Scaling Vision with Sparse Mixture of Experts

Anonymous Author(s)
Affiliation
Address
email

Abstract

Sparsely-gated Mixture of Experts networks (MoEs) have demonstrated excellent scalability in Natural Language Processing. In Computer Vision, however, almost all performant networks are "dense", that is, every input is processed by every parameter. We present a Vision MoE (V-MoE), a sparse version of the Vision Transformer, that is scalable and competitive with the largest dense networks. When applied to image recognition, V-MoE matches the performance of state-of-the-art networks, while requiring as little as *half* of the compute at inference time. Further, we propose an extension to the routing algorithm that can prioritize subsets of each input across the entire batch, leading to adaptive per-image compute. This allows V-MoE to trade-off performance and compute smoothly at test-time. Finally, we demonstrate the potential of V-MoE to scale vision models, and train a 15B parameter model that attains 90.24% on ImageNet.

1 Introduction

Deep learning historically shows that increasing network capacity and dataset size generally improves performance. In computer vision, large models pre-trained on large datasets often achieve the state of the art [41, 34, 27, 16, 3]. This approach has had even more success in Natural Language Processing (NLP), where large pre-trained models are ubiquitous, and perform very well on many tasks [32, 14]. Text Transformers [44] are the largest models to date, some with over 100B parameters [9]. However, training and serving such models is expensive [40, 30]. This is partially because these deep networks are typically "dense"– every example is processed using every parameter –thus, scale comes at high computational cost. In contrast, conditional computation [5] aims to increase model capacity while keeping the training and inference cost roughly constant by applying only a subset of parameters to each example. In NLP, sparse Mixture of Experts (MoEs) are gaining popularity [38, 28, 18], enabling training and inference with fewer resources while unlocking trillion parameter models.

In this work, we explore conditional computation for vision at scale. We introduce the Vision MoE (V-MoE), a sparse variant of the recent Vision Transformer (ViT) architecture [16] for image classification. The V-MoE replaces a subset of the dense feedforward layers in ViT with sparse MoE layers, where each image patch is "routed" to a subset of "experts" (MLPs). Due to unique failure modes and non-differentiability, routing in deep sparse models is challenging. We explore various design choices, and present an effective recipe for the pre-training and transfer of V-MoE, notably outperforming their dense counterparts. We further show that V-MoE models are remarkably flexible. The performance vs. inference-cost trade-off of *already trained* models can be smoothly adjusted during inference by modulating the sparsity level with respect to the input and/or the model weights.

With V-MoE, we can scale to model sizes of 15B parameters, the largest vision models to date. We match the performance of state-of-the-art dense models, while requiring fewer time to train. Alternatively, V-MoE can match the cost of ViT while achieving better performance. To help control this tradeoff, we propose Batch Prioritized Routing, a routing algorithm that repurposes model sparsity to skip the computation of some patches, reducing compute on uninformative image regions.

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

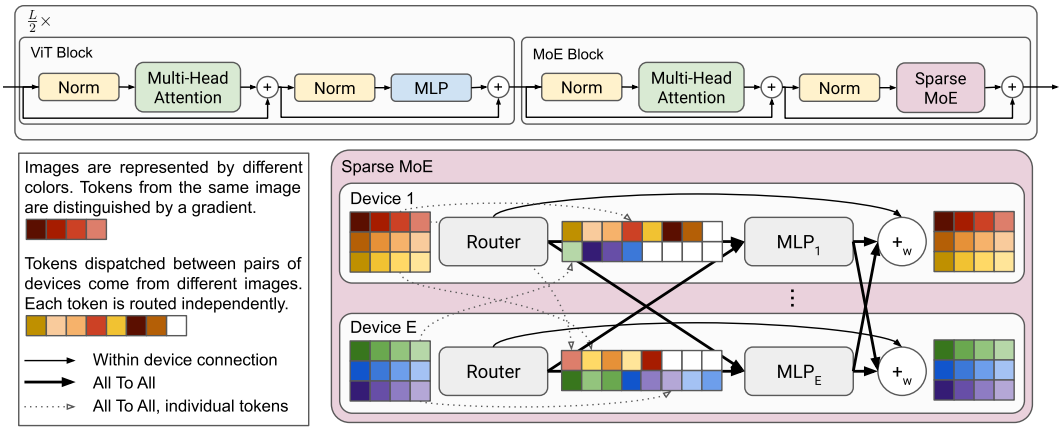


Figure 1: **Overview of the architecture.** V-MoE is composed of L ViT blocks. In some, we replace the MLP with a sparsely activated *mixture* of MLPs. Each MLP (the expert) is stored on a separate device, and processes a fixed number of tokens. The communication of these tokens between devices is shown in this example, which depicts the case when $k = 1$ expert is selected per token. Here each expert uses a capacity ratio $C = \frac{4}{3}$: the sparse MoE layer receives 12 tokens per device, but each expert has capacity for 16 ($\frac{16 \cdot 1}{12} = \frac{4}{3}$; see Section 2.4). Non-expert components of V-MoE such as routers, attention layers and normal MLP blocks are replicated identically across devices.

