From Sparse to Soft Mixtures of Experts

Joan Puigcerver* **Carlos Riquelme***
Basil Mustafa **Neil Houlsby**
Google DeepMind

Abstract

Sparse mixture of expert architectures (MoEs) scale model capacity without large increases in training or inference costs. Despite their success, MoEs suffer from a number of issues: training instability, token dropping, inability to scale the number of experts, or ineffective finetuning. In this work, we propose Soft MoE, a *fully-differentiable* sparse Transformer that addresses these challenges, while maintaining the benefits of MoEs. Soft MoE performs an implicit soft assignment by passing different weighted combinations of all input tokens to each expert. As in other MoE works, experts in Soft MoE only process a subset of the (combined) tokens, enabling larger model capacity at lower inference cost. In the context of visual recognition, Soft MoE greatly outperforms standard Transformers (ViTs) and popular MoE variants (Tokens Choice and Experts Choice). For example, Soft MoE-Base/16 requires 10.5× lower inference cost (5.7× lower wall-clock time) than ViT-Huge/14 while matching its performance after similar training. Soft MoE also scales well: Soft MoE Huge/14 with 128 experts in 16 MoE layers has over $40\times$ more parameters than ViT Huge/14, while inference time cost grows by only 2%, and it performs substantially better.

1 Introduction

Larger Transformers improve performance at increased computational cost. Recent studies suggest that model size and training data must be scaled together to optimally use any given training compute budget (Kaplan et al., 2020; Hoffmann et al., 2022; Zhai et al., 2022a). A promising alternative that allows to scale models in size without paying their full computational cost is sparse mixtures of experts (MoEs). Recently, a number of successful approaches have proposed ways to sparsely activate token paths across the network in language (Lepikhin et al., 2020; Fedus et al., 2022), vision (Riquelme et al., 2021), and multimodal models (Mustafa et al., 2022).

At the core of sparse MoE Transformers lies a discrete optimization problem: deciding which modules should be applied to each input token. These modules are commonly referred to as *experts* and are usually MLPs. Many techniques have been devised to find good token-to-expert matches: linear programs (Lewis et al., 2021), reinforcement learning (Bengio et al., 2015), deterministic fixed rules (Roller et al., 2021), optimal transport (Liu et al., 2022), greedy top-k experts per token (Shazeer et al., 2017), or greedy top-k tokens per expert (Zhou et al., 2022). In many cases, heuristic auxiliary losses are required to balance utilization of experts and minimize unassigned tokens. These challenges can be exacerbated in out-of-distribution scenarios: small inference batch sizes, novel inputs, or in transfer learning.

We introduce a new approach, Soft MoE, that overcomes many of these challenges. Rather than employing a sparse and discrete router that tries to find a good *hard* assignment between tokens and experts, Soft MoEs instead perform a *soft* assignment by mixing tokens. In particular, we compute several weighted averages of all tokens—with weights depending on both tokens and experts—and then we process each weighted average by its corresponding expert.

Soft MoE models avoid most of the challenges mentioned above which are caused by the discrete procedure at the core of sparse MoEs. Popular sparse MoE algorithms learn some router parameters, and the source of gradients is usually two-fold: post-multiplication of expert outputs with the *selected* routing scores, and

*Equal contribution. The order was decided by a coin toss.

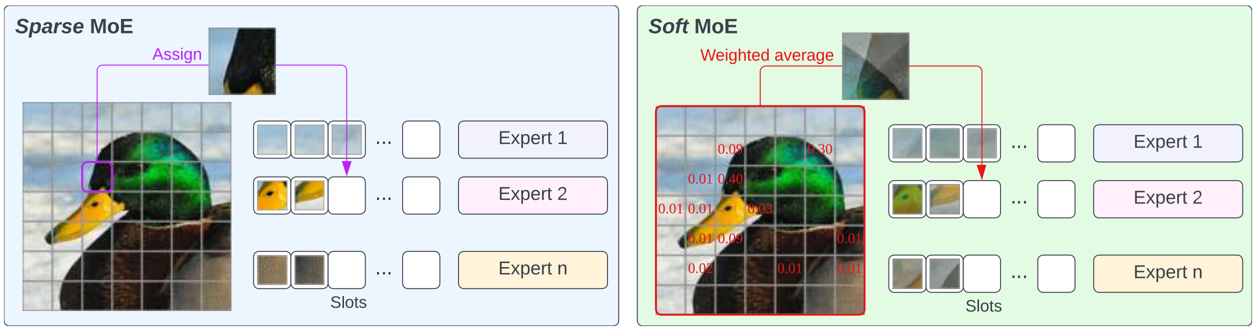


Figure 1: **Main differences between Sparse and Soft MoE layers.** While the router in Sparse MoE layers (left) learns to *assign* individual input tokens to each of the available slots, in Soft MoE layers (right) each slot is the result of a (different) *weighted average* of all the input tokens. Learning to make discrete assignments introduces several optimization and implementation issues that Soft MoE sidesteps.

auxiliary losses that enforce some desired behaviour and also depend on the routing scores. It has been observed that these mechanisms are often no better than random fixed routing (Roller et al., 2021). Soft MoE sidesteps this issue as every routing (or mixing) parameter is directly updated based on every single input token. Soft routing can provide *stability* while training a router; (Mustafa et al., 2022) observed that during training large fractions of input tokens can simultaneously change discrete routes through the network, leading to training challenges. Further, hard routing can be challenging with many experts, with most works training with just a few dozen. We show that Soft MoE scales to thousands of experts, and it is balanced by construction. Finally, there are no batch-effects at inference, where one input can affect routing (due to limited expert capacity), and hence prediction, for other inputs.

Soft MoE L/16 beats ViT H/14 on upstream, fewshot and finetuning while requiring almost half the training time, and being **2$\times$ faster at inference**. Moreover, Soft MoE B/16 matches ViT H/14 on fewshot and finetuning and outperforms it on upstream metrics after a comparable amount of training. Remarkably, Soft MoE B/16 is **5.7$\times$ faster at inference** despite having 5.5$\times$ the number of parameters of ViT H/14. Section 4 demonstrates Soft MoE's potential to extend to other tasks: we train a contrastive model text tower against the frozen vision tower, showing that representations learned via soft routing preserve their benefits for image-text alignment.

2 Soft Mixture of Experts

2.1 Algorithm description

The Soft MoE routing algorithm is depicted in Figure 2. We denote the inputs tokens for one sequence by $\mathbf{X} \in \mathbb{R}^{m \times d}$, where m is the number of tokens and d is their dimension. Each MoE layer uses a set of n expert functions[1] applied on individual tokens, namely $\{f_i : \mathbb{R}^d \to \mathbb{R}^d\}_{1:n}$. Each expert will process p *slots*, and each slot has a corresponding d-dimensional vector of parameters. We denote these parameters by $\mathbf{\Phi} \in \mathbb{R}^{d \times (n \cdot p)}$.

In particular, the input slots $\tilde{\mathbf{X}} \in \mathbb{R}^{(n \cdot p) \times d}$ are the result of convex combinations of all the m input tokens, $\mathbf{X}$:

$$\begin{aligned} \mathbf{D}_{ij} &= \frac{\exp((\mathbf{X\Phi})_{ij})}{\sum_{i'=1}^{m} \exp((\mathbf{X\Phi})_{i'j})} \\ \tilde{\mathbf{X}} &= \mathbf{D}^\top \mathbf{X}. \end{aligned} \qquad (1)$$

Notice that $\mathbf{D}$, which we call the *dispatch* weights, is simply the result of applying a softmax over the *columns* of $\mathbf{X\Phi}$. Then, as mentioned above, the corresponding expert function is applied on each slot (i.e. on rows of

[1] In practice, all experts apply the same function with different parameters, usually an MLP.

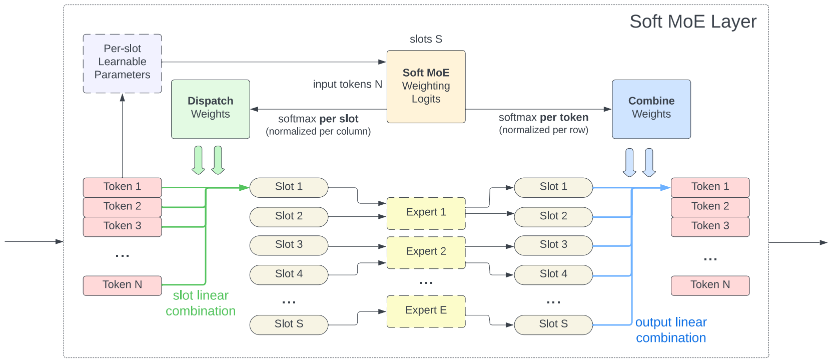


Figure 2: **The Soft MoE routing algorithm.** Soft MoE first computes scores or logits for every pair of input token and slot, based on some learnable per-slot parameters. These logits are then normalized per slot (columns) and every slot computes a linear combination of all the input tokens based on these weights (in green). Each expert (an MLP in this work) then processes its slots (e.g. 2 slots per expert, in this diagram). Finally, the same original logits are normalized per token (i.e. by row) and used to combine all the slot outputs, for every input token (in blue). Dashed boxes represent learnable parameters.

