16	$\log(9)$	$\log(20)$	8192	1024	819.2M	50.1	45.3	33.4	21.1	65.2
LIMoE	B/16	Num groups = 32	$\log(9)$	$\log(20)$	8192	1024	819.2M	23.8	31.7	11.6	8.3	38.9
LIMoE	B/16	Num groups = 64	$\log(9)$	$\log(20)$	8192	1024	819.2M	1.6	17.8	1.7	0.9	6.2
LIMoE	B/16	Num groups = 128	$\log(9)$	$\log(20)$	8192	1024	819.2M	0.1	38.3	0.0	0.0	0.1
Figure 6: Study different alternatives for routing dispatch ordering												
LIMoE	B/16	Dispatch = shuffle	$\log(9)$	$\log(20)$	8192	1024	819.2M	37.6	38.5	24.2	15.1	56.5
LIMoE	B/16	Dispatch = image first	$\log(9)$	$\log(20)$	8192	1024	819.2M	17.8	21.3	10.1	6.3	42.3
LIMoE	B/16	Dispatch = bpr	$\log(9)$	$\log(20)$	8192	1024	819.2M	56.8	50.5	40.1	25.7	70.8
LIMoE	B/16	Dispatch = bpr	$\log(9)$	$\log(20)$	8192	1024	819.2M	57.0	50.4	40.4	26.2	70.8
LIMoE	B/16	Dispatch = bpr	$\log(9)$	$\log(20)$	8192	1024	819.2M	56.9	50.6	38.5	24.9	70.7
LIMoE	B/16	Dispatch = text first	$\log(4)$	$\log(25)$	8192	1024	819.2M	0.1	1.6	0.0	0.0	3.3
Table 7: Variations on the joint, modality agnostic router used for LIMoE												
LIMoE	B/16	Router = per modality	$\log(4)$	$\log(25)$	8192	1024	819.2M	56.8	50.5	40.1	25.6	70.4
LIMoE	B/16	Router = partitioned	-	-	8192	1024	819.2M	56.8	50.1	39.1	25.1	70.8
Figure 5: With fixed text seq len 16, vary image seq len to study effect of modality balancing.												
LIMoE	B/12	Image seq len 324. Losses: classic	-	-	8192	1024	819.2M	40.8	42.5	23.4	15.6	53.7
LIMoE	B/16	Image seq len 196. Losses: classic	-	-	8192	1024	819.2M	17.5	32.1	11.9	8.9	41.1
LIMoE	B/24	Image seq len 81. Losses: classic	-	-	8192	1024	819.2M	28.9	33.4	12.3	10.5	37.5
LIMoE	B/32	Image seq len 49. Losses: classic	-	-	8192	1024	819.2M	29.9	29.8	12.0	8.8	39.4
LIMoE	B/48	Image seq len 16. Losses: classic	-	-	8192	1024	819.2M	26.8	24.2	13.0	8.2	36.7

Table 5: All models trained.

model type	arch	notes	τ_{text}	τ_{image}	batch train	batch eval	data seen	0shot	10shot	coco i2t	coco t2i	val acc
LIMoE	B/64	Image seq len 9. Losses: classic	-	-	8192	1024	819.2M	24.8	21.2	11.7	7.7	38.3
LIMoE	B/12	Image seq len 324. Losses: entropy	$\log(25)$	$\log(1)$	8192	1024	819.2M	58.1	50.4	40.4	27.0	72.4
LIMoE	B/16	Image seq len 196. Losses: entropy	$\log(25)$	$\log(1)$	8192	1024	819.2M	57.2	50.3	40.4	26.4	71.2
LIMoE	B/24	Image seq len 81. Losses: entropy	$\log(25)$	$\log(1)$	8192	1024	819.2M	54.1	45.4	37.3	24.0	67.5
LIMoE	B/32	Image seq len 49. Losses: entropy	$\log(25)$	$\log(1)$	8192	1024	819.2M	50.5	41.4	35.2	21.4	65.0
LIMoE	B/48	Image seq len 16. Losses: entropy	$\log(25)$	$\log(1)$	8192	1024	819.2M	40.7	31.0	26.4	15.2	55.4
LIMoE	B/64	Image seq len 9. Losses: entropy	$\log(25)$	$\log(1)$	8192	1024	819.2M	31.8	24.6	20.6	10.9	49.9
Appendix B.2: Varying global entropy thresholds τ_{text} and τ_{image} independently												
LIMoE	B/16		$\log(1)$	$\log(1)$	8192	1024	819.2M	54.4	50.8	37.1	24.4	69.1
LIMoE	B/16		$\log(1)$	$\log(4)$	8192	1024	819.2M	53.9	50.8	36.8	24.1	68.9
LIMoE	B/16		$\log(1)$	$\log(8)$	8192	1024	819.2M	54.0	50.9	36.7	24.1	68.8
LIMoE	B/16		$\log(1)$	$\log(12)$	8192	1024	819.2M	54.3	50.7	37.6	23.8	68.6
LIMoE	B/16		$\log(1)$	$\log(20)$	8192	1024	819.2M	53.8	50.5	37.0	24.1	68.5
LIMoE	B/16		$\log(1)$	$\log(28)$	8192	1024	819.2M	54.2	50.8	37.1	24.1	69.1
LIMoE	B/16		$\log(1)$	$\log(32)$	8192	1024	819.2M	54.2	50.5	37.3	24.2	69.1
LIMoE	B/16		$\log(2.5)$	$\log(1)$	8192	1024	819.2M	52.2	50.5	33.8	19.6	67.0
LIMoE	B/16		$\log(2.5)$	$\log(4)$	8192	1024	819.2M	44.6	49.2	23.2	17.3	57.7
LIMoE	B/16		$\log(2.5)$	$\log(8)$	8192	1024	819.2M	53.5	50.9	37.3	23.8	69.6
LIMoE	B/16		$\log(2.5)$	$\log(12)$	8192	1024	819.2M	16.7	46.9	23.3	14.1	49.4
LIMoE	B/16		$\log(2.5)$	$\log(20)$	8192	1024	819.2M	52.9	50.6	37.0	23.9	69.6
LIMoE	B/16		$\log(2.5)$	$\log(28)$	8192	1024	819.2M	53.4	50.6	38.9	24.5	69.7
LIMoE	B/16		$\log(2.5)$	$\log(32)$	8192	1024	819.2M	28.0	46.3	12.6	8.2	48.5
LIMoE	B/16		$\log(4)$	$\log(1)$	8192	1024	819.2M	45.3	47.7	28.2	20.8	64.2
LIMoE	B/16		$\log(4)$	$\log(4)$	8192	1024	819.2M	55.7	50.4	39.1	25.0	70.3
LIMoE	B/16		$\log(4)$	$\log(8)$	8192	1024	819.2M	47.4	47.0	31.0	18.2	63.9
LIMoE	B/16		$\log(4)$	$\log(12)$	8192	1024	819.2M	21.9	47.6	18.9	18.8	55.1
LIMoE	B/16		$\log(4)$	$\log(20)$	8192	1024	819.2M	27.1	45.7	15.5	17.0	51.8
LIMoE	B/16		$\log(4)$	$\log(28)$	8192	1024	819.2M	39.4	47.7	18.1	14.4	49.4
LIMoE	B/16		$\log(4)$	$\log(32)$	8192	1024	819.2M	43.2	47.4	19.8	15.5	54.2
LIMoE	B/16		$\log(6.5)$	$\log(1)$	8192	1024	819.2M	55.0	49.6	39.0	25.4	70.6
LIMoE	B/16		$\log(6.5)$	$\log(4)$	8192	1024	819.2M	56.6	50.3	40.3	25.1	70.7
LIMoE	B/16		$\log(6.5)$	$\log(8)$	8192	1024	819.2M	55.2	50.1	39.8	25.6	70.7
LIMoE	B/16		$\log(6.5)$	$\log(12)$	8192	1024	819.2M	55.7	50.1	39.3	25.2	70.8
LIMoE	B/16		$\log(6.5)$	$\log(20)$	8192	1024	819.2M	55.4	50.0	40.2	25.2	70.7
LIMoE	B/16		$\log(6.5)$	$\log(28)$	8192	1024	819.2M	56.1	50.1	39.2	25.4	70.6
LIMoE	B/16		$\log(6.5)$	$\log(32)$	8192	1024	819.2M	56.3	50.7	39.0	25.3	70.7
LIMoE	B/16		$\log(9)$	$\log(1)$	8192	1024	819.2M	56.6	50.3	39.8	25.6	71.0
LIMoE	B/16		$\log(9)$	$\log(4)$	8192	1024	819.2M	56.5	50.4	39.6	25.7	70.7
LIMoE	B/16		$\log(9)$	$\log(8)$	8192	1024	819.2M	57.0	50.9	40.4	26.4	70.8
LIMoE	B/16		$\log(9)$	$\log(12)$	8192	1024	819.2M	56.9	50.1	39.4	26.0	70.9
LIMoE	B/16		$\log(9)$	$\log(20)$	8192	1024	819.2M	56.9	49.9	39.3	25.6	70.7
LIMoE	B/16		$\log(9)$	$\log(28)$	8192	1024	819.2M	56.4	49.8	39.9	25.9	70.7
LIMoE	B/16		$\log(9)$	$\log(32)$	8192	1024	819.2M	56.1	50.2	39.3	24.8	70.8
LIMoE	B/16		$\log(12)$	$\log(1)$	8192	1024	819.2M	56.6	50.3	40.4	26.2	71.0
LIMoE	B/16		$\log(12)$	$\log(4)$	8192	1024	819.2M	57.2	50.2	40.8	25.8	71.1
LIMoE	B/16		$\log(12)$	$\log(8)$	8192	1024	819.2M	57.1	50.1	41.1	25.5	71.2
LIMoE	B/16		$\log(12)$	$\log(12)$	8192	1024	819.2M	56.7	50.1	40.1	25.2	71.2
LIMoE	B/16		$\log(12)$	$\log(20)$	8192	1024	819.2M	57.7	49.6	39.7	25.3	70.8
LIMoE	B/16		$\log(12)$	$\log(28)$	8192	1024	819.2M	57.0	50.2	39.9	26.0	71.1
LIMoE	B/16		$\log(12)$	$\log(32)$	8192	1024	819.2M	57.3	49.6	40.7	25.9	70.8
LIMoE	B/16		$\log(24)$	$\log(1)$	8192	1024	819.2M	57.1	50.4	40.2	25.8	71.3
LIMoE	B/16		$\log(24)$	$\log(4)$	8192	1024	819.2M	57.1	49.8	39.9	25.8	70.9
LIMoE	B/16		$\log(24)$	$\log(8)$	8192	1024	819.2M	58.3	49.5	40.0	26.1	71.1
LIMoE	B/16		$\log(24)$	$\log(12)$	8192	1024	819.2M	57.1	50.0	41.1	25.9	70.9
LIMoE	B/16		$\log(24)$	$\log(20)$	8192	1024	819.2M	57.1	50.3	39.8	25.4	71.1
LIMoE	B/16		$\log(24)$	$\log(28)$	8192	1024	819.2M	58.0	50.1	40.5	26.0	71.2
LIMoE	B/16		$\log(24)$	$\log(32)$	8192	1024	819.2M	58.0	50.1	41.2	25.7	71.0
LIMoE	B/16		$\log(32)$	$\log(1)$	8192	1024	819.2M	57.0	49.8	39.3	25.3	71.2
LIMoE	B/16		$\log(32)$	$\log(4)$	8192	1024	819.2M	57.2	49.9	40.4	25.9	71.1
LIMoE	B/16		$\log(32)$	$\log(8)$	8192	1024	819.2M	57.4	50.6	39.2	25.8	71.1
LIMoE	B/16		$\log(32)$	$\log(12)$	8192	1024	819.2M	56.9	50.2	41.3	26.4	71.0
LIMoE	B/16		$\log(32)$	$\log(20)$	8192	1024	819.2M	57.5	50.5	39.9	26.0	71.1
LIMoE	B/16		$\log(32)$	$\log(28)$	8192	1024	819.2M	57.3	50.1	40.7	26.0	71.0
LIMoE	B/16		$\log(32)$	$\log(32)$	8192	1024	819.2M	57.3	49.8	41.1	26.3	71.0
Table 8: Training on publically available LAION400M data.												
dense	B/16	Trial 0	-	-	16384	1024	1.4B	56.1	47.9	43.0	27.9	96.6
dense	B/16	Trial 1	-	-	16384	1024	1.4B	56.0	47.7	42.8	27.6	96.5
dense	B/16	Trial 2	-	-	16384	1024	1.4B	55.8	47.5	42.5	27.9	96.6
LIMoE	B/16	Trial 0	$\log(9)$	$\log(20)$	16384	1024	1.4B	61.1	54.4	44.1	28.9	97.9
LIMoE	B/16	Trial 1	$\log(9)$	$\log(20)$	16384	1024	1.4B	60.9	54.4	43.5	28.7	97.9
LIMoE	B/16	Trial 2	$\log(9)$	$\log(20)$	16384	1024	1.4B	61.1	54.1	43.6	29.0	97.9

D Computational costs of LIMoE

D.1 Unimodal evaluation with multimodal experts

Recall that each expert has a capacity C - it can process at most C tokens, and if it is assigned more, those above C will not be processed. This capacity is usually set relative to some 'ideal'. If there are N tokens and E experts, we usually assume each expert can handle at most $C_R \times \frac{N}{E}$ tokens, where $C_R \geq 1$ is a slack factor. This way we try to reach a balanced setup where most expert process a similar number of tokens.