We summarize our main contributions as follows:
Vision models at scale. We present the Vision Mixture of Experts, a distributed sparsely-activated Transformer model for vision. We train models with up to 24 MoE layers, 32 experts per layer, and almost 15B parameters. We show that these models can be stably trained, seamlessly used for transfer, and successfully fine-tuned with as few as 1 000 datapoints. Moreover, our largest model achieves 90.24% test accuracy on ImageNet when fine-tuned.
Performance and inference. We show that V-MoEs strongly outperform their dense counterparts on upstream, few-shot and full fine-tuning metrics in absolute terms. Moreover, at inference time, the V-MoE models can be adjusted to either (i) match the performance of the largest dense model while using as little as half of the amount of compute, or actual runtime, or (ii) significantly outperform it at the same cost.
Batch Prioritized Routing. We propose a new priority-based routing algorithm that allows V-MoEs to discard the least useful patches. Thus, we devote less compute to each image. In particular, we show V-MoEs match the performance of the dense models while saving 20% of the training FLOPs.
Analysis. We provide some visualization of the routing decisions, revealing patterns and conclusions which helped motivate design decisions and may further improve understanding in the field.

2 The Vision Mixture of Experts

We first describe MoEs and sparse MoEs. We then present how we apply this methodology to vision, before explaining our design choices for the routing algorithm and the implementation of V-MoEs.

2.1 Conditional Computation with MoEs

Conditional computation aims at activating different subsets of a network for different inputs [5]. A mixture-of-experts model is a specific instantiation whereby different model "experts" are responsible for different regions of the input space [25].

We follow the setting of [38], who present for deep learning a mixture of experts layer with E experts as $\text{MoE}(\mathbf{x}) = \sum_{i=1}^{E} g(\mathbf{x})_i \; e_i(\mathbf{x})$ where $\mathbf{x} \in \mathbb{R}^D$ is the input to the layer, $e_i : \mathbb{R}^D \mapsto \mathbb{R}^D$ is the function computed by expert i, and $g : \mathbb{R}^D \mapsto \mathbb{R}^E$ is the "routing" function which prescribes the input-conditioned weight for the experts. Both e_i and g are parameterized by neural networks. As defined, this is still a dense network. However, if g is sparse, i.e., restricted to assign only $k \ll E$

non-zero weights, then unused experts need not be computed. This unlocks super-linear scaling of the number of model parameters with respect to inference & training compute.

2.2 MoEs for Vision

We explore the application of sparsity to vision in the context of the Vision Transformer (ViT) [16]. ViT has been shown to scale well in the transfer learning setting, attaining better accuracies than CNNs with less pre-training compute. ViT processes images as a sequence of patches. An input image is first divided into a grid of equal-sized patches. These are linearly projected to the Transformer's [44] hidden size. After adding positional embeddings, the patch embeddings (tokens) are processed by a Transformer, which consists predominately of alternating self-attention and MLP layers.

The MLPs have two layers and a GeLU [23] non-linearity: $\mathrm{MLP}(\mathbf{x}) = \mathbf{W}_2\, \sigma_{\text{gelu}}(\mathbf{W}_1 \mathbf{x})$. For Vision MoE, we replace a subset of these with MoE layers, where each expert is an MLP; see Figure 1. The experts have the same architecture $e_i(\mathbf{x}) = \mathrm{MLP}_{\theta_i}(\mathbf{x})$ but with different weights $\theta_i = (\mathbf{W}_1^i, \mathbf{W}_2^i)$. This follows a similar design pattern as the M4 machine translation model [28].

2.3 Routing

For each MoE layer in V-MoE, we use the routing function $g(\mathbf{x}) = \mathrm{TOP}_k\left(\mathrm{softmax}\left(\mathbf{W}\mathbf{x} + \epsilon\right)\right)$, where TOP_k is an operation that sets all elements of the vector to zero except the elements with the largest k values, and ϵ is sampled independently $\epsilon \sim \mathcal{N}(0, \frac{1}{E^2})$ entry-wise. In practice, we use $k = 1$ or $k = 2$. In the context of the Vision Transformer, $\mathbf{x}$ is a representation of an image token at some layer of the network. Therefore, V-MoE routes patch representations, not entire images.

The difference between previous formulations [38] is that we apply TOP_k *after* the softmax over experts weights, instead of *before*. This allows us to train with $k = 1$ (otherwise gradients with respect to routings are zero almost everywhere) and also performs better for $k > 1$ (see Appendix A).

Finally, we add a small amount of noise with standard deviation $\frac{1}{E}$ to the activations $\mathbf{W}\mathbf{x}$, which we find improves performance. We empirically found this performed well but that the setup was robust to this parameter. The noise typically altered routing decisions ~15% of the time in earlier layers, and ~2–3% in deeper layers.

2.4 Expert's Buffer Capacity

During training, sparse models may favor only a small set of experts [21, 36]. This common failure mode can cause two problems. First, statistical inefficiency: in the limit of collapse to a single expert, the model is no more powerful than a dense model. Second, computational inefficiency: imbalanced assignment of items to experts may lead to a poor hardware utilization.

To combat imbalance and simplify our implementation, we fix the *buffer capacity* of each expert (i.e. the number of tokens that each expert processes), and train our model with auxiliary losses that encourage load balancing. This is essentially the same approach as followed by [38, 28, 18]. In our case, we use slight variants of two of the auxiliary losses proposed in [38], as described in Appendix A.

We define the buffer capacity of an expert (B_e) as a function of the number of images in the batch (N), the number of tokens per image (P), the number of selected experts per token (k), the total number of experts (E), and the *capacity ratio* (C): $B_e = \mathrm{round}\left(\frac{kNPC}{E}\right)$.

If the router assigns more than B_e tokens to a given expert, it will not process them. Such tokens are not entirely 'lost' as their information is preserved by residual connections (the top diagram of Figure 1). If an expert is assigned fewer than B_e tokens, the rest of its buffer is zero-padded.