$\tilde{\mathbf{X}}$) to obtain the output slots:

$$\tilde{\mathbf{Y}}_i = f_{\lfloor i/p \rfloor}(\tilde{\mathbf{X}}_i). \tag{2}$$

Finally, the output tokens $\mathbf{Y}$ are computed as a convex combination of all $(n \cdot p)$ output slots, $\tilde{\mathbf{Y}}$, whose weights are computed similarly as before:

$$\mathbf{C}_{ij} = \frac{\exp((\mathbf{X\Phi})_{ij})}{\sum_{j'=1}^{n \cdot p} \exp((\mathbf{X\Phi})_{ij'})} \qquad \mathbf{Y} = \mathbf{C}\tilde{\mathbf{Y}}. \tag{3}$$

We refer to $\mathbf{C}$ as the *combine* weights, and it is the result of applying a softmax over the *rows* of $\mathbf{X\Phi}$.

Following the usual design for Sparse MoEs (Lepikhin et al., 2020; Fedus et al., 2022; Riquelme et al., 2021; Zhou et al., 2022), we replace a subset of the Transformer's MLP blocks with Soft MoE blocks. In particular, we typically replace the second half of MLP blocks. The total number of slots is a key hyperparameter of Soft MoE layers because the time complexity depends on the number of slots rather than on the number of experts. For example, one can set the number of slots equal to the input sequence length to match the FLOPs of the equivalent dense Transformer.

```python
def soft_moe_layer(X, Phi, experts):
  # Compute the dispatch and combine weights.
  logits = jnp.einsum('md,dnp->mnp', X, Phi)
  D = jax.nn.softmax(logits, axis=(0,))
  C = jax.nn.softmax(logits, axis=(1, 2))
  # The input slots are a weighted average of all the input tokens,
  # given by the dispatch weights.
  Xs = jnp.einsum('md,mnp->npd', X, D)
  # Apply the corresponding expert function to each input slot.
  Ys = jnp.stack([
    f_i(Xs[i, :, :]) for i, f_i in enumerate(experts)],
    axis=0)
  # The output tokens are a weighted average of all the output slots,
  # given by the combine weights.
  Y = jnp.einsum('npd,mnp->md', Ys, C)
  return Y
```

Algorithm 1: Simple JAX (Bradbury et al., 2018) implementation of a Soft MoE layer. Full code is available at https://github.com/google-research/vmoe.

2.2 Properties of Soft MoEs and connections with Sparse MoEs

Fully differentiable At the core of all Sparse MoE algorithms there is an assignment problem between tokens and experts, which is usually subject to some specific capacity and balance constraints. Different algorithms relax the problem or approximate the solution in different ways: the Top-k or "Token Choice" router (Shazeer et al., 2017; Lepikhin et al., 2020; Riquelme et al., 2021), for instance, selects the top-k-scored experts for each token, while there are slots available in such expert (i.e. the expert has not filled its *capacity*). The "Expert Choice" router (Zhou et al., 2022) selects the top-*capacity*-scored tokens for each expert. Other works suggest more advanced (and often costly) algorithms to compute the assignments, such as approaches based on Linear Programming algorithms (Lewis et al., 2021), Optimal Transport (Liu et al., 2022; Clark et al., 2022) or Reinforcement Learning (Clark et al., 2022). Nevertheless virtually all of these approaches are discrete in nature, and thus non-differentiable. In contrast, all operations in Soft MoE layers are continuous and fully differentiable. Indeed, we can interpret the weighted averages with softmax scores as *soft* assignments –which motivates our algorithm's name– rather than the *hard* assignments that Sparse MoE methods typically use.

No token dropping and expert unbalance The classical routing mechanisms mentioned above tend to suffer from issues such as "token dropping" (i.e. some tokens are not assigned to any expert), or "expert unbalance" (i.e. some experts receive far more tokens than others). Unfortunately, performance can be severely impacted as a consequence. For instance, the popular Top-k or "Token Choice" router (Shazeer et al., 2017) suffers from both, while the "Expert Choice" router (Zhou et al., 2022) only suffers from the former (see Appendix B for some experiments regarding dropping in both cases). Soft MoEs are basically immune to token dropping and expert unbalance since every slot is filled with a weighted average of all tokens. All weights are (in theory) strictly positive thanks to the softmax (see Section 5 for detailed experiments).

Fast The total number of slots is the main hyperparameter that determines the cost of a Soft MoE layer. Every input applies such number of MLPs. The total number of *experts* is irrelevant in this calculation: few experts with many slots per expert or many experts with few slots per expert will have matching costs if the total number of slots is identical. The only constraint we must meet is that the number of slots has to be greater or equal to the number of experts (as each expert must process at least one slot). The main advantage of Soft MoE is completely avoiding sort or top-k operations which are slow and typically not well suited for hardware accelerators. As a result, Soft MoE is significantly *faster* than most sparse MoEs (Figure 6). See Section 2.3 for time complexity details.

Features of both sparse and dense The *sparsity* in Sparse MoEs comes from the fact that expert parameters are only applied to a subset of the input tokens. However, Soft MoEs are not technically sparse, since every slot is a weighted average of all the input tokens. Every input token *fractionally* activates all the model parameters. Likewise, all output tokens are fractionally dependent on all slots (and experts). Finally, notice also that Soft MoEs are not Dense MoEs, where every expert processes all input tokens, since every expert only processes a subset of the slots.

Per-sequence determinism Under capacity constraints, all Sparse MoE approaches route tokens in *groups* of a fixed size and enforce (or encourage) balance within the group. When groups contain tokens from different sequences or inputs, these tokens often *compete* against each other for available spots in expert buffers. As a consequence, the model is no longer deterministic at the sequence-level, but only at the batch-level, as some input sequences may affect the final prediction for other inputs. Models using larger groups tend to provide more freedom to the routing algorithm and usually perform better, while their computational cost is also higher. On the other hand, when groups contain tokens from a single sequence, the model is forced to use every expert on every input sequence. This may lead to more generalist experts. Moreover, changing the group size between training and inference can be problematic due to the potential distributional shift in token-to-expert assignments. We explore these aspects in Section 3.5.

Soft MoE gracefully sidesteps all these challenges. Since it combines all tokens in each input sequence, we just set the group size to be a single sequence. Every expert does handle tokens from every input, maybe somewhat limiting the amount of high-level specialization. Yet, this also implies that it is per-example deterministic and fast, while typical instances of Sparse MoEs are not.

2.3 Implementation

Time complexity Assume the per-token cost of a single expert function is $O(k)$. The time complexity of a Soft MoE layer is then $O(mnpd + npk)$. By choosing $p = O(m/n)$ slots per expert, i.e. the number of tokens over the number of experts, the cost reduces to $O(m^2d + mk)$. Given that each expert function has its own set of parameters, increasing the number of experts n and scaling p accordingly, allows us to increase the total number of parameters without any impact on the time complexity. Moreover, when the cost of applying an expert is large, the mk term dominates over m^2d, and the overall cost of a Soft MoE layer becomes comparable to that of applying a single expert on all the input tokens. Finally, even when m^2d is not dominated, this is the same as the (single-headed) self-attention cost, thus it does not become a bottleneck in Transformer models.

Normalization In Transformers, MoE layers are typically used to replace the feedforward layer in each encoder block. Thus, when using pre-normalization as most modern Transformer architectures (Domhan, 2018; Xiong et al., 2020; Riquelme et al., 2021; Fedus et al., 2022), the inputs to the MoE layer are "layer normalized". This causes stability issues when scaling the model dimension d, since the softmax approaches a one-hot vector as $d \to \infty$ (see Appendix E). Thus, in Line 3 of algorithm 1 we replace `X` and `Phi` with `l2_normalize(X, axis=1)` and `scale * l2_normalize(Phi, axis=0)`, respectively; where `scale` is a trainable scalar, and `l2_normalize` normalizes the corresponding axis to have unit (L2) norm, as Algorithm 2 shows.

```python
def l2_normalize(x, axis, eps=1e-6):
  norm = jnp.sqrt(jnp.square(x).sum(axis=axis, keepdims=True))
  return x * jnp.reciprocal(norm + eps)
```

Algorithm 2: JAX implementation of the L2 normalization used in Soft MoE layers.

For relatively small values of d (e.g. the model dimension used for ViT models up to ViT-H, that use $d \leq 1280$), the normalization has little impact on the model's quality. However, with the proposed normalization in the Soft MoE layer, we can eventually make the model dimension bigger and/or increase the learning rate (see Appendix E). Accordingly, we use it in all our experiments.

Distributed model When the number of experts increases significantly, it is not possible to fit the entire model in memory on a single device, especially during training or when using MoEs on top of large model backbones. In these cases, we employ the standard techniques to distribute the model across many devices, as in (Lepikhin et al., 2020; Riquelme et al., 2021; Fedus et al., 2022) and other works training large MoE models. Distributing the model typically adds an overhead in the cost of the model, which is not captured by the time complexity analysis based on FLOPs that we derived above. In order to account for this difference, in all of our experiments we measure not only the FLOPs, but also the wall-clock time in TPUv3-chip-hours.