Multimodal routing presents a unique issue here. During training, the model learns to balance tokens when it has both images *and* text available to it. When there is only one modality, it will not use all the experts due to natural emergence of modality-specific experts - but the expert capacity size will be set *assuming all experts are used*. This results in high rates of token dropping, depending on the ratio of modality-specific experts.

In this effort, we encounter this during zero-shot classification and retrieval; models first compute representations for all text tokens, and then separately for all image tokens. In order to get around this token dropping, we simply evaluate with a high slack factor $C_R = 16$.

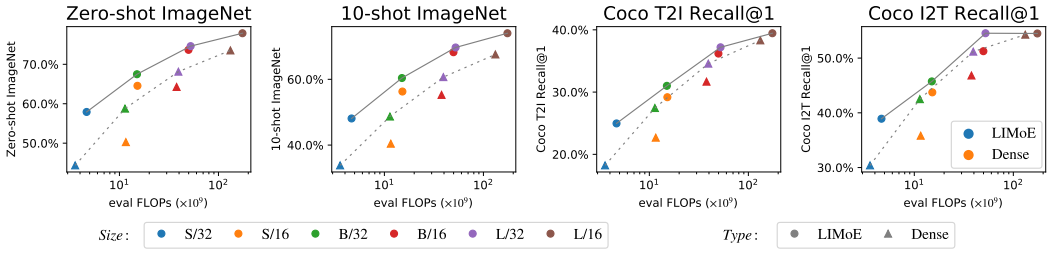
There are however other natural solutions; for many circumstances, one could trivially restructure evaluation such that image and text inputs are processed simultaneously. A more interesting, MoE specific solution is *pruning* modality specific experts, which is explored and shown to work in E.4. LIMoE models could have been evaluated at a 'normal' capacity, with pruned experts.

D.2 Understanding the compute costs of LIMoE

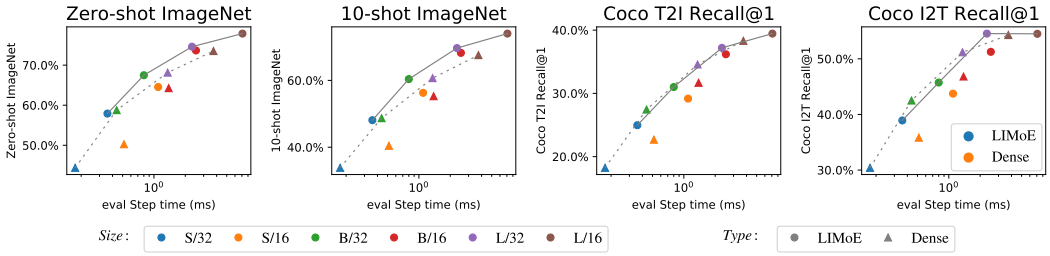
Zero-shot evaluation on ImageNet with 6 prompts requires 6000 text forward passes and 50000 image forward passes. With 80 prompts, a la CLIP [7], it is 80000 for text instead. How does one compare compute cost vs. performance? The costs of LIMoE, its dense baselines, and other two-tower models, were computed assuming a full batch of images and texts, as this is the approach which makes the least assumptions about the downstream setup. This does not generalise perfectly: if, for example, a particular use case processed very large numbers of texts but only few images, models with smaller/cheaper text towers would be clearly advantaged.

Full profiling data for Section 3. For training and evaluation, we used a variety of TPU versions. For consistency, we profiled computation times on a TPUv3 (v3-32 to be more precise[2]). Figure 11 shows performance with respect to different proxies for compute cost. As discussed in Section 3, LIMoE is clearly pareto optimal with respect to total FLOPs. However, this does not fully account for certain costs related to MoE models, such as cross-device communication. Figures 11b and 11d show the performance with respect to step time. With respect to zeroshot and 10-shot classification accuracy, the performance improvements of LIMoE are significant enough that it is still clearly pareto optimal; for retrieval metrics on COCO, LIMoE's gains exactly justify the costs, and it is not significantly more efficient than dense baselines. The story is similar whether looking at train or evaluation cost.

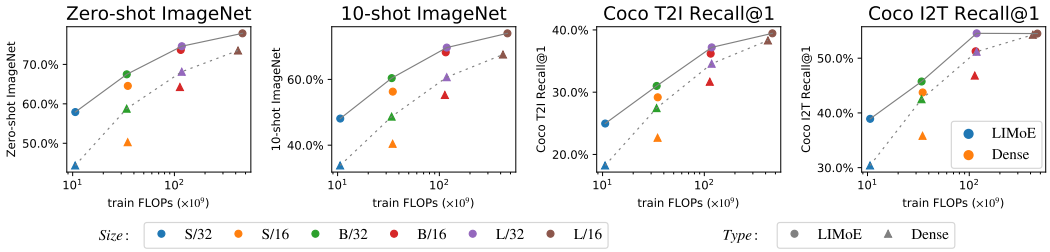
[2] https://cloud.google.com/tpu/docs/types-topologies



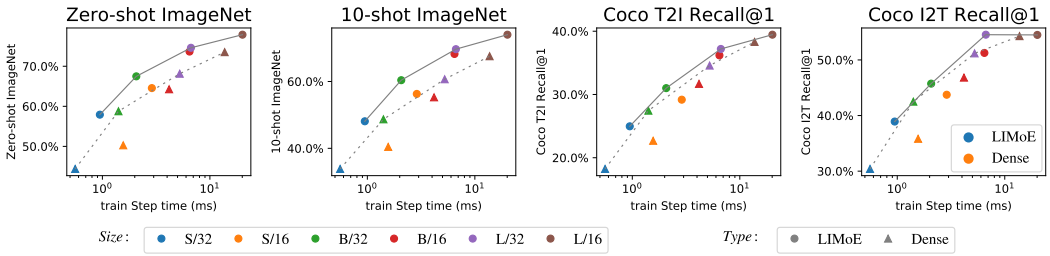
(a) Inference FLOPs



(b) Inference step time



(c) Train FLOPs



(d) Train step time

Figure 11: Pareto frontiers with respect to different measures of computational cost

Table 6: Increasing number of selected experts improves performance with appropriate changes to auxiliary losses and BPR. Table entries are ImageNet zero-shot accuracy in %.

	BPR score = max				BPR score = sum	
Local entropy method	None	Default	Target Ent	Merged	Target Ent	Merged
$K = 1$		$55.5^{57.7}_{53.3}$				
$K = 2$	46.8	58.3	55.9	46.4	58.2	59.0
$K = 3$	44.6	59.0	48.2	52.6	59.1	60.0
$K = 5$	17.5	59.8	11.7	60.3	60.4	61.0

E Further experiments

In this section, we present further ablations not included in the main text due to space constraints.

E.1 Increasing the number of selected experts

All models in this paper select $K = 1$ expert per token to match the cost of a dense backbone.

There are two main challenges with increasing K:

Modifications to auxiliary losses. The local entropy loss effectively encourages that router choices are one-hot. When increasing K, the model is still incentivized to only use 1 expert, assigning other experts weights near 0, thereby effectively behaving as $K = 1$.

We try two modifications to the local loss to ameliorate this:

- *Target entropy*: Encourage the local entropy to be $\log K$ – at least a uniform distribution over K experts – instead of 0: we minimize $\Omega^{\text{K}}_{\text{target}} = (\log(K) - \Omega_{\text{local}}(\mathbf{G}_m))^2$.
- *Merged entropy*: We sum the top K and the bottom $N - K$ routing probabilities to give a binomial distribution, and optimize this to have entropy 0. This encourages the routing weight to all be in the top K experts, but does not care exactly how it is distributed among them.

BPR modifications With these losses, the router uses $K > 1$ experts per token. However, BPR prioritises tokens according to their max routing probability, which decreases when probabilities are distributed over K choices. The stabilisation effect BPR provides training is consequently lost. We alter it to prioritise tokens by the sum of top K probabilities. In vision tasks, the two approaches perform identically [1], but here the latter stabilises training and unlocks $K > 1$.

Table 6 shows the final results. Without changes to the local entropy loss (BPR score = max, local entropy method = default), there are some improvements which stem from using $K > 1$ for image tokens - the local loss on text means it is effectively using $K = 1$ for text anyway. Without the modifications to the BPR score, the modifications to the local loss can result in fairly unstable models. Once the BPR score is modified, we see consistent improvements with increasing K, particularly with the 'Merged' variant.

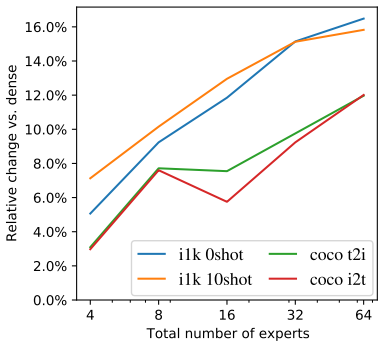


Figure 12: **Increasing the total number of experts** consistently improves model performance; all are better than the dense baseline.

E.2 Increasing the total number of experts

There is thus far no consensus on the optimal number of experts in MoEs; early NLP research scaled to 1000s of experts [14, 8], before reducing to 32 or 64 [4], which is the standard setup for vision [1]. In Figure 12, we vary this for `LIMoE`, and show that larger expert pools yield consistent performance improvements.

Table 7: A simple routing setup without modality-specific adjustments is competitive with specialized approaches. 0shot and 10shot columns show accuracy (%), t2i and i2t show recall@1 (%).

routers	i1k 0shot	i1k 10shot	coco t2i	coco i2t
joint	$56.9^{57.2}_{56.7}$	$50.5^{52.0}_{49.0}$	$25.5^{34.3}_{16.8}$	$39.5^{51.8}_{27.1}$
per modality	56.8	50.5	25.6	40.1
disjoint experts (5 for text, 27 for image)	56.8	50.1	25.1	39.1

E.3 Router design choice

Recall the router is simply a dense layer; by default we have a *joint* router for all tokens, independent of modality, with no constraints on gating. We consider two other options:

- *Per-modality router.* We consider modality-dependent routers which can leverage knowledge of token modality to improve performance (that is, one router for image tokens, and a different one for text tokens). They both output routing distributions over a shared pool of experts, similar to prior works have per-task routers for multitask learning [6].
- *Disjoint experts and routers.* We define separate pools of image and text experts. This way, image tokens can only go to a set of experts $\mathcal{E}_{\text{img}}$, and text tokens can only be assigned to another set of experts $\mathcal{E}_{\text{txt}}$. In principle, these sets may or may not intersect. In Table 7, we report results when the sets are indeed disjoint.