We use the *capacity ratio* to adjust the capacity of the experts. With $C > 1$, a *slack* capacity is added to account for a potential routing imbalance. This is typically useful for fine-tuning when the new data might come from a very different distribution than during upstream training. With $C < 1$, the router is forced to ignore some assignments. In Section 4 we propose a new algorithm that takes advantage of setting $C \ll 1$ to discard the least useful tokens and save compute during inference.

3 Transfer Learning

In this section, we first present training different variants of V-MoE on a large dataset (Section 3.2) in order to be used for Transfer Learning afterwards. The ability to easily adapt our massive models to new tasks, using a small amount of data from the new task, is extremely valuable: it allows to amortize the cost of pre-training across multiple tasks. We consider two different approaches to Transfer Learning: linear few-shot learning on fixed representations and full fine-tuning of the model.

3.1 Models

We build V-MoE on different variants of ViT [16]: ViT-S(mall), ViT-B(ase), ViT-L(arge) and ViT-H(uge), the hyperparameters of which are described in Appendix B.5. There are three additional major design decisions that affect the cost (and potentially the quality) of our model:

Number of MoE layers. Following [28], we place the MoEs on every other layer (we refer to these as V-MoE *Every-2*). In addition, we experimented with using fewer MoE layers, by placing them on the last-n *even* blocks (thus we dub these V-MoE *Last-n*). In Appendix E.1 we observe that, although using fewer MoE layers decreases the number of parameters of the model, it has typically little impact on quality and can speed-up the models significantly, since less communication overhead is incurred.

Number of selected experts k: The cost of our model does not depend on the total number of experts but the number of *selected* ones per token. Concurrent works in NLP fix $k = 1$ [18] or $k = 2$ [38, 28]. In our case, we use by default $k = 2$ (see Figure 10 in Appendix B for the exploration of different values of k), while we found the total number of experts $E = 32$ to be the sweet spot in our setting.

Buffer capacity C: As mentioned in Section 2.4, we use a fixed buffer capacity. While this is typically regarded as a downside or engineering difficulty to implement these models, we can adjust the *capacity ratio* to control different trade-offs. We can intentionally set it to a low ratio to save compute, using Batch Prioritized Routing (see Section 4). During upstream training, we set $C = 1.05$ by default to give a small amount of slack without increasing the cost noticeably.

Note that for a given trained model, the latter two—k and C—can be adjusted without further training, whereas the positioning and quantity of expert layers is effectively fixed to match pre-training.

3.2 Data

We pre-train our models on JFT-300M [41], a semi-automatically noisy-labeled dataset. It has ~ 305M training and 50 000 validation images, organised in a hierarchy of 18 291 classes (average 1.89 labels per image). We deduplicate it with respect to all our validation/test sets as in previous efforts [27].[1]

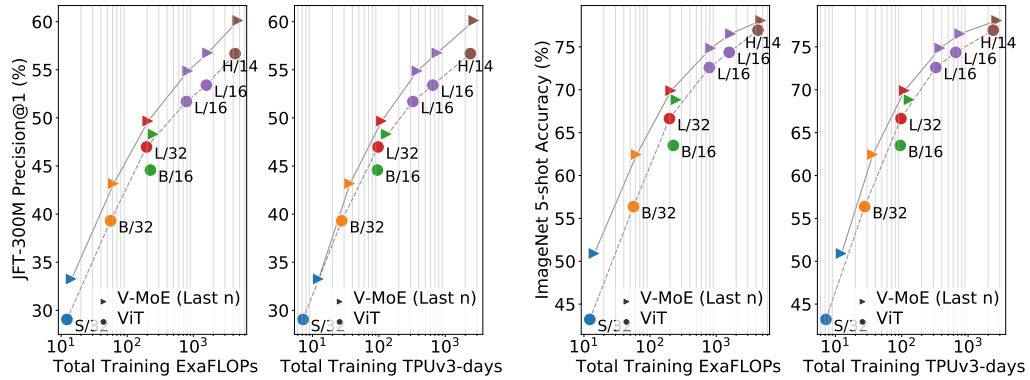
Our few-shot experiments on ImageNet (i.e. ILSVRC2012) use only 1, 5, or 10 shots per class to adapt the upstream model, evaluating the resulting model on the validation set.

We also fine-tuned the pre-trained models on the full training set (ca. 1M images). We report performance in a similar regime for four other datasets in Appendix B.5. Lastly, we explore the ability to fine-tune our large models in the low-data regime by evaluating them on the Visual Task Adaptation Benchmark (VTAB) [49], a diverse suite of 19 tasks with only 1 000 data points per task. As well as natural image classification, VTAB includes specialized tasks (e.g. medical or satellite imagery) and structured tasks (e.g. counting or assessing rotation/distance).

3.3 Upstream results

JFT is a multilabel dataset, so we measure model performance via precision@1 (see Appendix B.6 for details). Note that as in previous works[16], hyperparameters were tuned for transfer performance, and JFT precision could be improved at the expense of downstream tasks e.g. by reducing weight decay. Figure 2a shows the quality of different V-MoE and ViT variants with respect to total training compute & time. It shows models that select $k = 2$ experts and place MoEs in the last n even blocks ($n = 5$ for V-MoE-H, $n = 2$ otherwise), but the best results are achieved by V-MoE-H/14 *Every-2* (see Table 2, 14 is the patch size). See Appendix B.5 for results of all models.

[1] We also checked the effect of deduplication with respect to the ImageNet *training* set, showing negligible (within noise) impact on few-shot results (only 1-shot worsened, see Table 8).