2.4 Connections with other methods

Many existing works *merge*, *mix* or *fuse* input tokens to reduce the input sequence length (Jaegle et al., 2021; Ryoo et al., 2021; Renggli et al., 2022; Wang et al., 2022), typically using attention-like weighted averages with fixed keys, to try to alleviate the quadratic cost of self-attention with respect to the sequence length. Although our dispatch and combine weights are computed in a similar fashion to these approaches, our goal is not to reduce the sequence length (while it is possible), and we actually recover the original sequence length after weighting the experts' outputs with the *combine weights*, at the end of each Soft MoE layer.

Multi-headed attention also shows some similarities with Soft MoE, beyond the use of softmax in weighted averages: the h different *heads* can be interpreted as different (linear) experts. The distinction is that, if m is the sequence length and each input token has dimensionality d, each of the h heads processes m vectors of size d/h. The m resulting vectors are combined using different weights for each of the m' output tokens (i.e. the attention weights), on each head independently, and then the resulting (d/h)-dimensional vectors from each head are concatenated into one of dimension d. Our experts are non-linear and combine vectors of size d, at the *input and output* of such experts.

Finally, there are also connections with other MoE works that use a weighted combination of the experts parameters, rather than doing a sparse routing of the examples (Yang et al., 2019; Tian et al., 2020; Muqeeth et al., 2023). These approaches are also fully differentiable, although they can have a much higher cost, since 1) they must average the parameters of the experts, which can become a time and/or memory bottleneck when experts with many parameters are used; and 2) they cannot take advantage of vectorized operations as broadly as Soft (and Sparse) MoEs, since *every input uses a different weighted combination of the parameters*. We recommend the "computational cost" discussion in (Muqeeth et al., 2023) that addresses these issues.

2.5 Current limitations

Auto-regressive decoding One of the key aspects of Soft MoE consists in smartly merging all tokens in the input. This makes the use of Soft MoEs in auto-regressive decoders difficult, since causality between past and future tokens has to be preserved during training. Although causal masks used in attention layers could be used, one must be careful to not introduce any correlation between token and slot *indices*, since this would bias which token indices each expert is trained on. The use of Soft MoE in auto-regressive decoders is a promising research avenue that we leave for future works.

Lazy experts & memory consumption We extensively show in Section 3 that one slot per expert tends to be the optimal choice. In other words, rather than feeding one expert with two slots (or mixes of tokens), it is more effective from a performance standpoint to use two experts with one slot each. We hypothesize same-expert slots tend to somewhat align and provide small informational gains, and single experts may lack the flexibility to accommodate very different slot projections. We show this in Appendix H. Consequently, Soft MoE tends to leverage a large number of experts and –while its cost is still similar to the dense backbone– the memory requirements of the model can grow large.

3 Image Classification Experiments

We present three types of experiments on image classification:

Training Pareto frontiers. First, in Section 3.3 we systematically compare dense ViT models at the Small, Base, Large and Huge sizes with their sparse counterparts based on the most common routing techniques (Tokens Choice, Experts Choice) and Soft MoE routing. We study the training FLOPs versus performance and training time versus performance plots to conclude that Soft MoE dominates all other approaches.

Inference-time optimized models. Second, in Section 3.4, we present longer training runs ("overtraining"). Relative to ViT, Soft MoE brings large improvements in terms of inference speed (small models: S, B) and absolute performance (large models: L, H).

Model ablations. Third, in Section 3.5 we investigate some of the central aspects of Soft MoE routing (such as number of experts, slots per expert, etc), and compare their behavior with other routing algorithms. We present the optimal configurations for Soft MoE and the source of the improvement benefits.

3.1 Training and evaluation data

We pretrain our models on JFT-4B (Sun et al., 2017), a proprietary dataset whose latest version contains more than 4B images, covering more than 29k classes. During pretraining, we typically evaluate the models on two metrics: upstream validation precision-at-1 on JFT-4B, and ImageNet 10-shot accuracy. The latter is computed by freezing the model weights and replacing the head with a new one that is only trained on a dataset containing 10 images per class from ImageNet-1k (Deng et al., 2009). Finally, we provide the accuracy on the validation set of ImageNet-1k after finetuning on the training set of ImageNet-1k (1.3 million images).

3.2 Sparse routing algorithms

We compare to the following popular MoE routing algorithms:

Tokens Choice. Every token selects the top-K experts with the highest routing score for the token (Shazeer et al., 2017). Increasing K typically leads to better performance at the expense of extra computational cost. Batch Priority Routing (BPR) (Riquelme et al., 2021) significantly improves the model performance, especially in the case of $K = 1$ (see Appendix, Table 8). Accordingly we use Top-K routing with BPR and $K \in \{1, 2\}$. We also optimize the number of experts (Appendix, Figure 15).

Experts Choice. Alternatively, experts can select the top-C tokens in terms of routing scores (Zhou et al., 2022). In this case, C is the buffer size, and we typically set $E \cdot C = c \cdot T$ where E is the number of experts, T is the total number of tokens in the group, and c is the capacity multiplier. When $c = 1$, all tokens can be served via the union of experts. Note that in this type of routing, it is common that some tokens are simultaneously selected by several experts whereas some other tokens are not selected at all. Figure 14 illustrates this phenomenon. We experiment with $c = 0.5, c = 1$ and $c = 2$.

3.3 Training Pareto-optimal models

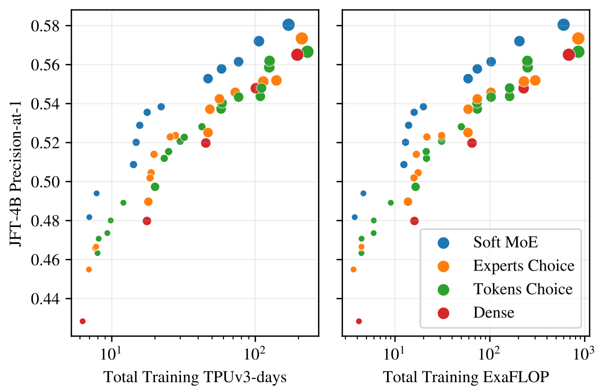
We train VIT-S/8, VIT-S/16, VIT-S/32, VIT-B/16, VIT-B/32, VIT-L/16, VIT-L/32 and VIT-H/14 models, and their sparse counterparts. We consider three routing algorithms for the sparse models: Token Choice, Expert Choice, and Soft MoE. In each case, we train several model variants (different K, C and number of experts where it corresponds). In total, we train 106 models. The models are trained for 300k steps with batch size 4096 in all cases, and inverse square root learning rate decay.

Figure 3a and Figure 3b show the results for models in each class that lie on their respective training cost / performance Pareto frontiers. On both metrics, Soft MoE strongly outperforms dense and other sparse approaches for any given FLOPs or time budget. Table 9 in Appendix I list all the models, with their parameters, performance and costs, and are shown in Figure 22. Figures 23 to 25 in Appendix F compare Soft MoE individually to Dense, Token Choice and Expert Choice models respectively.

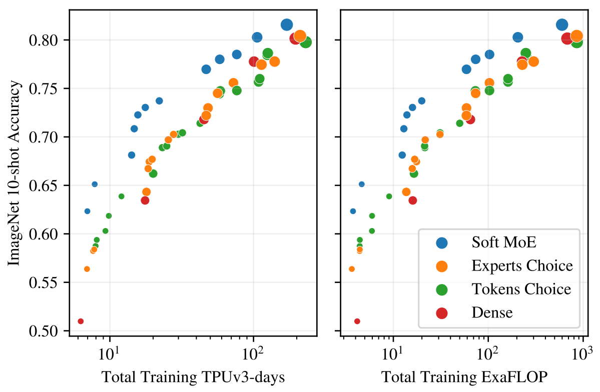
3.4 Long training runs

In addition to shorter runs and ablations, we trained a number of models for much longer (a few million steps) to test the Soft MoE capabilities at larger computational scales. We present two experiments.

First, in Section 3.4.1, we trained ViT and Soft MoE of different sizes ranging from Small to Huge for 4M steps. Figure 4 and Table 2 show the results. Second, in Section 3.4.2, we kept training some of the previous Soft MoE models beyond the optimal point suggested by standard *dense* scaling laws. Sparse models can leverage the extra capacity to steadily improve their performance, leading to very strong Soft MoE models that are notably cheap at inference.



(a) JFT-4B Precision-at-1



(b) ImageNet 10-shot Accuracy

Figure 3: **Pareto Models.** Soft MoE dominates both ViTs (dense) and popular MoEs (Experts Choice,Tokens Choice) on the training cost / performance Pareto frontier. Each point is a model trained for 300k steps, and larger marker sizes indicate larger models: S/32, S/16, B/32, B/16, L/16 and H/14. Cost is shown both in terms of FLOPS and realized TPU-v3 training time. MoE runs include different configuration; for clarity, only models on their respective Pareto frontier are displayed. Figure 22 in Appendix F shows all models.

3.4.1 Comparison with large-scale Vision Transformers

We trained a number of Soft MoEs on JFT, following a comparable setting to that used by Zhai et al. (2022a). We replace the last half of the blocks in ViT S/16, B/16, L/16, and H/14 with Soft MoE layers with 128 experts, using one slot per expert. We train models ranging from 1B to 54B parameters. Large Soft MoE models incur in a small wall-clock time overhead compared to their dense counterparts due to the extra data transfers required by model parallelism. All variants were trained for 4M steps, except for H/14s which was trained for 2M steps for cost reasons.