The results in Table 7 show the three approaches lead to comparable performance. In general, the disjoint setup was more stable, and did not need entropy regularisation as per-modality balance/independence is enforced by design. While convenient and well-behaved here, this approach may not be as general for the case with dozens of tasks and modalities.

E.4 Pruning Multimodal Experts

During training we track what fraction of each modality's tokens went to each expert. It is therefore trivial to identify which experts are processing predominately text and which are processing predominately images. We show here that this information can be trivially used to prune experts for single-modality forward passes, demonstrating on two 32-expert `LIMoE-S/16` models: one trained with global text entropy threshold $\tau_{\text{text}} = \log(4)$, and one with $\tau_{\text{text}} = \log(9)$.

Choosing what to prune. Note that we separately choose what experts to prune per-modality; we use text as an illustrative example. Pruning is simple: For each MoE layer, we rank experts according to the fraction of text tokens they processed during training (we average over the last 2500 steps with measurements sampled every 50 steps). We then start pruning according to the one that processed the least tokens, and so on. Figure 13 shows how the coverage of different modalities changes as experts are pruned. Following the relationship between the global text entropy threshold and the idea of the 'soft minimum', we see that around $e^{\tau_{\text{text}}}$ text experts are needed to process the majority of text tokens; e.g. with $\tau_{\text{text}} = 4$ for a single-modality forward pass, 28 experts could be comfortably pruned. Image experts are more distributed; almost all the experts are needed to process all image tokens, as expected.

How to run `LIMoE` inference with fewer experts. While some experts are pruned, the model is not further trained to adapt to this new situation. One must therefore think carefully on the best way to apply models with a subset of experts. The router predicts $p(\texttt{expert}|\mathbf{x})$. The top-$K$ experts are activated, and the output of the expert layer is the weighted average of the expert outputs. The weighting used for expert i is the *unnormalized* $p(\texttt{expert}_i|\mathbf{x})$. This is important, as removing some of the experts and their logits modifies the concentration of $p(\texttt{expert}|\mathbf{x})$, and could result in expert weights higher than those used at training time.

When removing some of the experts, there are therefore two natural options:

1. `router-drop`: Completely remove the experts from the router. The softmax for $p(\texttt{expert}|\mathbf{x})$ will be computed over a subset of experts, thereby adjusting the weights as discussed above.

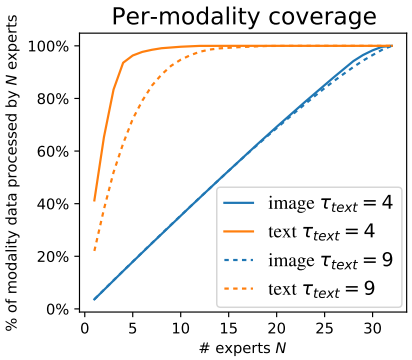


Figure 13: **Per-modality coverage**; for each modality, we progressively prune the least important experts. The coverage shows the percentage of router top-1 predictions which are still serviceable with the remaining experts. For text (orange), many experts can be pruned, but that is not the case for images. The global entropy threshold τ controls the prunability, as it encourages use of at least $\log(\tau)$ experts.

2. `router-pred`: The router still predicts probabilities for pruned experts. However, it is unable to actually use the pruned experts; the top-K operation will ignore those that are unavailable. This preserves the original scaling the model was trained with.

The two approaches are naturally very similar if very few experts are removed. Illustrating with ImageNet-10shot (linear few-shot evaluation), Figure 14 compares the two options. When a large number of experts are pruned, the `router-pred` is significantly better, but if only a few experts are pruned, they both perform similarly.

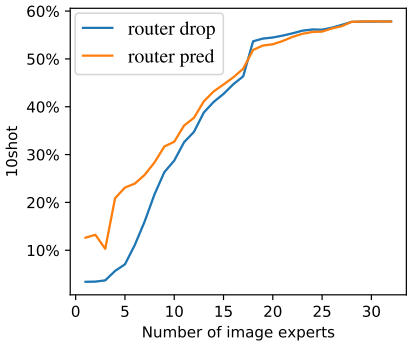


Figure 14: It is better to predict weights for pruned experts and mask them out *after* the softmax.

The effect of pruning on performance Figure 15 shows the impact of pruning image and text experts on zero-shot ImageNet accuracy. Recall that image and text inputs are processed independently for this evaluation, and so the experts used for each modality can be independently pruned.

As expected, we can prune down to only 4 experts during text evaluation without significantly harming performance. On the other hand, the less pruning of image experts, the better.

E.5 Grouped routing

Splitting batches into groups before dispatching can reduce routing cost significantly, which depending on implementation can scale $\sim \mathcal{O}(\texttt{num tokens}^2)$. There are two sources of potential issues though: in our implementation, auxiliary losses are computed in each group then averaged. The necessary batch-wise statistics become less reliable with more numerous, smaller groups. Secondly, with smaller groups, it is more likely to get an almost homogenous batch, which makes distributing across

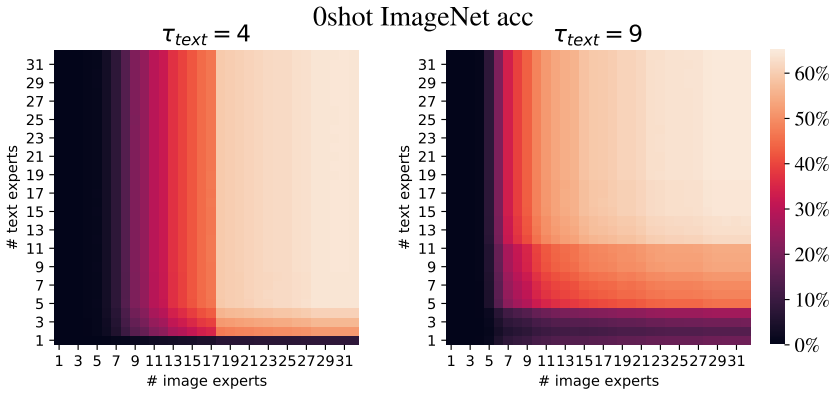


Figure 15: The impact of pruning on ImageNet zero-shot accuracy, comparing two LIMoE-S/16 models trained with different global text entropy thresholds.

Table 8: LIMoE performance on LAION400M, against a dense baseline, three trials.

	10shot	0shot	COCO t2i	COCO i2t
Dense	$47.7^{48.1}_{47.3}$	$56.0^{56.3}_{55.6}$	$27.8^{28.3}_{27.4}$	$42.8^{43.4}_{42.1}$
LIMoE	$54.3^{54.7}_{53.9}$	$61.0^{61.4}_{60.7}$	$28.9^{29.2}_{28.5}$	$43.7^{44.6}_{42.9}$

experts harder. To study this, we sweep the group size in a parallel setup with 128 examples per device. Group size 1 means processing and dispatching $128 \times (196 + 16) = 27136$ tokens at once, whereas e.g. group size 8 involves splitting into 8 groups of 3392 tokens. Figure 16 shows the effect of this; up to 4 groups, performance is good, but any more than that and training becomes unstable, harming performance. This is more fragile than image-only routing, where group sizes as small as 400 are stable (equivalent to ~ 68 groups here). Nonetheless, with 4 groups, step time is reduced by 30%, capturing 75% of potential efficiency gains from grouped routing.

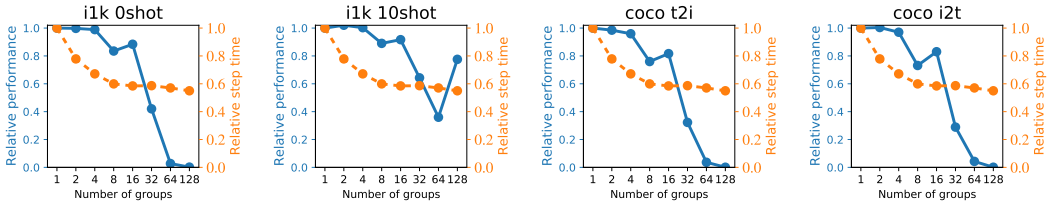


Figure 16: **Grouped routing can reduce step time (orange)**, but too much becomes unstable.

E.6 Experiments on public data

In order to ascertain LIMoE's efficacy on public data, and reproducibility, we train B/16 models on LAION-400M [49]. We train for 5 epochs at batch size 16,384. Table 8 shows the outcome of three trials, compared against a dense baseline. Once again, we see significant improvements performance, especially in ImageNet zero-shot (+5.0% absolute, +8.9% relative) and 10-shot (+6.6% absolute, +13.8% relative) performance.

F Model Analysis

F.1 Routing Distributions

In this section, we explore how routing is distributed across different layers, experts, and modalities. In particular, we focus on which tokens are dropped. We analyze two models, B/32 and B/16, each with 8 experts. This way we can appreciate the impact of having a significantly different ratio of text:image tokens. Moreover, the global entropy targets $S = e^{\tau}$ for (text, image) tokens are (3, 25) and (6, 6) for the B/32 and B/16 models, respectively.

We first show the routing distributions under the training distribution in Figures 17 and 18. In both cases –as expected– routing works very well. Moreover, most experts handle both image and text tokens.

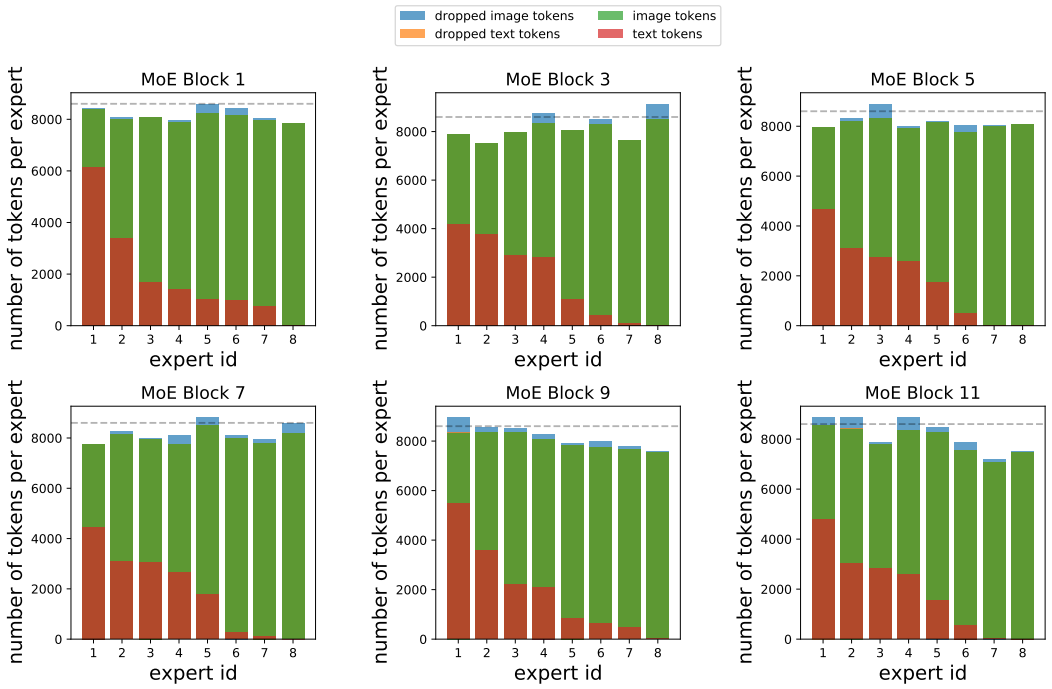


Figure 17: **Token Distribution for training data.** B/32 model with 8 experts. We display utilization and dropping for a forward pass with batch size 1024. The discontinuous line represents the maximum capacity per expert. Note that we enforce capacity locally per device, so some tokens may not be able to be dispatched even within global capacity constraints. We observe very little token dropping as this is the training data for which auxiliary losses lead to balance.