(a) JFT-300M (b) ImageNet 5-shot

Figure 2: **JFT-300M Precision@1 and ImageNet 5-shot accuracy.** Colors represent different ViT variants, markers represent either standard ●ViT or ▶V-MoEs on the last n even blocks. The lines represent the Pareto frontier of ViT (dashed) and V-MoE (solid) variants.

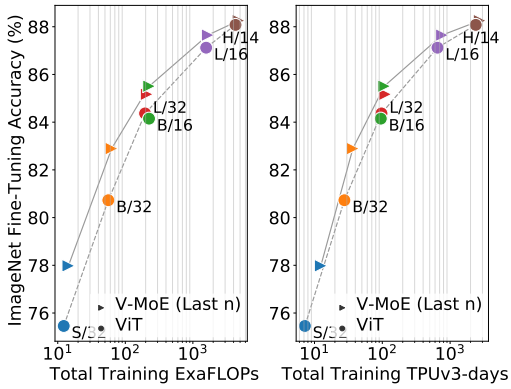


Figure 3: **ImageNet Fine-Tuning Accuracy**. Colors represent different VIT variants, markers represent either standard ●ViT or ▶V-MoEs on the last n even blocks. Lines represent the Pareto frontier of VIT (dashed) and V-MoE (solid) variants.

	ViT	V-MoE
L/16	$76.3_{\pm 0.5}$	$77.2_{\pm 0.4}$
H/14	$77.6_{\pm 0.2}$	$77.8_{\pm 0.4}$

Table 1: **VTAB.** Scores and 95% confidence intervals for ViT and V-MoE.

Expert models provide notable gains across all model sizes, for only a mild increase in FLOPs, establishing a new Pareto frontier (gray lines). Alternatively, we can match or improve performance of ViT models at lower cost (e.g. V-MoE-L/16 improves upon ViT-H/14). Similar conclusions hold for training time, which includes communication overhead of dispatching data across devices.

3.4 Linear few-shot results

We evaluate the quality of the representations learned using few-shot linear transfer. Given training examples from the new dataset $\{(X, Y)_i\}$, we use the pre-trained model $\mathcal{M}$ to extract a fixed representation $\mathcal{M}(x_i)$ of each image. We fit a linear regression model mapping $\mathcal{M}(x_i)$ to the one-hot encoding of the target labels Y_i, following [16] (see [22, Chapter 5] for background).

Figure 2b shows that the upstream gains are preserved under 5-shot ImageNet evaluation, considering both compute and time; in other words, the quality of the representations learned by V-MoE also outperforms ViT models when looking at a new task. Table 2 further shows the results on $\{1, 10\}$-shot for some selected models, and the full detailed results are available in Appendix B.5.

3.5 Full fine-tuning results

The typically most performant approach for Transfer Learning [15] consists of replacing the upstream classification head with a new task-specific one and fine-tuning the whole model. Though one may expect that massive models like V-MoEs require special handling for fine-tuning, we broadly follow the standard fine-tuning protocol for Vision Transformers. We use the auxiliary loss during fine-tuning as well, although we observe that it is often not needed in this step, as the router is already well trained. We explore the two sets of tasks considered therein:

Table 2: Main V-MoE & VIT models; Appendix B.5 shows results for additional models and datasets.

Model	Params	JFT prec@1	IN/1shot	IN/5shot	IN/10shot	IN/Fine-t.	ExaFLOPs	TPUv3-days
VIT-H/14	656M	56.68	62.34	76.95	79.02	88.08	4.27k	2.38k
V-MoE-L/16, Every-2	3.4B	57.65	62.41	77.10	79.01	87.39	2.17k	1.20k
V-MoE-H/14, Last-5	2.7B	60.12	62.95	78.08	80.10	88.25	4.75k	2.73k
V-MoE-H/14, Every-2	7.2B	60.62	63.38	78.21	80.33	88.08	5.79k	3.47k
V-MoE-15B, Every-2	14.7B	—	68.66	82.78	84.29	90.24	33.9k	16.8k
NFNet-F4+ [8]	527M	—	—	—	—	89.20	—	1.86k
MPL [33]	480M	—	—	—	—	90.20	—	22.5k

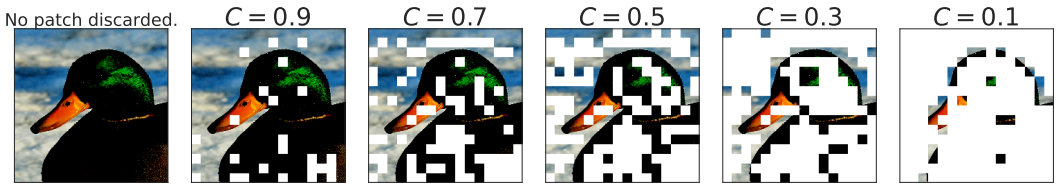


Figure 4: White patches are discarded tokens in the first layer of experts, for different capacities, using Batch Prioritized Routing (Section 4.1) with a V-MoE-H/14. See Appendix D for more examples.

Full data. We follow the setup of [16], except that we apply a dropout rate of 0.1 on the expert MLPs, and we halve the number of fine-tuning steps for all datasets other than ImageNet. Figure 3 shows the results on ImageNet (averaged over three runs). Here, V-MoE also performs better than dense counterparts, though we suspect the fine-tuning protocol could be further improved and tailored to the sparse models. See Appendix B.5 for all details, including results on other datasets.

Low-data regime. On the VTAB benchmark, we use a similar setup and hyperparameter budget as [16] (but fine-tune with half the schedule length). Table 1 shows that, while performance is similar for V-MoE-H/14, experts provide significant gains at the ViT-L/16 level, indicating that despite the large size of these models, they can still be fine-tuned with small amounts of data.