Figure 4 shows the JFT-4B precision, ImageNet 10-shot accuracy, and the ImageNet finetuning accuracy for Soft MoE and ViT versus training cost in ExaFLOPS. Table 2 contains all the results, and Figure 19 shows performance versus core-hours. Soft MoE models widely outperform Vision Transformer models for a given compute budget. For instance, the Soft MoE S/16 performs better than ViT B/16 on JFT and 10-shot ImageNet, and it also improves finetuning scores on the full ImageNet data, even though its training (and inference) cost is significantly smaller. Similarly, Soft MoE B/16 outperforms ViT L/16 upstream, and only lags 0.5 behind after finetuning while being 3x faster and requiring almost 4x fewer FLOPs. Finally, the Soft MoE L/16 model outperforms the dense H/14 one while again being around 3x faster to train and serve at inference.

3.4.2 Soft MoEs optimized for inference

Encouraged by the fact that Soft MoEs with smaller backbones can match the quality of larger Vision Transformers, we continue training the small backbones to obtain models of higher quality at very low inference cost. Even after additional (over) training, the overall training time with respect to larger ViT models is comparable. For these long runs, we observe that longer cooldowns (period where the learning rate is decreased linearly to zero (Zhai et al., 2022a)) work well for Soft MoE. Therefore, we increase the cooldown from 50k steps (used elsewhere) to up to 500k steps. Figure 5 presents these models.

We now summarize our main results. Soft MoE B/16 trained for 1k TPUv3 days outperforms ViT H/14 trained on a similar time budget (see Table 1, ViT H/14, 1M steps) while being **10$\times$ cheaper at inference** in FLOPs and 5.7$\times$ in wall-clock time, and it almost matches the ViT H/14 model performance even if we **double** ViT-H/14's training budget (2M steps and 2039.8 train days for ViT H/14 versus 1011.4 days for Soft MoE B/16). Soft MoE L/16 beats all models substantially while being almost 2$\times$ faster at inference than ViT H/14.

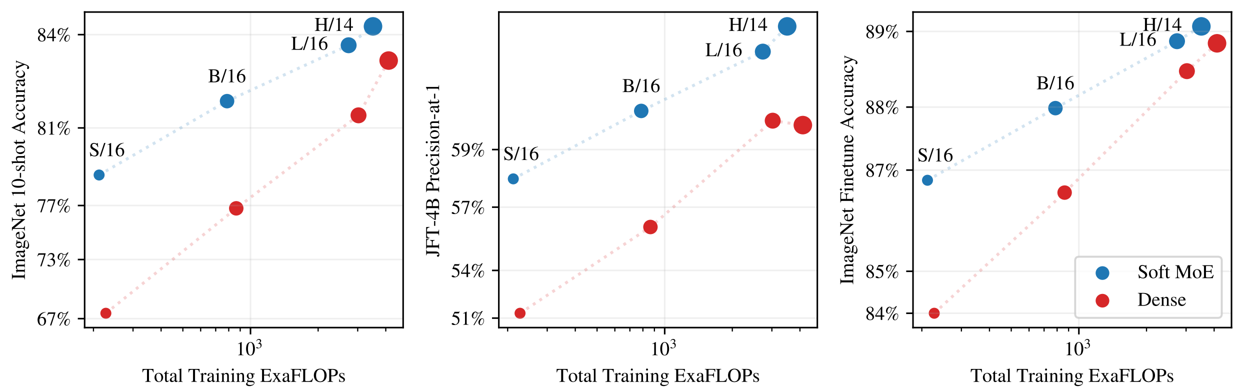


Figure 4: **Long runs.** Soft MoE and ViT models trained for 4 million steps with batch size 4096 (H/14 models trained for 2 million steps instead). Equivalent model classes (S/16, B/16, L/16, H/14) have similar training costs, but Soft MoE outperforms ViT on all metrics. We show ImageNet 10-shot (left), JFT precision at 1 (middle) and ImageNet accuracy after finetuning (right), versus total training FLOPs. See Table 2. We report training wall-clock time in Figure 19.

Table 1: Training and finetuning results for Soft MoE and dense models. Finetuning performed on ImageNet at 384 resolution. Steps used for linear cooldown indicated in parentheses, these are included in the total train steps count. Results are plotted in Figure 5.

Model	Params	Train steps	Train days	& exaFLOP	Eval Ms/img	& GFLOP/img	JFT P@1	IN/10shot	IN/ft
ViT S/16	33M	4M (50k)	153.5	227.1	0.5	9.2	51.3	67.6	84.0
ViT B/16	108M	4M (50k)	410.1	864.1	1.3	35.1	56.2	76.8	86.6
ViT L/16	333M	4M (50k)	1290.1	3025.4	4.9	122.9	59.8	81.5	88.5
ViT H/14	669M	2M (50k)	2039.8	4120.3	8.6	334.2	59.7	83.3	88.9
Soft MoE S/14 256E	1.8B	10M (50k)	494.7	814.2	0.9	13.2	60.1	80.6	87.5
Soft MoE B/16 128E	3.7B	9M (500k)	1011.4	1769.5	1.5	32.0	62.4	82.9	88.5
Soft MoE L/16 128E	13.1B	4M (500k)	1355.4	2734.1	4.8	111.1	63.0	84.3	89.2

3.5 Soft MoE Ablations

Here we establish the optimal configurations for Soft MoE models by exploring the following:

Optimal number of slots per expert. One or two slots per expert work best. We demonstrate this by fixing the total number of slots (which determines the compute cost of the model), and changing the number of experts, i.e. the slots per expert (Figure 6).

Optimal number of experts. Roughly the same number of experts as input tokens work best when using one slot per expert. The model is then similarly expensive in terms of FLOPs as its dense equivalent. To show this, we increase the number of experts and train models for the same amount of time, and find the best performing model (Figure 8).

Architectural/algorithmic ablations. To disentangle the source of the benefits, we compare Soft MoE to a number of ablated versions: route token i deterministically to expert i, fixed uniform dispatch/combine weights, and others (TablTable 3).

MoE layers placement. An additional ablation regarding where to place MoE layers is presented in Appendix D.

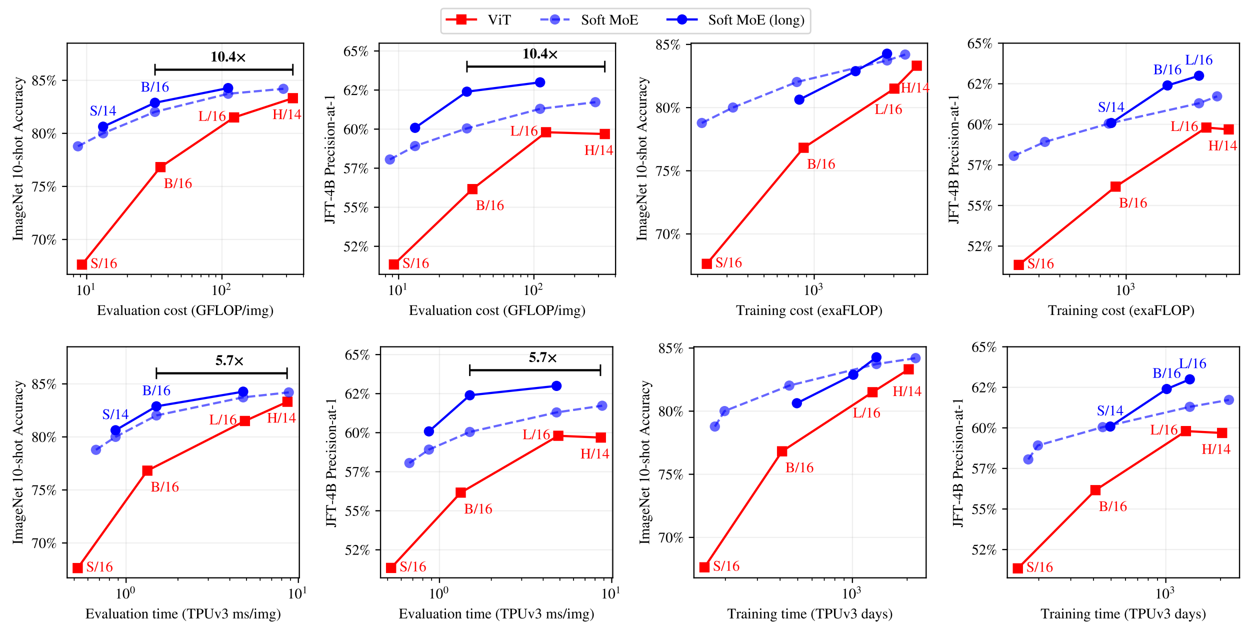


Figure 5: **Soft MoE optimized for inference.** These plots show the quality on JFT-4B (Precision-at-1) and ImageNet (10-shot Accuracy) achieved by different models with different training and inference cost (measured both in TPUv3 time and FLOPs). Red and light blue curves correspond (respectively) to ViT and Soft MoE S/16, S/14, B/16, L/16 and H/14 trained for 4 million steps (except H/14, that was trained for 2 million steps), following a recipe similar to (Zhai et al., 2022a). Dark blue curves correspond to Soft MoE S/14, B/16, L/16 trained for additional steps as detailed in Table 1. We observe that the overtrained Soft MoE B/16 is better than the best ViT model (H/14) while using 10× less computation (5.7× time). Soft MoE L/16 is the most performant model requiring one third of the inference FLOPs (one half of the time). Detailed results in Tables 1 and 2.