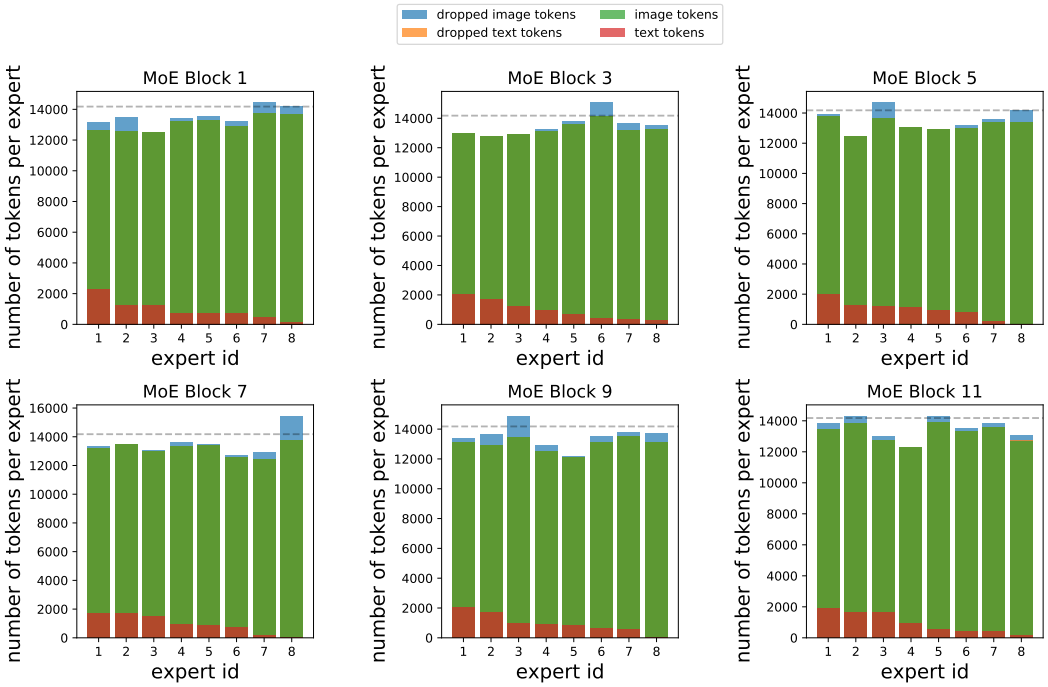


Figure 18: **Token Distribution for training data.** B/16 model with 8 experts. We display utilization and dropping for a forward pass with batch size 512. The discontinuous line represents the maximum capacity per expert. Note that we enforce capacity locally per device, so some tokens may not be able to be dispatched even within global capacity constraints. We observe very little token dropping as this is the training data for which auxiliary losses lead to balance. Compared to Figure 17, we can see how text tokens generally represent a quite small fraction of the in-flow for every expert.

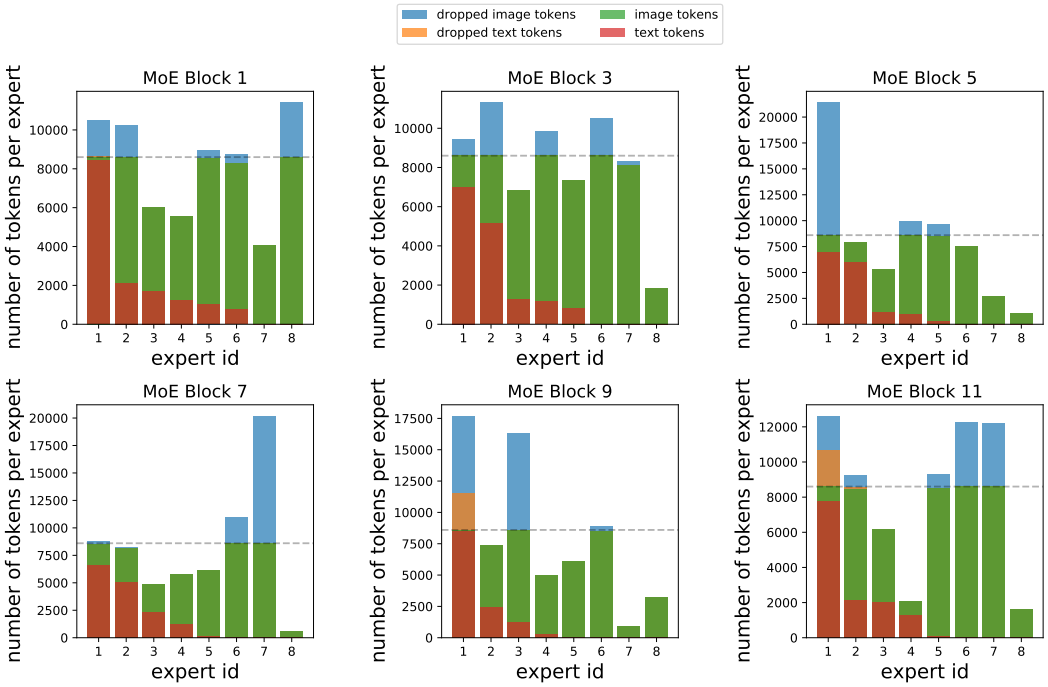


Figure 19: **Token Distribution for COCO data.** B/32 model with 8 experts. We display utilization and dropping for a forward pass with batch size 1024. The discontinuous line represents the maximum capacity per expert. Note that we enforce capacity locally per device, so some tokens may not be able to be dispatched even within global capacity constraints. Compared to Figure 17, in this case, as there is a distribution shift –while no further training or finetuning–, we see distributions of tokens per expert becoming fairly unbalanced. Moreover, a non-trivial amount of tokens are dropped (above discontinuous horizontal line). Even text tokens are dropped sometimes, and some experts –like Expert 1 in the MoE Block 1 or 9– end up only processing text tokens.

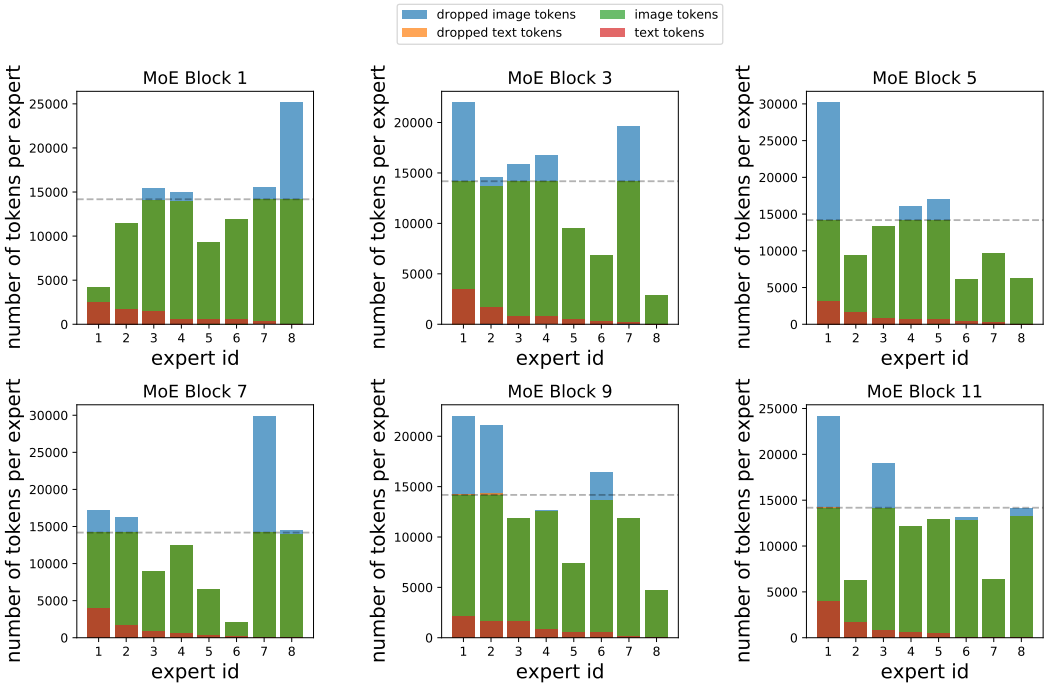


Figure 20: **Token Distribution for COCO data.** B/16 model with 8 experts. We display utilization and dropping for a forward pass with batch size 512. The discontinuous line represents the maximum capacity per expert. Note that we enforce capacity locally per device, so some tokens may not be able to be dispatched even within global capacity constraints. Compared to Figure 18, in this case, as there is a distribution shift –while no further training or finetuning–, we see distributions of tokens per expert becoming fairly unbalanced. Moreover, a non-trivial amount of tokens are dropped (above discontinuous horizontal line). Text tokens are still mostly processed as BPR shields them via their high priorities and they still represent a small percentage of the tokens.

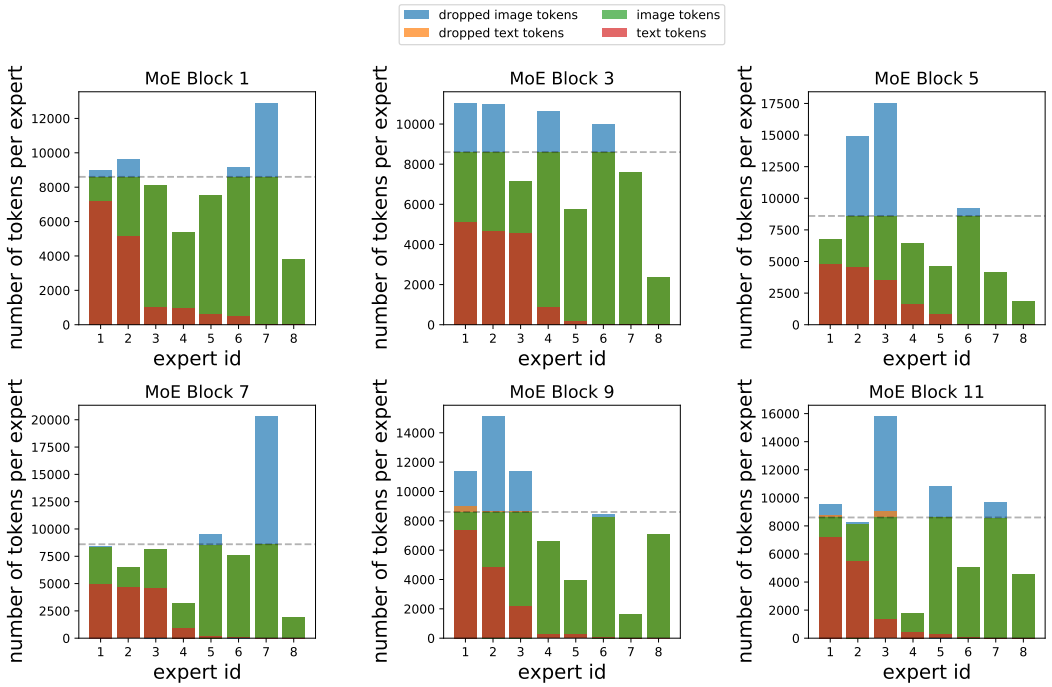


Figure 21: **Token Distribution for ImageNet data.** B/32 model with 8 experts. We display utilization and dropping for a forward pass with batch size 1024. The discontinuous line represents the maximum capacity per expert. Note that we enforce capacity locally per device, so some tokens may not be able to be dispatched even within global capacity constraints. Compared to Figure 17, in this case, as there is a distribution shift –while no further training or finetuning–, we see distributions of tokens per expert becoming fairly unbalanced. Moreover, a non-trivial amount of tokens are dropped (above discontinuous horizontal line). Very few text tokens are dropped (there is a significant amount of padding, and prompt tokens that are probably processed with very high confidence scores by BPR).