3.6 Scaling up V-MoE

Finally, we test how well V-MoE can scale vision models to a very large number of parameters, while continuing to improve performance. For this, we increase the size of the model and use a larger pre-training dataset: JFT-3B is a larger version of JFT-300M, it contains almost 3B images and is noisily annotated with 30k classes. Inspired by Anon et al. [2021][2], we apply the changes detailed in Appendix B.3, and train a 48-block V-MoE model, with every-2 expert placement (32 experts and $k = 2$), resulting in a model with 14.7B parameters, which we denote by V-MoE-15B.

We successfully train V-MoE-15B, which is, as far as we are aware, the largest vision model to date. It has an impressive accuracy of 82.78% on 5-shot ImageNet and 90.24% when fully fine-tuned, as shown in Table 2. Appendix B.5 includes more details about the model. Training this model required 16.8k TPUv3-core-days. To contextualize this result, the current state of the art on ImageNet is Meta Pseudo-Labelling (MPL) [33]. MPL trains an EfficientNet-based model on unlabelled JFT-300M using ImageNet pseudo-labelling, achieving 90.2% while requiring 22.5k TPUv3-core-days.

4 Skipping Tokens with Batch Prioritized Routing

We present a new routing algorithm that allows the model to prioritize important tokens (corresp. patches). By simultaneously reducing the capacity of each expert, we can discard the least useful tokens. Intuitively, not every patch is equally important to classify a given image, e.g., most background patches can be dropped to let the model only focus on the ones with the relevant entities.

[2] *Improvements to large scale Vision Transformers* provided through private communication.

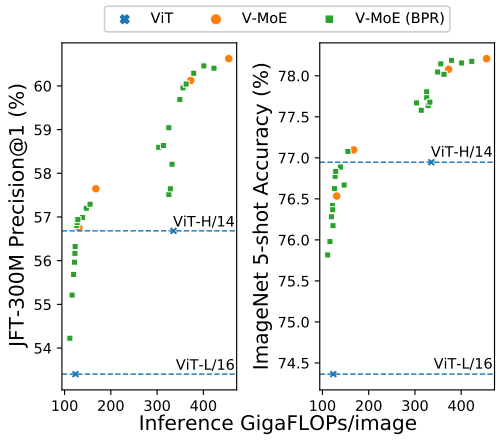


Figure 5: **Reducing compute with priority routing.** Performance vs. *inference* FLOPs for large models. V-MoEs with the original vanilla routing are represented by •, while ■ shows V-MoEs where BPR and a mix of $C \in \{0.6, 0.7, 0.8\}$ and $k \in \{1, 2\}$ are used to reduce compute. ViT models shown as **x**.

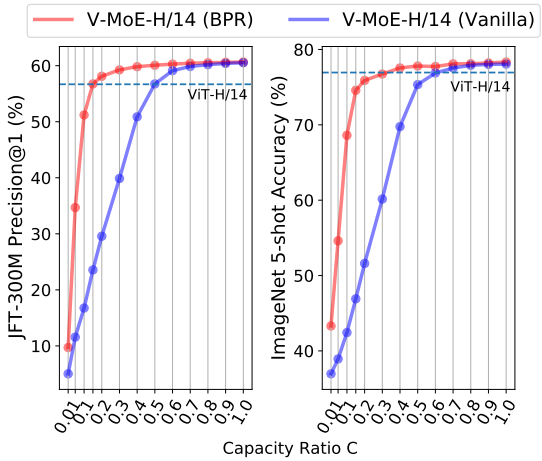


Figure 6: **Priority routing works where vanilla fails.** Performance vs. *inference* capacity ratio for a V-MoE-H/14 model with $k = 2$. Even for large C's BPR outperforms vanilla; at low C the difference is stark. BPR is competitive with dense by processing only 15-30% of the tokens.

4.1 From Vanilla Routing to Batch Prioritized Routing

With the notation from Section 2, the routing function g is applied row-wise to a batch of inputs $\mathbf{X} \in \mathbb{R}^{N \cdot P \times D}$. A batch contains N images composed of P tokens each; each row of $\mathbf{X}$ corresponds to the D-dimensional representation of a particular token of an image. Accordingly, $g(\mathbf{X})_{t,i} \in \mathbb{R}$ denotes the routing weight for the t-th token and the i-th expert.

In all routing algorithms considered, for $i < j$, every TOP-i assignment has priority over any TOP-j assignment. The router first tries to dispatch *all* i^{th} expert choices before assigning *any* j^{th} choice[3].

Given the TOP-i position, the default—or *vanilla*—routing, as used in [38, 28, 18], assigns tokens to experts as follows. It sequentially goes over the rows of $g(\mathbf{X})$ and assigns each token to its TOP-i expert *when* the expert's buffer is not full. As a result, priority is given to tokens depending on the rank of their corresponding row. While images in a batch are randomly ordered, tokens within an image follow a pre-defined *fixed* order. The algorithm is detailed in Algorithm 1 of Appendix C.

Batch Prioritized Routing (BPR). To favour the "most important" tokens, we propose to compute a *priority score* $s(\mathbf{x})$ on each token, and sort $g(\mathbf{X})$ accordingly before proceeding with the allocation. We sort tokens based on their maximum routing weight, formally $s(\mathbf{X})_t = \max_i g(\mathbf{X})_{t,i}$. The sum of TOP-$k$ weights, i.e. $s(\mathbf{X})_t = \sum_i g(\mathbf{X})_{t,i}$, worked equally well. These two simple approaches outperformed other options we explored, e.g., directly parameterising and learning the function s.