3.5.1 Number of Experts and Slots per Expert

When applying Soft MoE to a given architecture and input sequence length, one must decide how many experts and how many slots per expert to use. The total number of slots determines the amount of work (FLOPs) applied in the MoE layer (ignoring the small the routing cost). If the total number of slots is greater than the number of input tokens, the model will require more FLOPs than dense Transformers: more "tokens" will be processed in the MoE layer. Conversely, if the number of slots is lower than the original number of tokens, Soft MoE will save some compute relative to dense Transformers.

Unless stated otherwise, the following experiments use a ViT-S/16 backbone trained for 300k steps with batch size 4096. The MoEs have expert layers in their last six of twelve blocks.

Optimal number of slots per expert. In this experiment the total amount of compute is fixed, and we compare different configurations. Specifically, we fix the total number of slots to 4096, and we train models with different number of experts. MoE algorithms are often unable to scale well to a large number of experts (over 100). The model sizes range from just 38M (with 2 experts) to 9.7B parameters (when using 4096 experts). Figure 6 (and Figure 26) shows the results in terms of pre-training precision (left) and the few-shot evaluation (middle). In the case of Experts Choice and Tokens Choice MoEs, the size of the union of all expert buffers is also 4096 per input image. We just vary the number of experts keeping the total number of tokens processed across the union of experts constant, as for Soft MoE. For the Sparse MoEs (Experts/Tokens Choice), there is an implementation detail: The "group size" is the subset of the batch that is routed together. All tokens in a group compete to be selected by each expert. This can range from one image/group to the entire batch/group; the latter is more flexible, but increases computational overhead in routing (sorting the

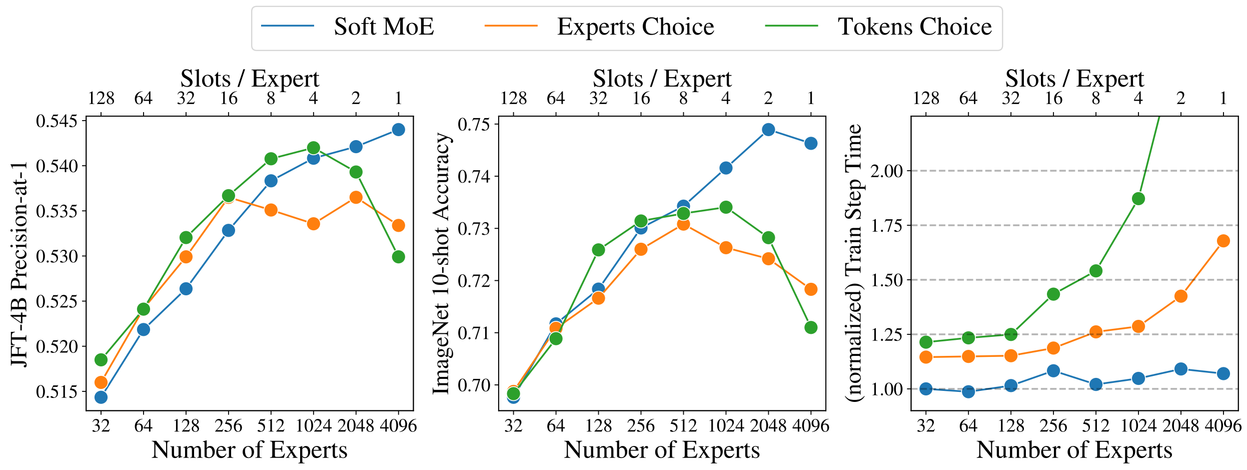


Figure 6: **Performance (left, center), and training step time (right) as a function of number of experts, for models with a fixed number of slots (Soft MoE) or expert buffer capacity (Sparse MoEs)** on a ViT-S/16 backbone with MoEs in the last two layers. Soft MoE achieves much better scaling with more experts, while cost is roughly constant. However, with Experts and Tokens Choice routers, having too many experts not only hurts performance but also significantly increases the cost (Tokens Choice reaches 3.9x with 4096 experts).

items). In Figure 6, we use group size eight. Figure 20, Appendix, shows other options.

Figure 6 shows that Soft MoE scales with increased experts. The best configurations are 2048 and 4096 experts, at one/two slots per experts, respectively. In contrast, Experts Choice and Tokens Choice do not scale well with the number of experts, and performance degrades after 512 experts. In addition, Figure 6, right, shows the step time for each model. Due to sorting leading to increased computational overhead, the Sparse MoE's step time increases substantially with more experts, which is not the case for Soft MoE.

Optimal number of experts. From the previous analysis, we set the number of slots per expert to one. The next question is how many experts to use. Here, the cost of models are *not* matched: more experts will increase cost (through more slots). Figure 7 shows that, both for Soft MoE and Experts Choice, more experts do better (up to 1024).

Next, we match the total training time for each model by adjusting the number of training steps (Figure 8). At this scale (ViT-S), the optimal number of experts for a given training budget is around 128 or 256 experts. The number of input tokens is 196, this corresponds to the minimum number of experts that does not lead to a strong token bottleneck (many fewer than 196 slots) in the MoE layer. For any number of experts, Soft MoE outperforms Experts Choice. Both models have the same capacity, but Experts Choice is significantly more expensive, especially with large group size.

More slots per expert. Appendix C explores how Soft MoE behaves when increasing the number of slots per expert. Appendix H looks at the (strong) correlation between the learned slot parameters in this case.

3.5.2 Algorithmic Ablations: Identity & Uniform Routing

Soft MoE relies on learning how to mix tokens for each expert. To understand the impact of finding useful linear combinations of input tokens, we ablate this aspect by testing some natural choices:

Identity routing. Tokens are not mixed: the first token goes to first expert, second token goes to second expert, etc.

Uniform Mixing. Every slot mixes all input tokens in the same way: by uniformly averaging them, both for dispatching and combining. In this case, we must independently and randomly initialize every expert as

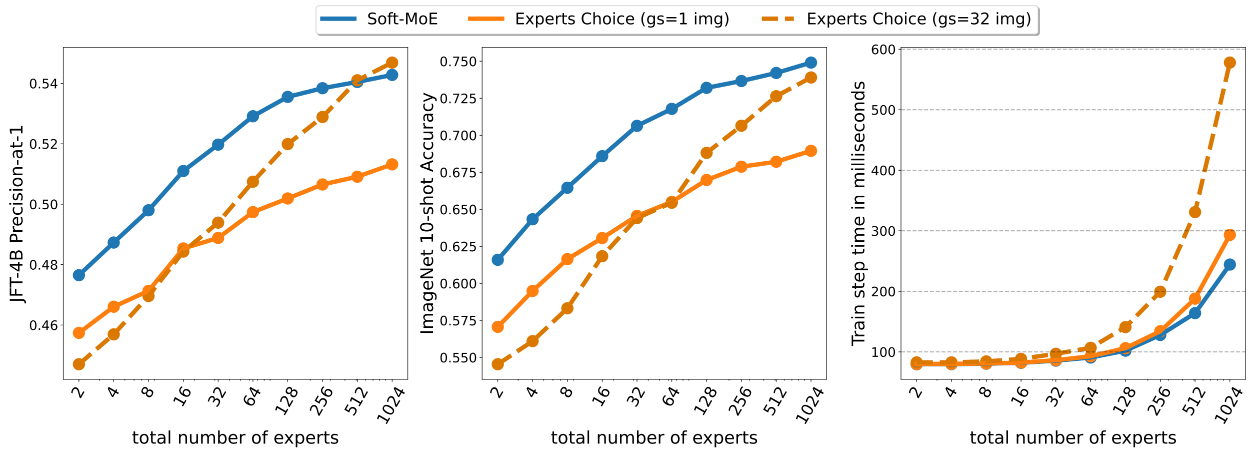


Figure 7: **Performance (left, center) and step time (right) for models trained with increased experts and one slot (or token) per expert for a fixed number of steps (300k).** The performance of all models improves as their capacity increases. However, the cost of Experts Choice grows faster than that of Soft MoE, especially when the group size is larger (gs= 32).

otherwise the additional capacity coming from different experts will not be used (we end up with copies).

Soft / Uniform. We learn to mix tokens to create the slots (dispatch weights), but we uniformly average all expert outputs. This implies every input token is identically updated before the residual connection.

Uniform / Soft. All slots are filled with the uniform average of the input tokens. We learn to mix the expert output tokens depending on the input tokens.

Table 3 summarizes our results, and Appendix A contains further details. Learning to mix tokens for dispatching and for combining tokens after expert processing seems essential to perform well, and dispatch mixing is slightly more important than the combine mixing. Dense underperform all variants.