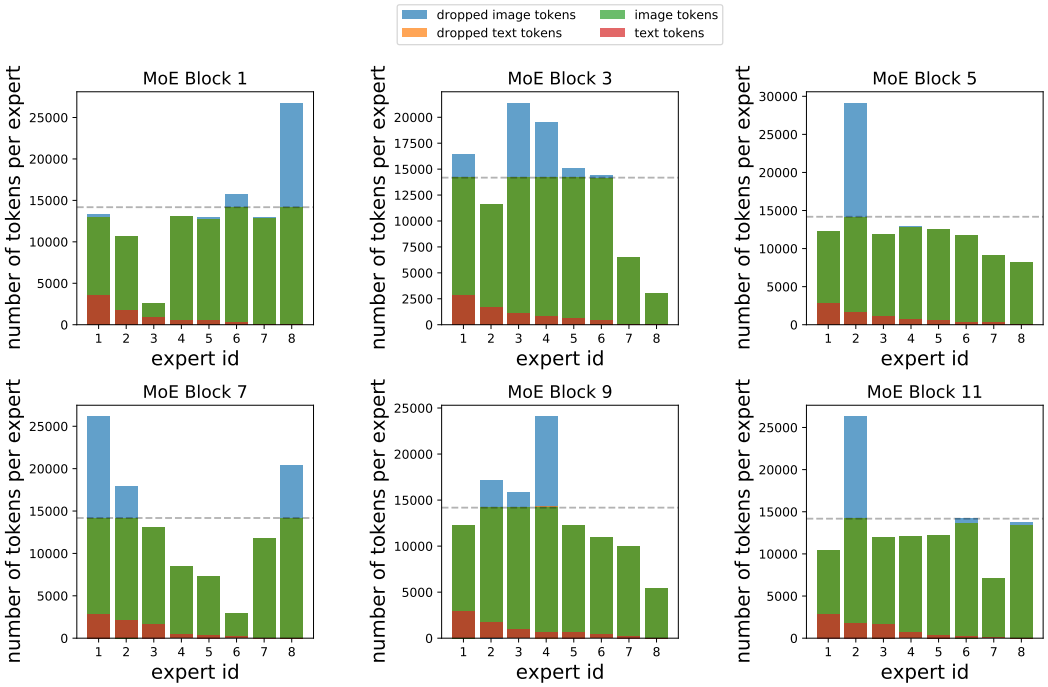


Figure 22: **Token Distribution for ImageNet data.** B/16 model with 8 experts. We display utilization and dropping for a forward pass with batch size 512. The discontinuous line represents the maximum capacity per expert. Note that we enforce capacity locally per device, so some tokens may not be able to be dispatched even within global capacity constraints. Compared to Figure 18, in this case, as there is a distribution shift –while no further training or finetuning–, we see distributions of tokens per expert becoming fairly unbalanced. Moreover, a non-trivial amount of tokens are dropped (above discontinuous horizontal line). Almost no text tokens are dropped (there is a significant amount of padding, and prompt tokens that are probably processed with very high confidence scores by BPR).

F.2 Routing Examples

In this section, we share practical examples of image and text token routing on the B/32 and B/16 models introduced at the beginning of the section. All evaluations are on ImageNet (that is, not on the training data). While the number of experts is clearly smaller than the number of different semantic concepts in images and text, we still highlight some cool patterns in most experts – especially in the context of images, as text tokens tend to use a reduced number of experts. We show some of the patches with the highest routing confidence, as analyzing all the thousands of patches that are assigned to each expert is difficult. However, we expect many other semantic concepts present in the training data to be almost exclusively served by individual experts.

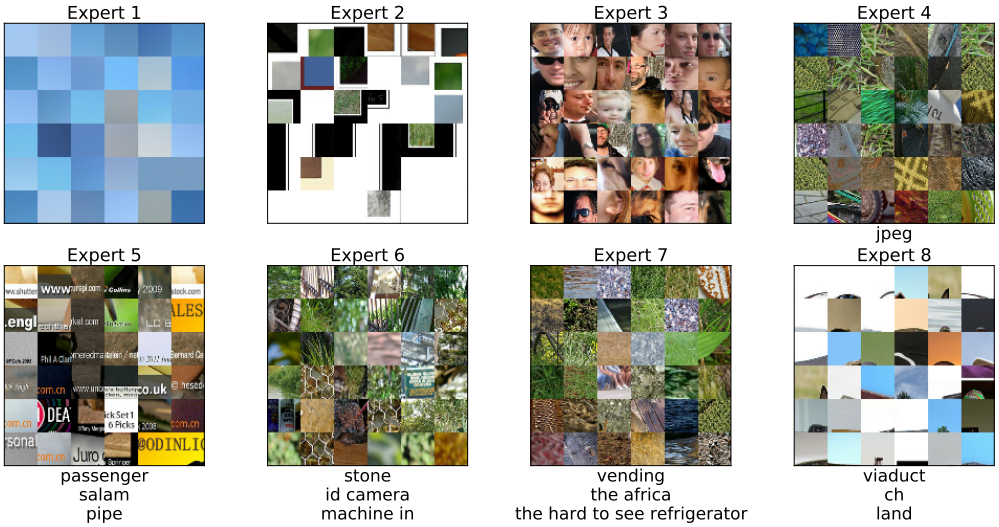


Figure 23: **Token routing for Imagenet.** B/32 model with 8 experts, we show some of the original tokens (both image and text) as routed at the second MoE layer (corresponds to the fourth encoder).

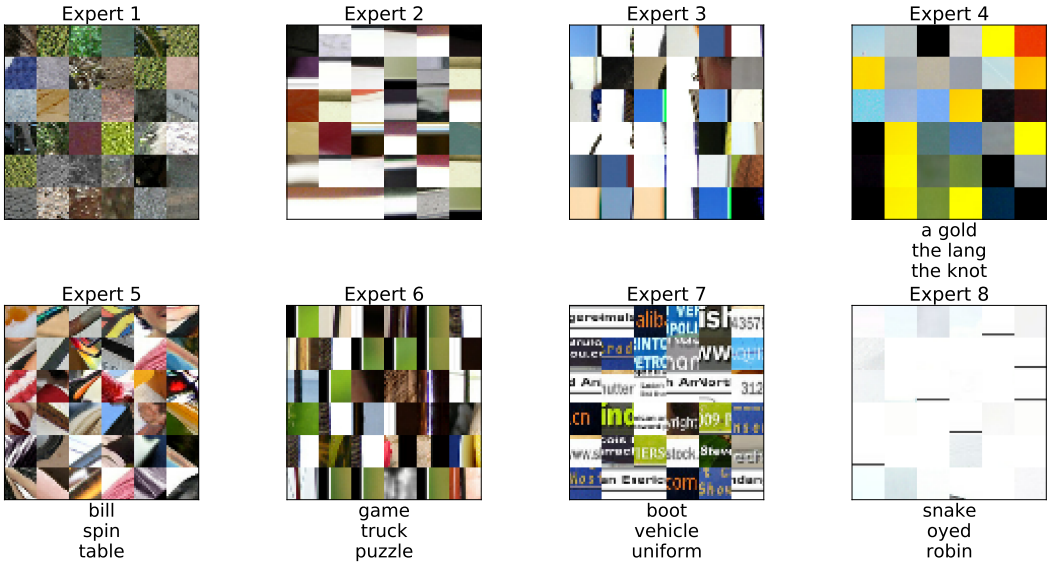


Figure 24: **Token routing for Imagenet.** B/16 model with 8 experts, we show original tokens (both image and text) as routed at the first MoE layer (corresponds to the second encoder block).

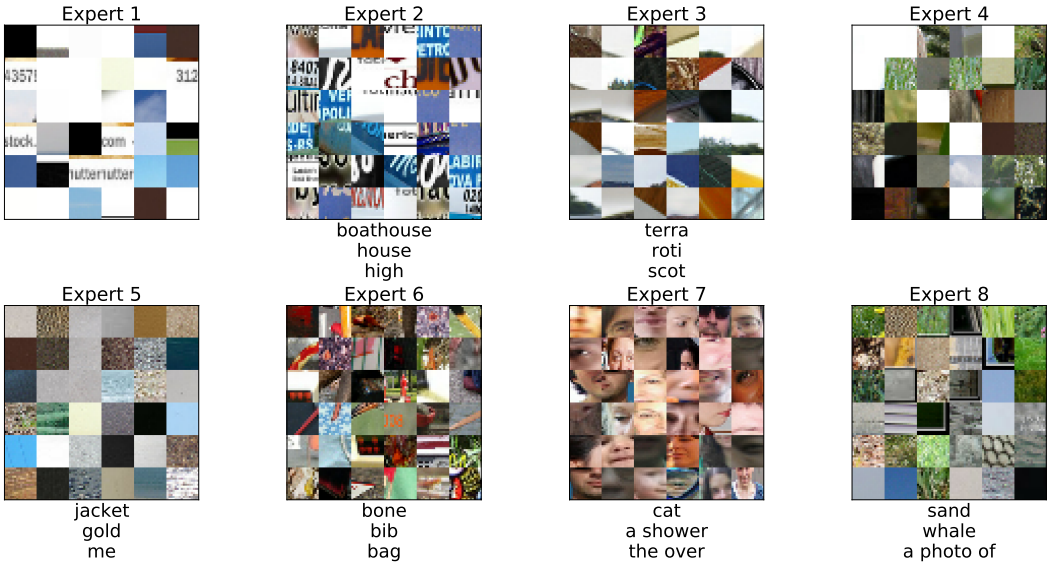


Figure 25: **Token routing for Imagenet.** B/16 model with 8 experts, we show original tokens (both image and text) as routed at the second MoE layer (corresponding to the fourth encoder block).

F.3 Routing for Individual Inputs

In this subsection, we show the expert split for a specific given input – image and text. Recall tokens from different modalities do not interact in the forward pass (other than via sharing expert capacity).

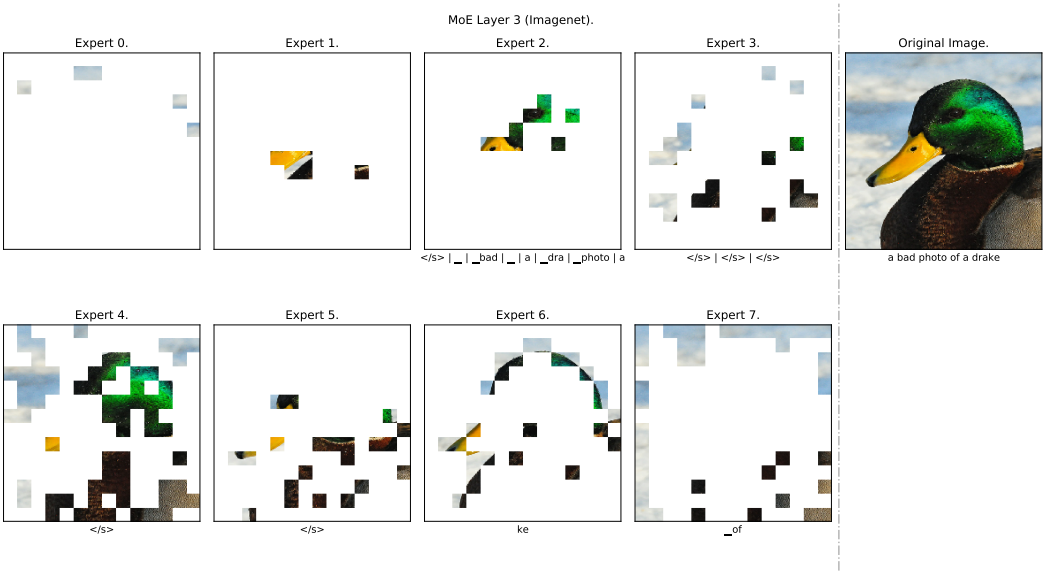


Figure 26: **Token routing for an Imagenet input.** B/16 model with 8 experts, we show original tokens (both image and text) as routed at the second MoE layer (corresponding to the fourth encoder block, while we use zero-indexing). The original image and text are displayed on the right-hand side.