We reuse the router outputs as a proxy for the priority of allocation. Our experiments show this preserves the performant predictive behaviour of the model, even though the router outputs primarily encode how well tokens and experts can be paired, not the token's "importance" for the final classification task. Figure 4 visualizes token prioritisation with Batch Prioritized Routing for increasingly small capacities. Since all tokens across all images in the batch $\mathbf{X}$ compete with each other, different images may receive different amounts of compute. We summarize BPR in Algorithm 2, in Appendix C.

4.2 Skip tokens with low capacity C

Batch Prioritized Routing opens the door to reducing the buffer size by smartly selecting which tokens to favor. This can have a dramatic impact in the computational cost of the overall sparse model. We discuss now inference and training results with C defined in Section 2.4 in the regime $C \ll 1$.

[3] A token may however successfully assign all its TOP-k choices while another may not allocate a single one. This can happen for instance if the latter selects very popular experts that ran out of capacity.

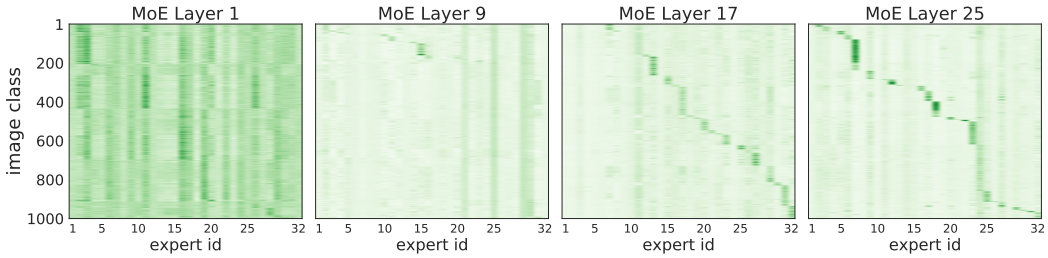


Figure 7: **Deeper routing decisions correlate with image classes**. We show 4 MoE layers of a V-MoE-H/14. The x-axis corresponds to the 32 experts in a layer. The y-axis are the 1 000 ImageNet classes; orderings for both axes are different across plots. For each pair (expert e, class c) we show the average routing weight for the tokens corresponding to all images with class c for that particular expert e. Figure 29 includes all the remaining layers; see Appendix E.2 for details.

At inference time. Prioritized routing is agnostic to how the model was originally trained. Figure 6 shows the effect of reducing compute at inference time by using BPR versus vanilla routing, on a VIT-H/14 model *trained* using vanilla routing. The difference in performance between both methods is remarkable —especially for $C \leq 0.5$, where the model truly starts fully dropping tokens, as $k = 2$. Also, BPR allows the model to be competitive with the dense one even at quite low capacities. As shown in Figure 5 for V-MoE-L/16 and V-MoE-H/14, Batch Prioritized Routing and low C allow V-MoE to smoothly trade-off performance and FLOPS at inference time, quite a unique model feature. More concretely, Table 9 shows V-MoE models can beat the dense VIT-H performance by using less than half the FLOPs and less than 60% of the runtime. Conversely, we can match the inference FLOPs cost and preserve a one-point accuracy gain in ImageNet/5shot and almost three-point in JFT precision at one (Table 10). Dense models generally require less runtime for the same amount of FLOPs due to the data transfer involved in the V-MoE implementation.

At training time. Batch Prioritized Routing can also be leveraged during training. In Appendix C we show how expert models with max-weight routing can match the dense performance while saving around 20% of the total training FLOPs, and strongly outperform vanilla with a similar FLOP budget.

5 Model Analysis

Although large-scale sparse MoEs have led to strong performance [18, 28, 38], little is known and understood about how the internals of those complex models work. We argue that such exploratory experiments can inform the design of new algorithms. In this section, we provide the first such analysis at this scale, which guided the development of the algorithms presented in the paper.

Specialized experts. Intuitively, routers should learn to distribute images across experts based on their similarity. For instance, if the model had three experts, and the task mainly involved three categories—say animals, cars, and buildings—one would expect an expert to specialize in each of those. We test this intuition, with some obvious caveats: (a) experts are placed at several network depths, (b) k experts are combined, and (c) routing happens at the token rather than the image level.

Figure 7 illustrates how many images of a given ImageNet class use each expert. The plots were produced by running a fine-tuned V-MoE-H *Every-2* model. Interestingly, we saw similar patterns with the upstream model without fine-tuning. Experts specialize in discriminating between small sets of classes (those primarily routed through the expert). In earlier MoE layers we do not observe this. Experts may instead focus on aspects common to all classes (background, basic shapes, colours) - for example, Figure 30 (Appendix E) shows correlations with patch location in earlier layers.

The value of routers. After training a sparse MoE, it is natural to study the usefulness of the learned routers, in the light of several pitfalls. For example, the routers may just act as a load balancer if experts end up learning very similar functions, or the routers may simply choose poor assignments. In Appendix E.1, we replace, after training, one router at a time with a uniformly random router. The models are robust to early routing changes while more sensitive to the decisions in the last layers.

Routing weights distributions. We analyse the router outputs in Appendix E.3, and observe the distribution of selected weights varies wildly across different mixture of experts layers.

Changing k at inference time. We have observed expert models are remarkably flexible. Somewhat surprisingly, sparse models are fairly robust to mismatches between their training and inference configurations. In Appendix E.4, we explore the effect of training with some original value of k while applying the model at inference time with a different $k' \neq k$. This can be handy to control (decrease or increase) the amount of FLOPs per input in a particular production system.