4 Contrastive learning experiments

We test whether the learned representations are also significantly better when used for other tasks. In this section we explore a popular paradigm, image-language contrastive learning. We follow the approach in Zhai et al. (2022b) where the image tower is pre-trained on an image classification task, and then frozen while training the text encoder on a dataset of image-text pairs.

We re-use the models trained on JFT in the previous section and compare their performance on a number of downstream applications. For contrastive learning we train on WebLI (Chen et al., 2022), a proprietary dataset consisting of 10B images and their ALT texts crawled from the internet. The image encoder is frozen, while the text encoder is trained from scratch.

Table 4 shows our results. Overall, the gaps we observed on image classification are preserved in this setting. For instance, Soft MoE-L/16 outperforms ViT-L/16 by more than 1% and 2% on Imagenet and Cifar-100 zero-shot, respectively. Retrieval numbers are generally modest.

5 Model Inspection

In this section, we take a look at various aspects of the routing the model learns.

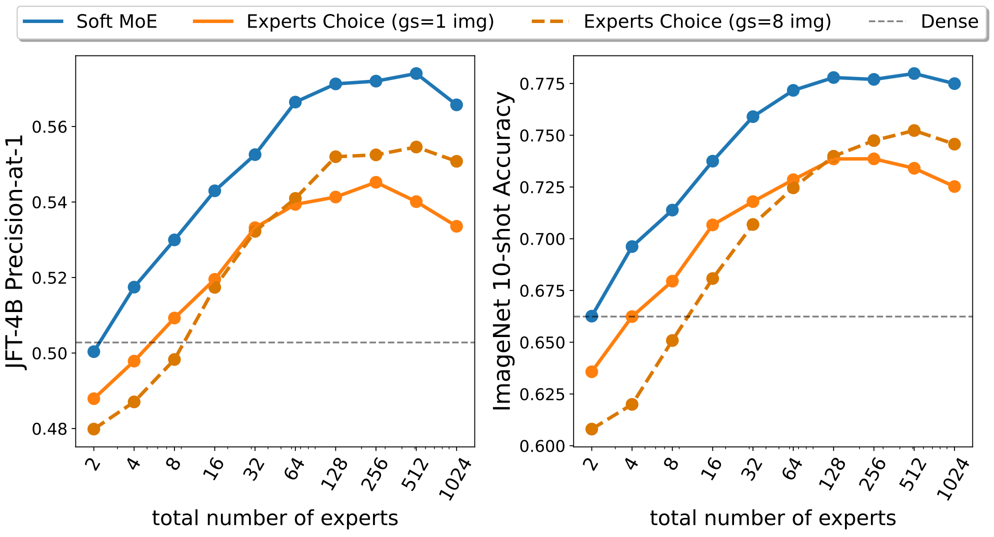


Figure 8: **Performance of models trained with increasing experts (one slot/token per expert), with matched training duration.** The total number of steps in each case is computed to match the total training time of 300k steps for 1024-expert Experts Choice with 32 images per group. For context, the dashed line corresponds to Dense ViT-S/16. Here, Soft MoE outperforms Experts Choice at all capacities, and the optimum point is at around 512 experts.

Tokens contributions to slots. While there is no dropping in Soft MoE, it is still possible that some tokens contribute little to *all* slots if their logits are much lower than those of other tokens. We would like to see if some tokens contribute to slots in a disproportionate manner. Figure 9 (left) shows the distribution across tokens for the total weight each token provides to slots (i.e. summed over all slots). This was computed over a batch with 256 images with 196 tokens each on a Soft MoE S/16 finetuned on ImageNet. We see there is a heavy tail of tokens that provide a stronger total contribution to slots, and the shape is somewhat similar across layers. Around 2-5% of the tokens provide a summed weight above 2. Also, between 15% and 20% of the tokens only contribute up to 0.25 in total weight. The last layer is slightly different, where token contribution is softer tailed. Appendix G further explores this.

Experts contributions to outputs. Similarly, we would like to understand how much different slots end up contributing to the output tokens. We focus on the case of one slot per expert. We can approximate the total contribution of each expert (equivalently, slot) by averaging their corresponding coefficients in the linear combinations for all output tokens in a batch. Figure 9 (center) shows such (normalized) importance across experts for different MoE layers. We see that, depending on the layer, some experts can impact output tokens between 3x and 14x more than others.

Number of input tokens per slot. For each slot, Figure 9 (right) shows how many input tokens are required to achieve a certain cumulative weight in its linear combination. The distribution varies significantly across slots. For a few slots the top 20-25 tokens account for 90% of the slot weight, while for other slots the distribution is more uniform and many tokens contribute to fill in the slot. In general, we see that slots tend to mix a large number of tokens unlike in standard Sparse MoEs.

Visual inspection. In order to provide some intuition regarding how slots average input tokens, Figure 10 graphically shows the linear combinations for 8 different slots for the image shown in Figure 1. We shade patches inversely proportionally to their weight in the slots; note that all tokens representations are eventually combined into a single one (with hidden dimension h) before being passed to the expert (unlike in our plot, where they are arranged in the usual way). These plots correspond to a Soft MoE S/16 with 128 experts and one slot per expert, and we handpicked 8 out of the 128 slots to highlight how different slots tend to focus on different elements of the image.

Table 2: Training and finetuning results for Soft MoE and dense models. Finetuning results on ImageNet at 384 resolution. We use one slot per expert and did not increase this number during finetuning, thus Soft MoEs become cheaper than ViT, as the number of input tokens grows to 576 (patch size 16x16) and 752 (patch size 14x14) but the number slots is fixed to a much smaller number (either 128 or 256).

Model	Params	Train steps	Train days	& exaFLOP	Eval Ms/img	& GFLOP/img	JFT P@1	IN/10s	IN/ft
ViT S/16	33M	4M (50k)	153.5	227.1	0.5	9.2	51.3	67.6	84.0
Soft MoE S/16 128E	933M	4M (50k)	175.1	211.9	0.7	8.6	58.1	78.8	86.8
Soft MoE S/16 128E	933M	10M (50k)	437.7	529.8	0.7	8.6	59.2	79.8	87.1
Soft MoE S/14 256E	1.8B	4M (50k)	197.9	325.7	0.9	13.2	58.9	80.0	87.2
Soft MoE S/14 256E	1.8B	10M (500k)	494.7	814.2	0.9	13.2	60.9	80.7	87.7
ViT B/16	108M	4M (50k)	410.1	864.1	1.3	35.1	56.2	76.8	86.6
Soft MoE B/16 128E	3.7B	4M (50k)	449.5	786.4	1.5	32.0	60.0	82.0	88.0
ViT L/16	333M	4M (50k)	1290.1	3025.4	4.9	122.9	59.8	81.5	88.5
Soft MoE L/16 128E	13.1B	1M (50k)	338.9	683.5	4.8	111.1	60.2	82.9	88.4
Soft MoE L/16 128E	13.1B	2M (50k)	677.7	1367.0	4.8	111.1	61.3	83.3	88.9
Soft MoE L/16 128E	13.1B	4M (50k)	1355.4	2734.1	4.8	111.1	61.3	83.7	88.9
ViT H/14	669M	1M (50k)	1019.9	2060.2	8.6	334.2	58.8	82.7	88.6
ViT H/14	669M	2M (50k)	2039.8	4120.3	8.6	334.2	59.7	83.3	88.9
Soft MoE H/14 128E	27.3B	1M (50k)	1112.7	1754.6	8.8	284.6	61.0	83.7	88.9
Soft MoE H/14 128E	27.3B	2M (50k)	2225.4	3509.2	8.8	284.6	61.7	84.2	89.1
Soft MoE H/14 256E	54.1B	1M (50k)	1276.9	2110.1	10.9	342.4	60.8	83.6	88.9
Soft MoE H/14 256E	54.1B	2M (50k)	2553.7	4220.3	10.9	342.4	62.1	84.3	89.1

Table 3: Algorithmic ablation on an S/14 backbone trained for 300k steps (with 256 experts).

Method	Experts	Mixing	Learned Dispatch	Learned Combine	JFT p@1	IN/10shot
Soft MoE	✓	✓	✓	✓	54.3%	74.8%
Soft / Uniform	✓	✓	✓		53.6%	72.0%
Uniform / Soft	✓	✓		✓	52.6%	71.8%
Uniform	✓	✓			51.8%	70.0%
Identity	✓				51.5%	69.1%
Dense ViT					48.3%	62.3%

Table 4: LIT-style evaluation with a ViT-g text tower trained for 18B input images (~ 5 epochs).