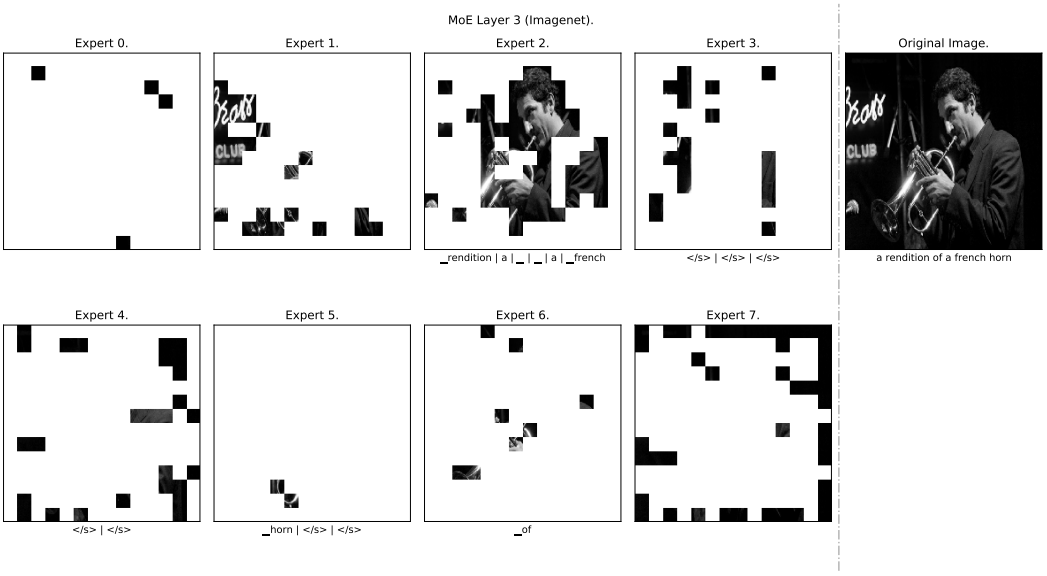


Figure 27: **Token routing for an Imagenet input.** B/16 model with 8 experts, we show original tokens (both image and text) as routed at the second MoE layer (corresponding to the fourth encoder block, while we use zero-indexing). The original image and text are displayed on the right-hand side.

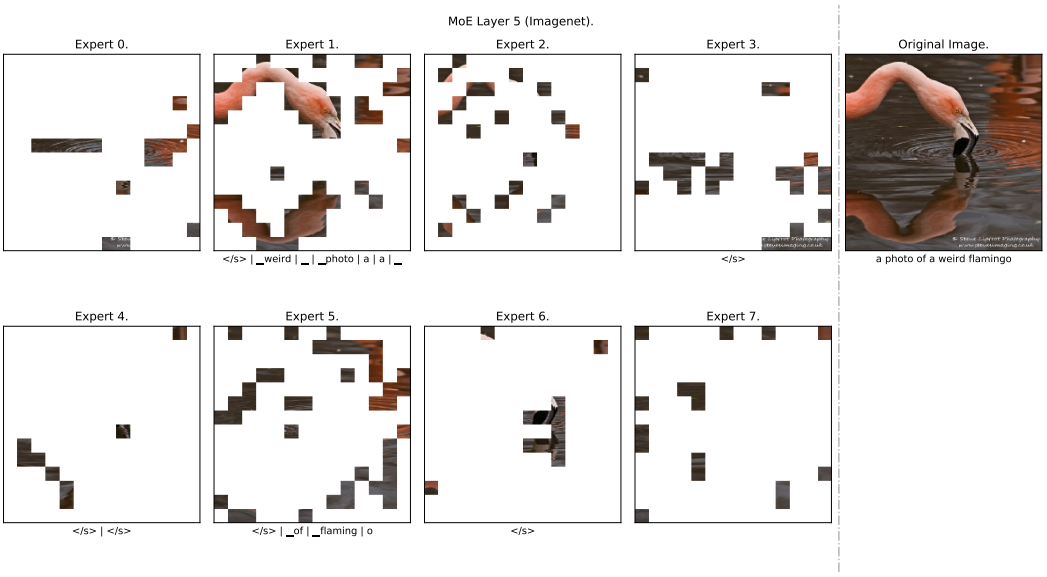


Figure 28: **Token routing for an Imagenet input.** B/16 model with 8 experts, we show original tokens (both image and text) as routed at the third MoE layer (corresponding to the sixth encoder block). The original image and text are displayed on the right-hand side.

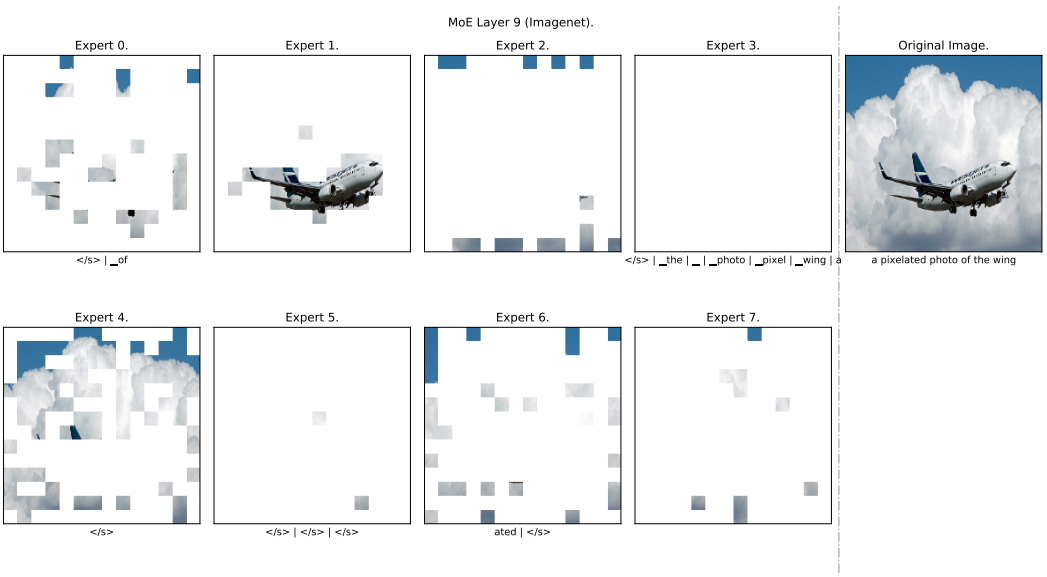


Figure 29: **Token routing for an Imagenet input.** B/16 model with 8 experts, we show original tokens (both image and text) as routed at the previous-to-last MoE layer (corresponding to the tenth encoder block, while we use zero-indexing). The original image and text are displayed on the right-hand side.

F.4 Routing Trajectories

In this section, we try to have a look at the overall trajectories followed by both image and text tokens across the network. While definitely a complex endeavor, we show in Figure 30 for B/32 and Figure 31 for B/16 the main trajectories followed by such tokens. Interestingly enough, it seems that for both models and image tokens, the first two/three MoE layers are fairly interconnected – in other words, given the expert selected for some token in one layer, it may be hard to predict the next steps. Text tokens (probably given that very few experts are indeed often used for text) have more predictable trajectories.

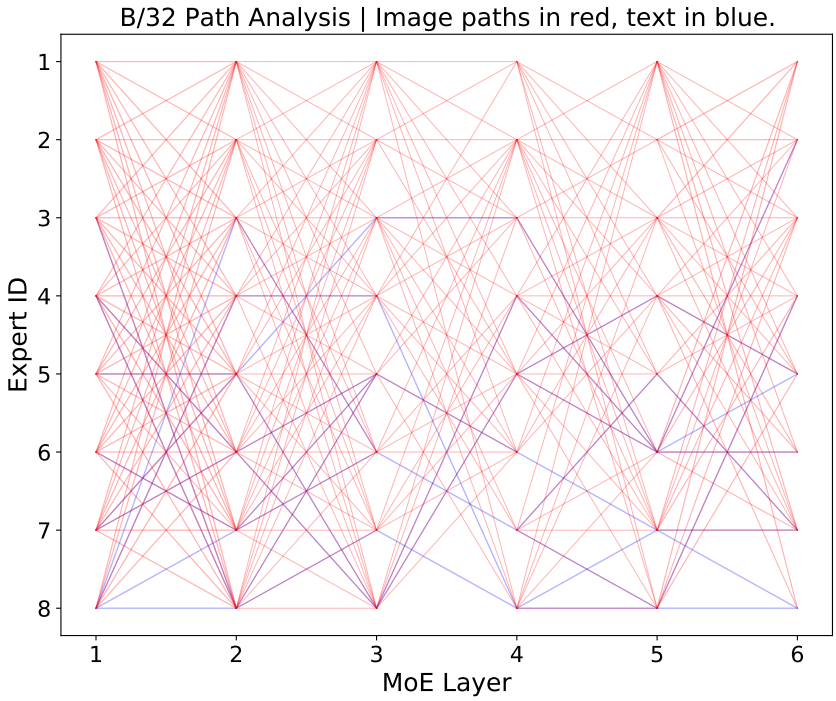


Figure 30: **Token trajectories.** B/32 model with 8 experts, we show the main expert-routes followed by text tokens (in blue) and image tokens (in red).

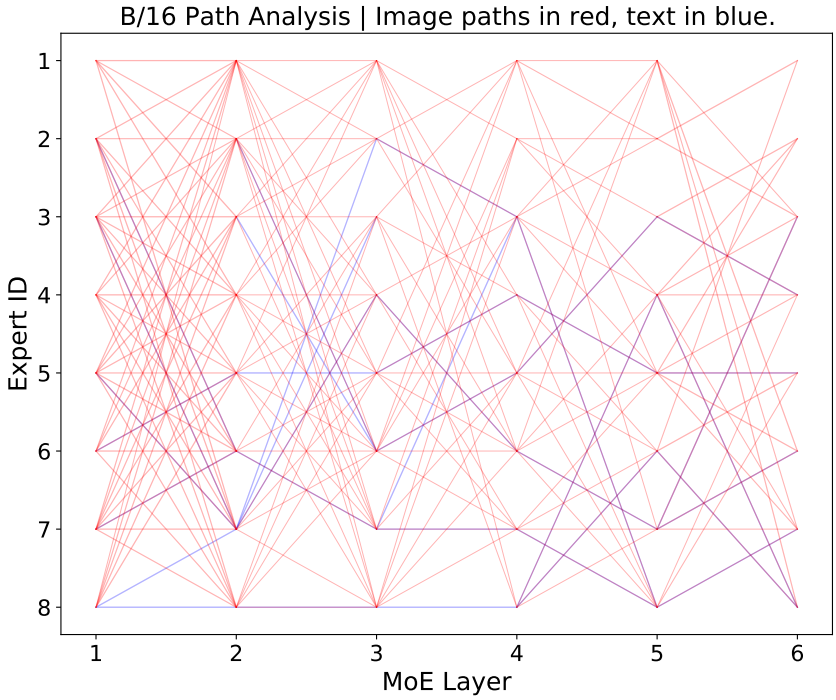


Figure 31: **Token trajectories.** B/16 model with 8 experts, we show the main expert-routes followed by text tokens (in blue) and image tokens (in red).