6 Related work

Conditional Computation. To grow the number of model parameters without proportionally increasing the computational cost, conditional computation [5, 12, 11] only activates some relevant parts of the model in an *input-dependent* fashion, like in decision trees [7]. In deep learning, the activation of portions of the model can use stochastic neurons [6] or reinforcement learning [4, 13, 37].

Mixture of Experts. MoEs [25, 26, 10, 47] combine the outputs of sub-models known as *experts* via a *router* in an input-dependent way. MoEs have successfully used this form of conditional computation in a range of applications [19, 24, 42, 39, 48]. An input can select either all experts [17] or only a sparse mixture thereof as in recent massive language models [38, 28, 18].

MoEs for Language. MoEs have recently scaled language models up to trillions of parameters. Our approach is inspired by [38] who proposed a top-k gating in LSTMs, with auxiliary losses ensuring the expert balancedness [21]. [28] further scaled up this approach for transformers, showing strong gains for neural machine translation. With over one trillion parameters and one expert per input, [18] sped up pre-training compared to a dense baseline [34] while showing gains thanks to transfer and distillation. [29] alternatively enforced a balanced routing by solving a linear assignment problem.

MoEs for Vision. For computer vision, previous work on MoEs [17, 2, 20, 1, 45, 31, 46] focused on architectures whose scale is considerably smaller than that of both language models and our model. In DeepMoE [45], the "experts" are the channels of convolutional layers that are adaptively selected by a multi-headed sparse gate. This is similar to [46] where the kernels of convolutional layers are activated on a per-example basis. Other approaches use shallow MoEs, learning a *single router*, either disjointly [20] or jointly [2], together with CNNs playing the role of experts. [1] further have a cost-aware procedure to bias the assignments of inputs across the experts. Unlike shallow MoEs, we operate with up to several tens of routing decisions *per token* along the depth of the model. Scaling up routing depth was marked as a major challenge in [35], which we successfully tackle in our work.

7 Conclusions

We have employed sparse conditional computation to train some of the largest vision models to date, showing significant improvements in representation learning and transfer learning. Alongside V-MoE, we have proposed Batch Prioritized Routing, which allows successful repurposing of *model* sparsity to introduce sparsity *with respect to the inputs*. This can be done without further adapting the model, allowing the re-use of trained models with sparse conditional computation.

This has interesting connotations for recent work in NLP using sparse models; recent analysis shows model sparsity is the most promising way to reduce model CO_2 emissions [30] and that 90% of the footprint stems from inference costs — we present an algorithm which takes the most efficient models and makes them even *more* efficient without any further model adaptation.

This is just the beginning of conditional computation at scale for vision; extensions include scaling up the expert count, reducing dependance on data and improving transfer of the representations sparse models. Directions relating to heteregonous expert architectures and conditional variable-length routes should also be fruitful. We expect increasing importance of sparse model scaling, especially in data rich domains such as large scale multimodal or video modelling.

References

[1] A. Abbas and Y. Andreopoulos. Biased mixtures of experts: Enabling computer vision inference under data transfer limitations. *IEEE Trans. Image Processing*, 29:7656–7667, 2020.

[2] K. Ahmed, M. H. Baig, and L. Torresani. Network of experts for large-scale image categorization. In *ECCV*, 2016.

[3] A. Arnab, M. Dehghani, G. Heigold, C. Sun, M. Lučić, and C. Schmid. ViViT: A video vision transformer. *arXiv preprint arXiv:2103.15691*, 2021.

[4] E. Bengio, P.-L. Bacon, J. Pineau, and D. Precup. Conditional computation in neural networks for faster models. *arXiv preprint arXiv:1511.06297*, 2015.

[5] Y. Bengio. Deep learning of representations: Looking forward. In *International Conference on Statistical Language and Speech Processing*, pages 1–37, 2013.

[6] Y. Bengio, N. Léonard, and A. Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[7] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen. *Classification and regression trees*. CRC press, 1984.

[8] A. Brock, S. De, S. L. Smith, and K. Simonyan. High-performance large-scale image recognition without normalization. *arXiv preprint arXiv:2102.06171*, 2021.

[9] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[10] K. Chen, L. Xu, and H. Chi. Improved learning algorithms for mixture of experts in multiclass classification. *Neural networks*, 12(9):1229–1252, 1999.

[11] K. Cho and Y. Bengio. Exponentially increasing the capacity-to-computation ratio for conditional computation in deep learning. *arXiv preprint arXiv:1406.7362*, 2014.

[12] A. Davis and I. Arel. Low-rank approximations for conditional feedforward computation in deep neural networks. *arXiv preprint arXiv:1312.4461*, 2013.

[13] L. Denoyer and P. Gallinari. Deep sequential neural network. *arXiv preprint arXiv:1410.0510*, 2014.

[14] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019.

[15] G. S. Dhillon, P. Chaudhari, A. Ravichandran, and S. Soatto. A baseline for few-shot image classification. In *ICLR*, 2020.

[16] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

[17] D. Eigen, M. Ranzato, and I. Sutskever. Learning factored representations in a deep mixture of experts. *arXiv preprint arXiv:1312.4314*, 2013.

[18] W. Fedus, B. Zoph, and N. Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *arXiv preprint arXiv:2101.03961*, 2021.

[19] D. M. Gavrila and S. Munder. Multi-cue pedestrian detection and tracking from a moving vehicle. *International journal of computer vision*, 73(1):41–59, 2007.