Model	Experts	IN/0shot	Cifar100/0shot	Pet/0shot	Coco Img2Text	Coco Text2Img
ViT-S/16	–	74.2%	56.6%	94.8%	53.6%	37.0%
Soft MoE-S/16	128	81.2%	67.2%	96.6%	56.0%	39.0%
Soft MoE-S/14	256	82.0%	75.1%	97.1%	56.5%	39.4%
ViT-B/16	–	79.6%	71.0%	96.4%	58.2%	41.5%
Soft MoE-B/16	128	82.5%	74.4%	97.6%	58.3%	41.6%
ViT-L/16	–	82.7%	77.5%	97.1%	60.7%	43.3%
Soft MoE-L/16	128	83.8%	79.9%	97.3%	60.9%	43.4%
Souped Soft MoE-L/16	128	84.3%	81.3%	97.2%	61.1%	44.5%
ViT-H/14	–	83.8%	84.7%	97.5%	62.7%	45.2%
Soft MoE-H/14	256	84.6%	86.3%	97.4%	61.0%	44.8%

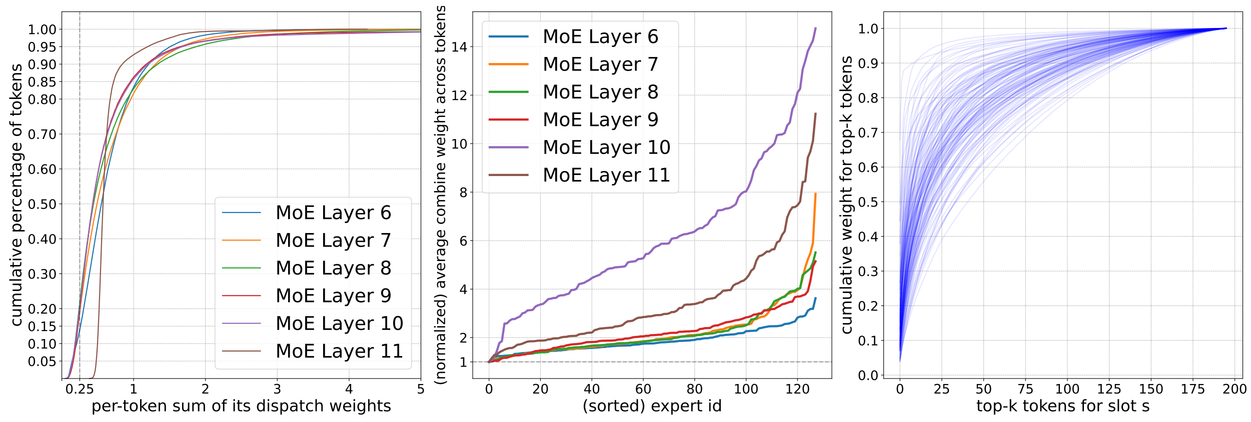


Figure 9: (**Left**) Distribution of summed dispatch weights per token for different MoE layers. For instance, in layer 11, the dispatch weights for 90-95% of the input tokens summed over all the slots are at most 1. Only a tiny fraction of tokens contribute to slots by summing more than 3. (**Middle**) Distribution of combine weights per slot (or expert, as we use one slot per expert) summed across all input tokens. We normalize the sum by its minimum value across experts. (**Right**) Each curve corresponds to one slot. Dispatch weights from all tokens to each slot add up to 1. Distribution of how many inputs tokens are needed to achieve a certain fraction of the total weight for the slot.

6 Discussion

Sparse models can face infrastructural challenges which may have slowed down their broad adoption. Since these models were originally conceived to unlock massive model sizes, they tend to be distributed and most routing algorithms require additional communication costs: additional activations, gradients, or expert parameters are sent across devices. This is also true for Soft MoEs, where the experts may also be distributed. However, modern dense models are now sufficiently large that they are also distributed, thus closing the gap in this axis. In addition, the benefits of sparsity shine at small model scales, both in prior work (Riquelme et al., 2021) and with Soft MoE, fitting with the current needs of the industry for faster inference.

We presented Soft MoE, a new sparse Transformer architecture that avoids the discrete token-to-expert assignment problem that is common in sparse mixture of experts models. By merging input tokens into linear combinations before dispatching them to experts, we are able to train a fast and fully-differentiable model. We perform extensive image-classification and image-language contrastive learning experiments comparing the performance of dense models and several sparse methods (Tokens Choice, Experts Choice, Soft MoE). These experiments suggest Soft MoE is surprisingly effective and strongly outperforms the other approaches while often being computationally cheaper. How to deal with causal masking for language decoders is an exciting and impactful research direction for future work.

Acknowledgements

We thank Rodolphe Jenatton, who provided extremely valuable feedback on an earlier version of this manuscript; Ilya Tolstikhin, who suggested the “Identity router” used in Appendix A (or “Liquid router”, as he dubbed it); and the rest of Google DeepMind folks for providing a supportive research environment, very especially to our colleagues in Europe.

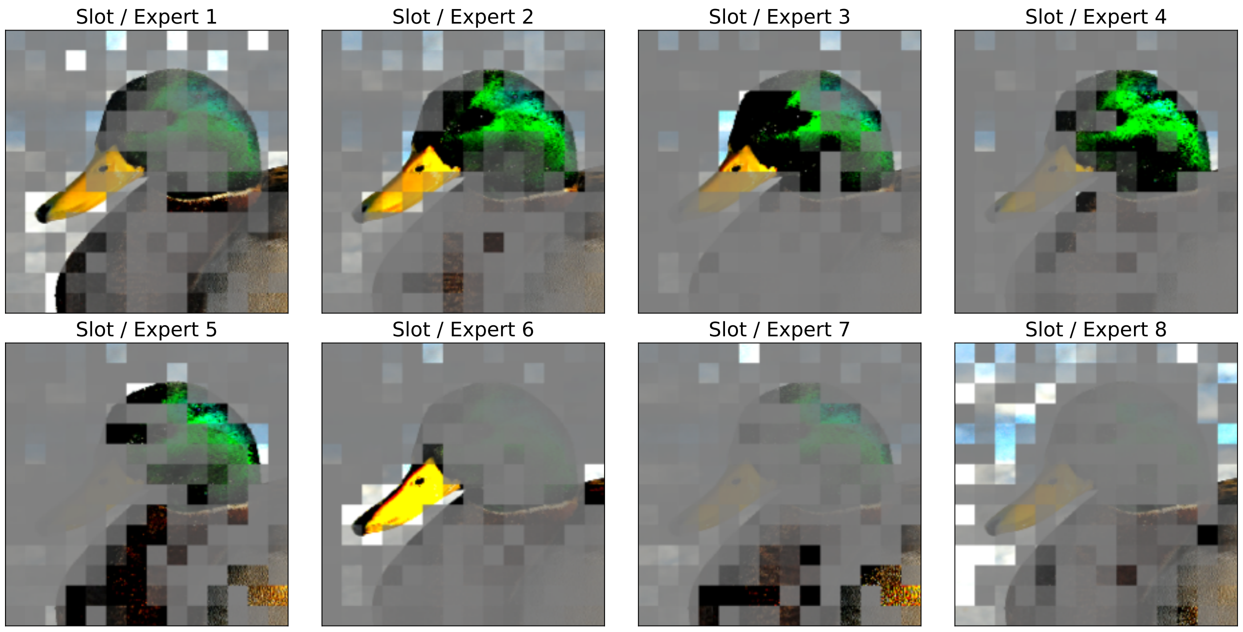


Figure 10: Linear combinations for 8 slots when using input image in Figure 1. Model is Soft MoE S/16 with 128 experts and one slot per expert, and it was finetuned on ImageNet. We show results for the first MoE layer (seventh block). The selected slots (among 128) are cherry-picked to highlight differences across slots.

References

Emmanuel Bengio, Pierre-Luc Bacon, Joelle Pineau, and Doina Precup. Conditional computation in neural networks for faster models. *arXiv preprint arXiv:1511.06297*, 2015.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, et al. JAX: composable transformations of Python+ NumPy programs, 2018.

Xi Chen, Xiao Wang, Soravit Changpinyo, AJ Piergiovanni, Piotr Padlewski, Daniel Salz, Sebastian Goodman, Adam Grycner, Basil Mustafa, Lucas Beyer, et al. Pali: A jointly-scaled multilingual language-image model. *arXiv preprint arXiv:2209.06794*, 2022.

Aidan Clark, Diego De Las Casas, Aurelia Guy, Arthur Mensch, Michela Paganini, Jordan Hoffmann, Bogdan Damoc, Blake Hechtman, Trevor Cai, Sebastian Borgeaud, et al. Unified scaling laws for routed language models. In *International Conference on Machine Learning*, pages 4057–4086. PMLR, 2022.

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 conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

Tobias Domhan. How much attention do you need? a granular analysis of neural machine translation architectures. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1799–1808, 2018.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *The Journal of Machine Learning Research*, 23(1):5232–5270, 2022.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

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

Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. Perceiver: General perception with iterative attention. In *International conference on machine learning*, pages 4651–4664. PMLR, 2021.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. *Advances in neural information processing systems*, 30, 2017.

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. *arXiv preprint arXiv:2006.16668*, 2020.

Mike Lewis, Shruti Bhosale, Tim Dettmers, Naman Goyal, and Luke Zettlemoyer. Base layers: Simplifying training of large, sparse models. In *International Conference on Machine Learning*, pages 6265–6274. PMLR, 2021.

Tianlin Liu, Joan Puigcerver, and Mathieu Blondel. Sparsity-constrained optimal transport. *arXiv preprint arXiv:2209.15466*, 2022.

Mohammed Muqeeth, Haokun Liu, and Colin Raffel. Soft merging of experts with adaptive routing, 2023.