F.5 BPR rankings

The local entropy loss encourages concentrated routing predictions with high $p_{\max}$ for text. At the same time, BPR prioritises tokens with high $p_{\max}$. One might assume that this combination is effectively just ranking all text tokens first. The following plots give us some insight into how the buffers end up sorting tokens from both modalities. Figures 32 and 33 show the priority distribution on the training data for the B/32 and B/16 models, respectively. Under a data shift, Figures 34 and 35 show the same statistics for COCO data, and Figures 36 and 37 for ImageNet. In these cases, no extra training was performed (i.e., it is zero-shot). Overall, we see that while text tokens enjoy by default a much higher priority, this is not always the case, and some (important?) image patches are sometimes processed before other text tokens.

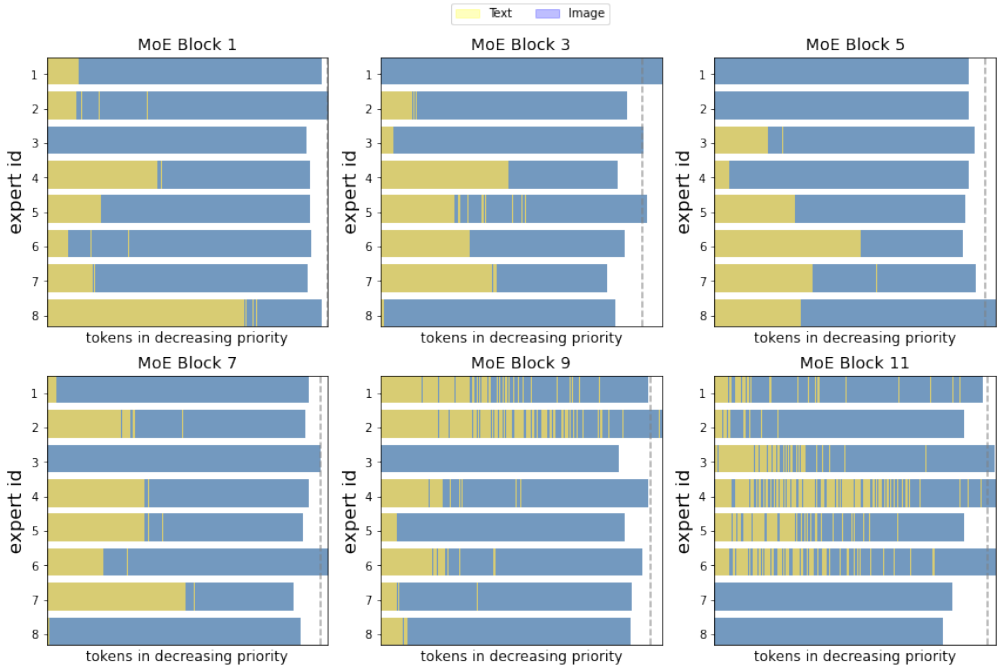


Figure 32: **Token priorities for training data.** B/32 model with 8 experts. We see that –especially in later layers– token priorities are mingled across modalities, whereas text tokens tend to have higher scores (and, thus, BPR priorities). Tokens to the left of the x-axis are given more priority. The vertical discontinuous line corresponds to the per-expert global capacity limit. Tokens beyond that point are not processed by the expert.

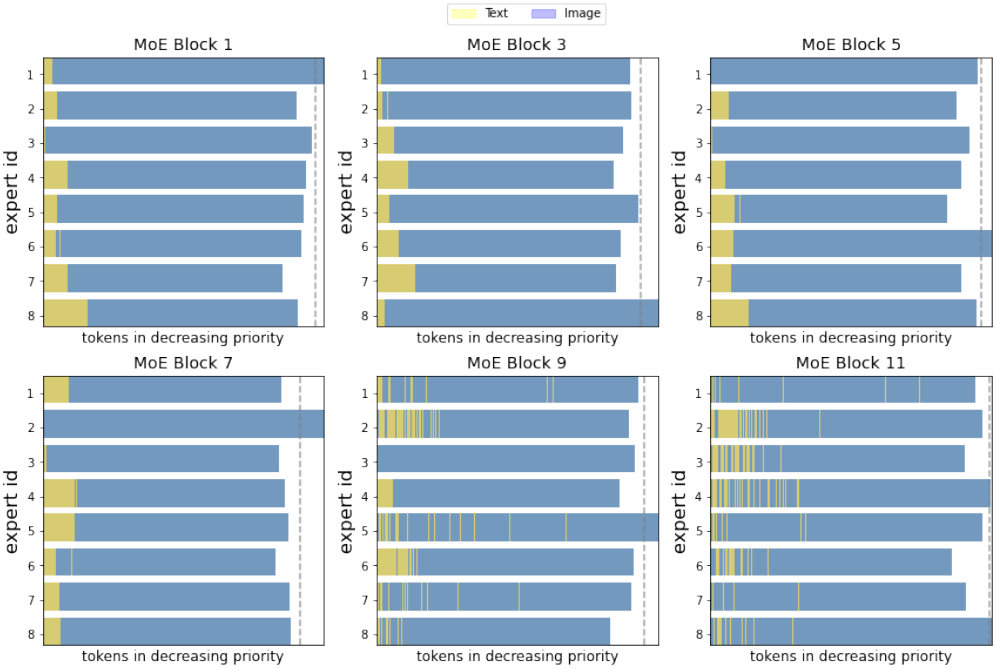


Figure 33: **Token priorities for training data.** B/16 model with 8 experts. We see that –especially in later layers– token priorities are mingled across modalities, whereas text tokens tend to have higher scores (and, thus, BPR priorities). Compared to the B/32 model, here we see a longer tail of low-priority image tokens. Tokens to the left of the x-axis are given more priority. The vertical discontinuous line corresponds to the per-expert global capacity limit. Tokens beyond that point are not processed by the expert.

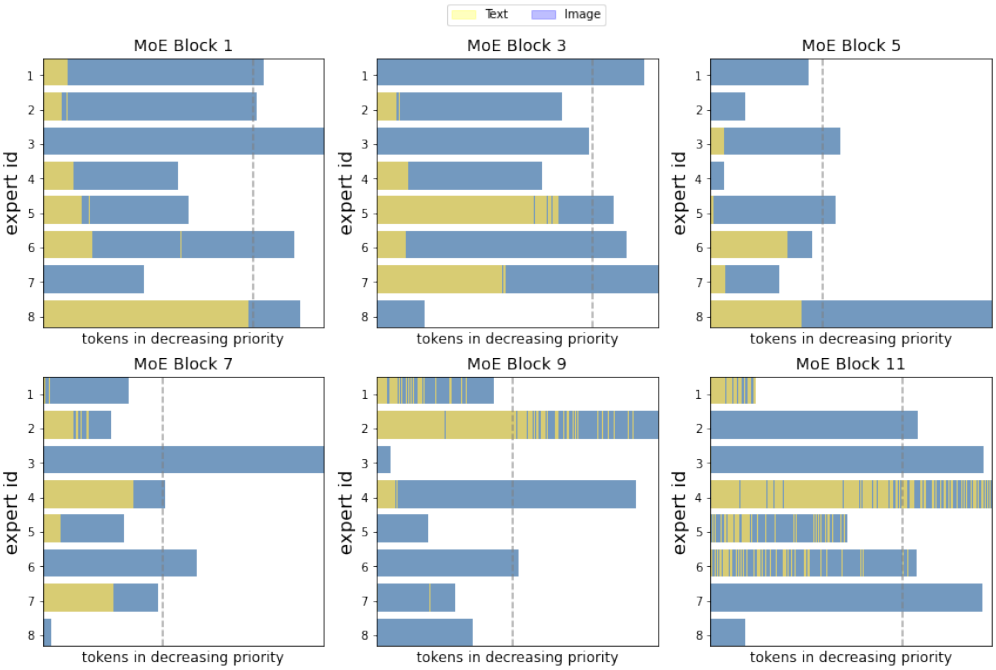


Figure 34: **Token priorities for COCO data.** B/32 model with 8 experts. We see that –especially in later layers– token priorities are mingled across modalities, whereas text tokens tend to have higher scores (and, thus, BPR priorities). Tokens to the left of the x-axis are given more priority. The vertical discontinuous line corresponds to the per-expert global capacity limit. Tokens beyond that point are not processed by the expert. Due to the distribution shift (this is evaluated on COCO, which was not the training data), we see lots of dropping is actually happening (mostly images, but also some text tokens).

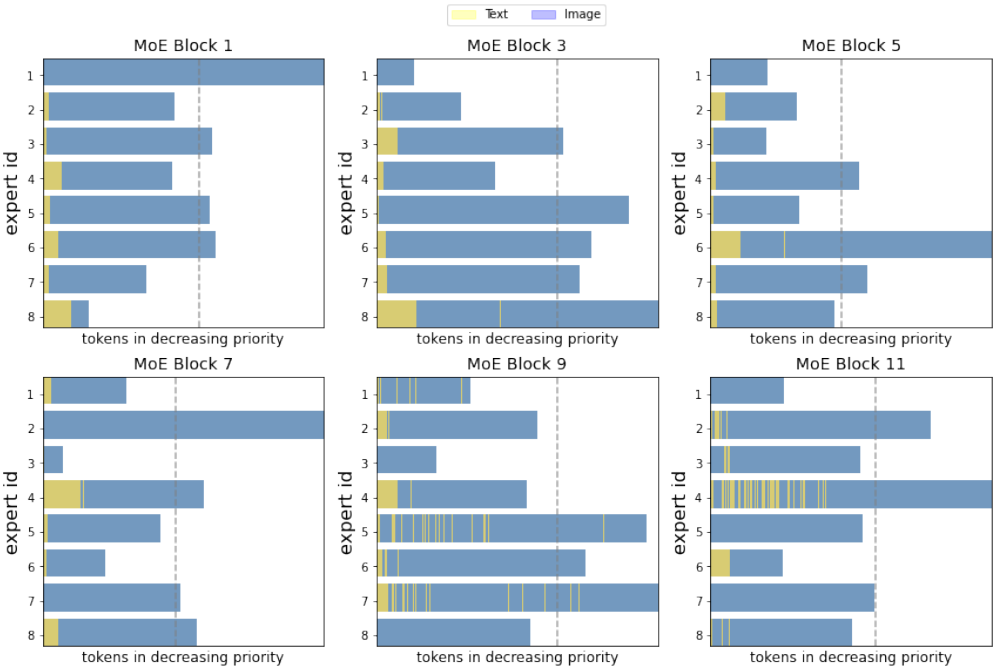


Figure 35: **Token priorities for COCO data.** B/16 model with 8 experts. We see that –especially in later layers– token priorities are mingled across modalities, whereas text tokens tend to have higher scores (and, thus, BPR priorities). Tokens to the left of the x-axis are given more priority. The vertical discontinuous line corresponds to the per-expert global capacity limit. Tokens beyond that point are not processed by the expert. Due to the distribution shift (this is evaluated on COCO, which was not the training data), we see lots of dropping is actually happening (while pretty much only image tokens).