[20] S. Gross, M. Ranzato, and A. Szlam. Hard mixtures of experts for large scale weakly supervised vision. In *CVPR*, 2017.

[21] J. V. Hansen. Combining predictors: comparison of five meta machine learning methods. *Information Sciences*, 119(1-2):91–105, 1999.

[22] T. Hastie, R. Tibshirani, and J. Friedman. *The elements of statistical learning: data mining, inference, and prediction*. Springer, 2017.

[23] D. Hendrycks and K. Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

[24] Y. H. Hu, S. Palreddy, and W. J. Tompkins. A patient-adaptable ECG beat classifier using a mixture of experts approach. *IEEE Trans. Biomedical Engineering*, 44(9):891–900, 1997.

[25] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991.

[26] M. I. Jordan and R. A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural computation*, 6(2):181–214, 1994.

[27] A. Kolesnikov, L. Beyer, X. Zhai, J. Puigcerver, J. Yung, S. Gelly, and N. Houlsby. Big transfer (BiT): General visual representation learning. In *ECCV*, 2020.

[28] D. Lepikhin, H. Lee, Y. Xu, D. Chen, O. Firat, Y. Huang, M. Krikun, N. Shazeer, and Z. Chen. GShard: Scaling giant models with conditional computation and automatic sharding. In *ICLR*, 2021.

[29] M. Lewis, S. Bhosale, T. Dettmers, N. Goyal, and L. Zettlemoyer. Base layers: Simplifying training of large, sparse models. *arXiv preprint arXiv:2103.16716*, 2021.

[30] D. Patterson, J. Gonzalez, Q. Le, C. Liang, L.-M. Munguia, D. Rothchild, D. So, M. Texier, and J. Dean. Carbon emissions and large neural network training. *arXiv preprint arXiv:2104.10350*, 2021.

[31] S. Pavlitskaya, C. Hubschneider, M. Weber, R. Moritz, F. Huger, P. Schlicht, and M. Zollner. Using mixture of expert models to gain insights into semantic segmentation. In *CVPR Workshops*, 2020.

[32] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. In *NAACL*, 2018.

[33] H. Pham, Z. Dai, Q. Xie, M.-T. Luong, and Q. V. Le. Meta pseudo labels. *arXiv preprint arXiv:2003.10580*, 2020.

[34] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

[35] P. Ramachandran and Q. V. Le. Diversity and depth in per-example routing models. In *ICLR*, 2018.

[36] C. Rosenbaum, I. Cases, M. Riemer, and T. Klinger. Routing networks and the challenges of modular and compositional computation. *arXiv preprint arXiv:1904.12774*, 2019.

[37] C. Rosenbaum, T. Klinger, and M. Riemer. Routing networks: Adaptive selection of non-linear functions for multi-task learning. *arXiv preprint arXiv:1711.01239*, 2017.

[38] N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *ICLR*, 2017.

[39] C. Sminchisescu, A. Kanaujia, Z. Li, and D. Metaxas. Learning to reconstruct 3D human motion from Bayesian mixtures of experts. A probabilistic discriminative approach. *Dept. Comput. Sci., Univ. Toronto, Tech. Rep. CSRG-502*, 2004.

[40] E. Strubell, A. Ganesh, and A. McCallum. Energy and policy considerations for deep learning in NLP. *arXiv preprint arXiv:1906.02243*, 2019.

[41] C. Sun, A. Shrivastava, S. Singh, and A. Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *ICCV*, 2017.

[42] J. Tani and S. Nolfi. Learning to perceive the world as articulated: an approach for hierarchical learning in sensory-motor systems. *Neural Networks*, 12(7-8):1131–1141, 1999.

[43] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jégou. Training data-efficient image transformers & distillation through attention. *arXiv preprint arXiv:2012.12877*, 2020.

[44] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

[45] X. Wang, F. Yu, L. Dunlap, Y.-A. Ma, R. Wang, A. Mirhoseini, T. Darrell, and J. E. Gonzalez. Deep mixture of experts via shallow embedding. In *Uncertainty in Artificial Intelligence*, 2020.

[46] B. Yang, G. Bender, Q. V. Le, and J. Ngiam. Condconv: Conditionally parameterized convolutions for efficient inference. *arXiv preprint arXiv:1904.04971*, 2019.

[47] S. E. Yuksel, J. N. Wilson, and P. D. Gader. Twenty years of mixture of experts. *IEEE transactions on neural networks and learning systems*, 23(8):1177–1193, 2012.

[48] A. J. Zeevi, R. Meir, and R. J. Adler. Time series prediction using mixtures of experts. In *NeurIPS*, 1997.

[49] X. Zhai, J. Puigcerver, A. Kolesnikov, P. Ruyssen, C. Riquelme, M. Lucic, J. Djolonga, A. S. Pinto, M. Neumann, A. Dosovitskiy, L. Beyer, O. Bachem, M. Tschannen, M. Michalski, O. Bousquet, S. Gelly, and N. Houlsby. A large-scale study of representation learning with the visual task adaptation benchmark. *arXiv preprint arXiv:1910.04867*, 2019.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See Figure 2 and Table 2.
   (b) Did you describe the limitations of your work? [Yes] See Section 5.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] We acknowledge in the introduction the massive scale of models, training and inference in modern deep learning. The work presented in this paper is definitely part of this trend, but as we argue in Section 7, we hope sparse models can help reduce the environmental impact and footprint of practical machine learning at scale.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] The code and the data are proprietary.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Table 5.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] Most of our large models could not be trained several times. We provide error bars only for small scale experiments, e.g., see Table 1.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Table 2.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [No] We cite the original assets only.
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]