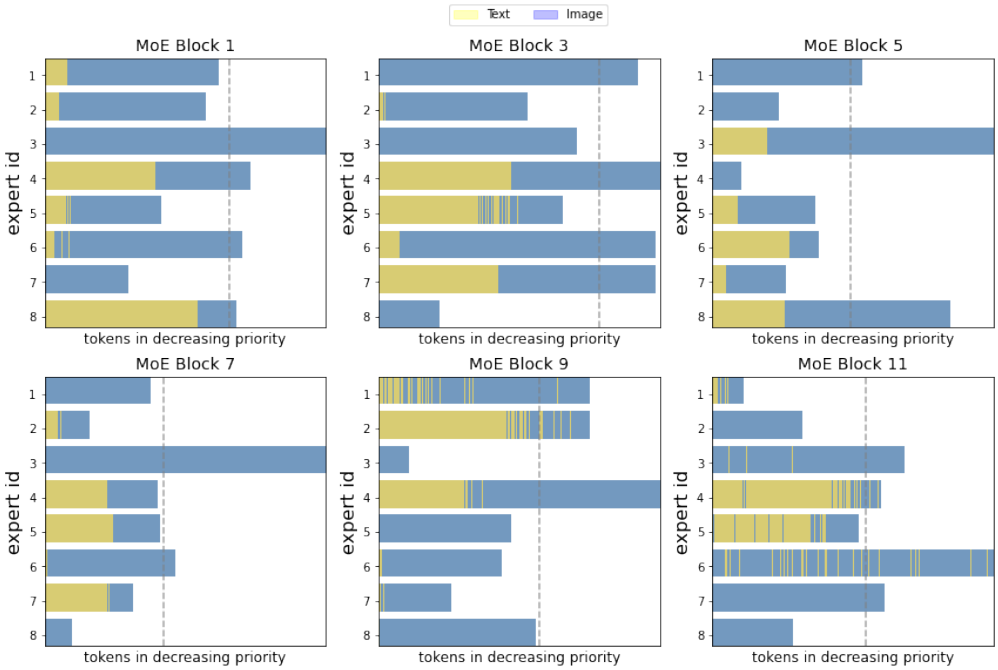


Figure 36: **Token priorities for ImageNet data.** B/32 model with 8 experts. We see that –especially in later layers– token priorities are mingled across modalities, whereas text tokens tend to have higher scores (and, thus, BPR priorities). Tokens to the left of the x-axis are given more priority. The vertical discontinuous line corresponds to the per-expert global capacity limit. Tokens beyond that point are not processed by the expert. Due to the distribution shift (this is evaluated on ImageNet, which was not the training data), we see lots of dropping is actually happening (mostly images, but also some text tokens).

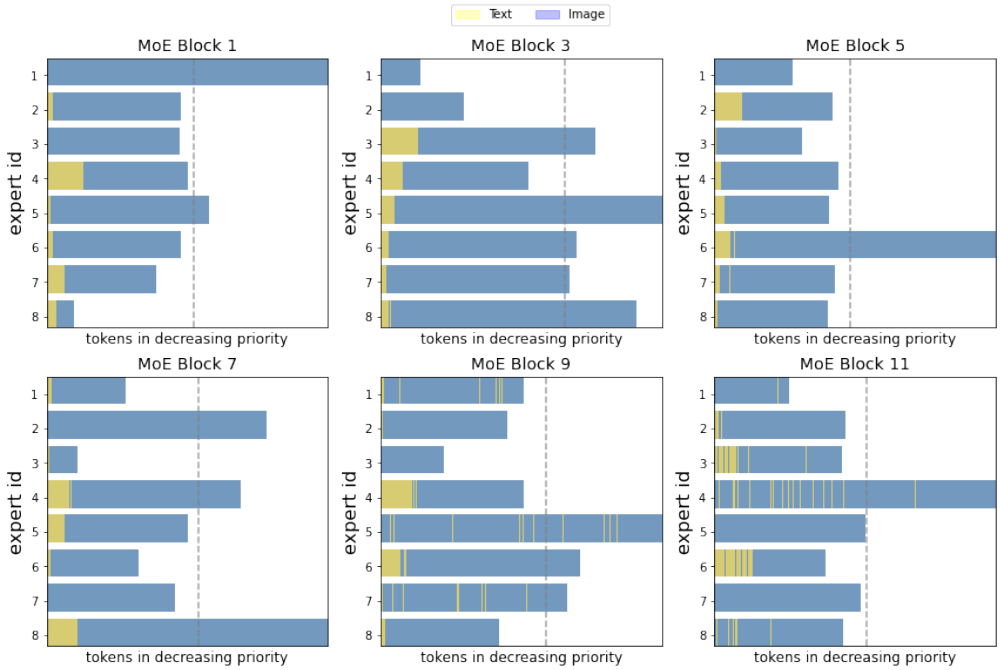


Figure 37: **Token priorities for ImageNet data.** B/16 model with 8 experts. We see that –especially in later layers– token priorities are mingled across modalities, whereas text tokens tend to have higher scores (and, thus, BPR priorities). Tokens to the left of the x-axis are given more priority. The vertical discontinuous line corresponds to the per-expert global capacity limit. Tokens beyond that point are not processed by the expert. Due to the distribution shift (this is evaluated on ImageNet, which was not the training data), we see lots of dropping is actually happening (mostly images, but also some text tokens).

G LIMoE-H/14 Analysis

In this section, we share some details and analysis regarding our largest model, the LIMoE-H/14. Figure 38 shows the development of the max routing probability across different MoE layers. Figure 2 shows qualitatively the specialization of image experts. Experts naturally specializing on semantic concepts such as body parts (hands, eyes), textures, fauna, food and doors. In Figure 39, we show the distribution of tokens per type and expert for every layer. Note that we set the entropy loss to approximately require at least 4 text experts, something that seems to agree well with the plot (in this case the ratio text:image tokens was close to 1:27).

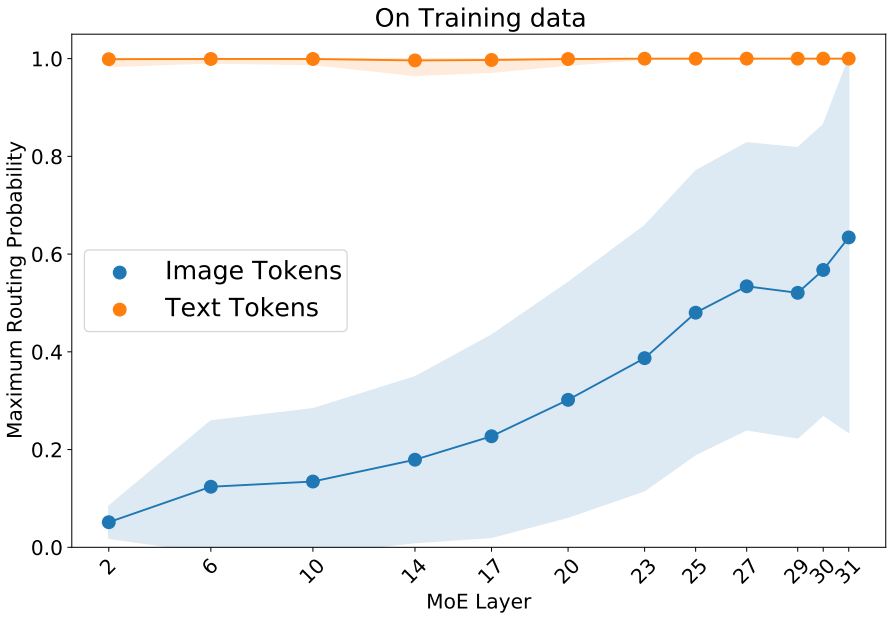


Figure 38: **Per-token $p_{\max}$ distribution for training data.** For LIMoE-H/14 model, we show the average and one standard deviation of the per-token maximum routing probability (corresponding to the selected expert). We see that for image tokens the model is increasingly confident, whereas for text tokens –given the local entropy loss– most of the predictions are close to one-hot.

G.1 Preliminary analysis of text routings

We analyse the routing distributions of text tokens for LIMoE-H/14, using NLTK [50] to distinguish between verbs, nouns, adjectives, prepositions and determiners. Note that the SentencePiece tokenizer breaks words into smaller units, which are not necessarily always handled by the same expert, so it is not possible to perfectly parse every token processed by every expert.

The majority of tokens are from images, so only 3-4 experts handle text in this scenario. Figure 40 contains preliminary analysis, showing for each expert the breakdown of tokens it handles. Though some experts process a bit of everything (e.g. experts 0 and 1 in layer 6 and 31), there are signs of some semantic specialization. There are often experts which process mostly padding tokens. In Layer 14, expert 1 processes no prepositions, determiners or verbs, focussing on nouns and adjectives (and some padding); similarly expert 1 processes very few nouns or adjectives, instead handling padding tokens.

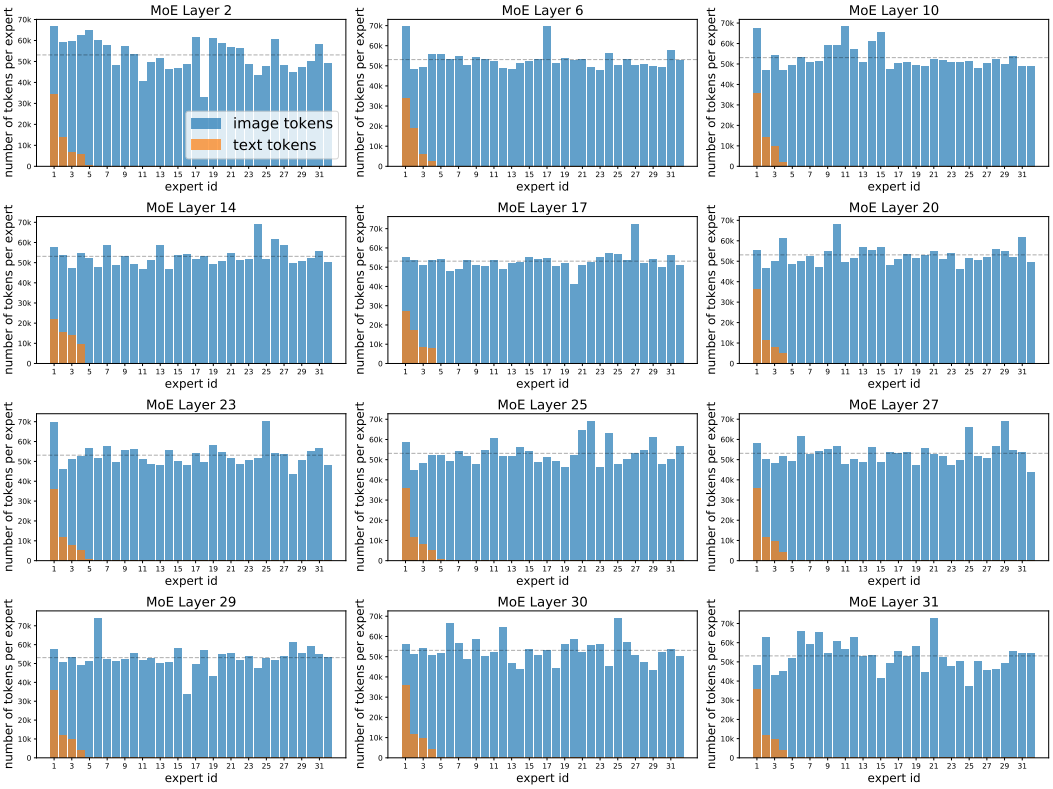


Figure 39: **Token routing per expert for** `LIMoE`**-H/14.** We show for each MoE layer and expert, the number of tokens per modality that were routed in a number of forward passes from the training data. When above the expert capacity (discontinuous horizontal line), some tokens were dropped – but not necessarily the image ones; for simplicity, we always show image tokens on top of text ones. In this setup, the ratio text:image tokens was close to 1:27.

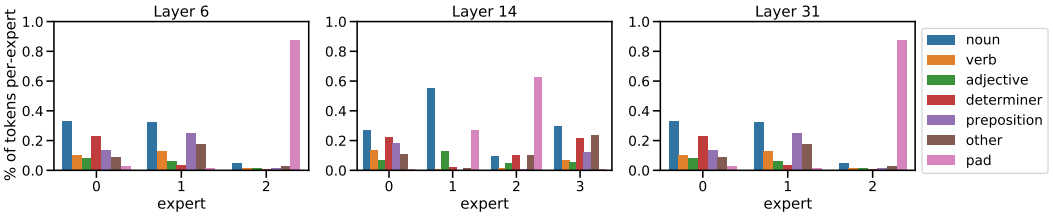


Figure 40: **Analysis of text routing for** `LIMoE`**-H/14**