Basil Mustafa, Carlos Riquelme, Joan Puigcerver, Rodolphe Jenatton, and Neil Houlsby. Multimodal contrastive learning with limoe: the language-image mixture of experts. *arXiv preprint arXiv:2206.02770*, 2022.

Cedric Renggli, André Susano Pinto, Neil Houlsby, Basil Mustafa, Joan Puigcerver, and Carlos Riquelme. Learning to merge tokens in vision transformers. *arXiv preprint arXiv:2202.12015*, 2022.

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. *Advances in Neural Information Processing Systems*, 34:8583–8595, 2021.

Stephen Roller, Sainbayar Sukhbaatar, Jason Weston, et al. Hash layers for large sparse models. *Advances in Neural Information Processing Systems*, 34:17555–17566, 2021.

Michael S Ryoo, AJ Piergiovanni, Anurag Arnab, Mostafa Dehghani, and Anelia Angelova. Tokenlearner: What can 8 learned tokens do for images and videos? *arXiv preprint arXiv:2106.11297*, 2021.

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. *arXiv preprint arXiv:1701.06538*, 2017.

Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE international conference on computer vision*, pages 843–852, 2017.

Zhi Tian, Chunhua Shen, and Hao Chen. Conditional convolutions for instance segmentation. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*, pages 282–298. Springer, 2020.

Yikai Wang, Xinghao Chen, Lele Cao, Wenbing Huang, Fuchun Sun, and Yunhe Wang. Multimodal token fusion for vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12186–12195, June 2022.

Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tieyan Liu. On layer normalization in the transformer architecture. In *International Conference on Machine Learning*, pages 10524–10533. PMLR, 2020.

Brandon Yang, Gabriel Bender, Quoc V Le, and Jiquan Ngiam. Condconv: Conditionally parameterized convolutions for efficient inference. *Advances in Neural Information Processing Systems*, 32, 2019.

Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12104–12113, 2022a.

Xiaohua Zhai, Xiao Wang, Basil Mustafa, Andreas Steiner, Daniel Keysers, Alexander Kolesnikov, and Lucas Beyer. Lit: Zero-shot transfer with locked-image text tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18123–18133, 2022b.

Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M Dai, Quoc V Le, James Laudon, et al. Mixture-of-experts with expert choice routing. *Advances in Neural Information Processing Systems*, 35:7103–7114, 2022.

A Soft vs. Uniform vs. Identity dispatch and combine weights

In this section, we compare Soft MoE (i.e. the algorithm that uses the dispatch and combine weights computed by Soft MoE in eq. (1) and eq. (3)) with different "fixed routing" alternatives, where neither the expert selected nor the weight of the convex combinations depend on the *content* of the tokens.

We consider the following simple modifications of Soft MoE:

Identity. The first token in the sequence is processed by the first expert, the second token by the second expert, and so on in a round robin fashion. When the sequence length is the same as the number of slots and experts, this is equivalent to replacing the matrix **D** in eq. (1) (resp. **C** in eq. (3)) with an identity matrix.

Uniform. Every input slot is filled with a uniform average of all input tokens, and every output token is a uniform average of all output slots. This is equivalent to replacing the matrix **D** from eq. (1) with values $\frac{1}{m}$ in all elements, and a matrix **C** from eq. (3) with values $\frac{1}{np}$ in all elements. We randomly and independently initialize every expert.

Uniform / Soft. Every input slot is filled with a uniform average of all input tokens, but we keep the definition of **C** from eq. (3).

Soft / Uniform. Every output token is a uniform average of all output slots, but we keep the definition of **D** in eq. (1).

Figure 11 and Table 3 shows the results from this experiment, training a S/14 backbone model with MoEs on the last 6 layers. Since the sequence length is 256, we choose 256 experts and slots (i.e. 1 slot per expert), so that the matrices **D** and **C** are squared. As shown in the figure, Soft MoE is far better than all the other alternatives. For context, we also add the dense ViT S/14 to the comparison.

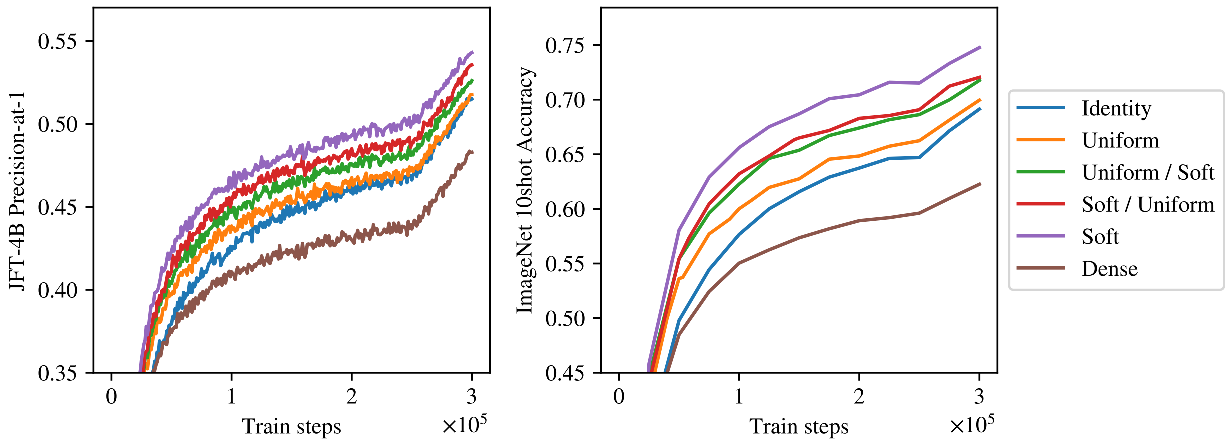


Figure 11: **Soft MoE compared against different "fixed routing" strategies**. *Identity* processes the i-th token with the i-th expert; *Uniform* replaces both the dispatch and combine matrices with uniform averages; *Uniform / Soft* replaces the dispatch weights with a uniform average, but the combine weights are computed as in Soft MoE; *Soft / Uniform* does the opposite replacement; and *Soft* uses the algorithm we present in Section 2.

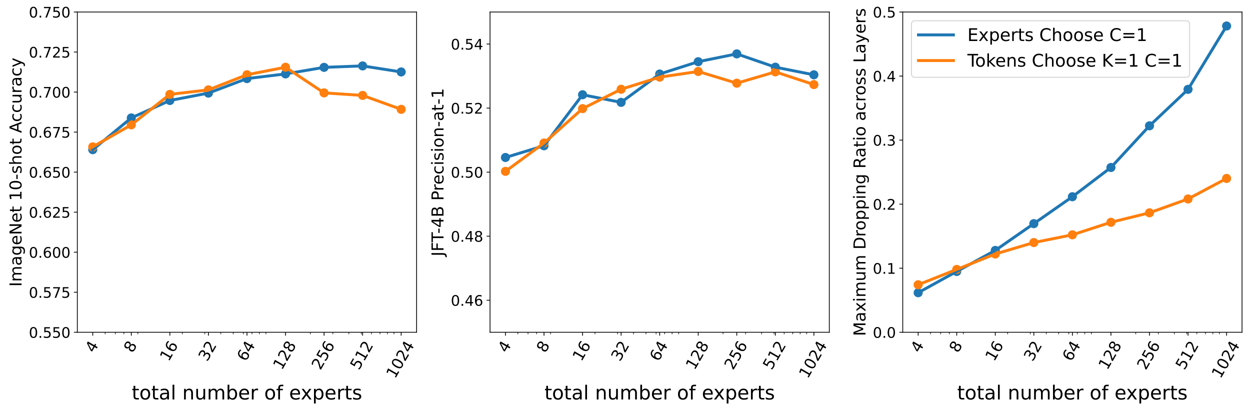


Figure 12: S/14. Performance and amount of token dropping for increasing experts for Experts Choose ($C = 1$) and Tokens Choose ($K = 1$ and $C = 1$).

B Token Dropping

In this appendix, we briefly explore token dropping for the Experts Choose and Tokens Choose algorithms. For Tokens Choose, each token selects K experts. When experts are full, some tokens assigned to that expert will not be processed. A token is "dropped" when none of its choices go through, and no expert at all processes the token. Expert Choose algorithms lead to an uneven amount of processing per token: some input tokens are selected by many experts, while some others are not selected by any. We usually define the number of tokens to be processed by each expert in a way that the combined capacity of all experts corresponds to the number of input tokens (or a multiple C of them). If we use a multiplier C higher than one (say, 2x or 3x), the amount of dropping will decrease but we will pay an increased computational cost. Thus, we mainly explore the $K = 1$ and $C = 1$ setup, where there is no slack in the buffers.

In all cases to follow we see a common trend: fixing everything constant, increasing the number of experts leads to more and more dropping both in Experts Choose and Tokens Choose.

Figure 12 compares Experts Choose and Tokens Choose with the same multiplier $C = 1$. This is the cheapest setup where every token *could* be assigned to an expert with balanced routing. We see that in both cases the amount of dropping quickly grows with the number of experts. Moreover, even though Experts Choose has higher levels of dropping (especially for large number of experts), it is still more performant than Tokens Choose. Note there is a fundamental difference: when Tokens Choose drops a token, the model wastes that amount of potential compute. On the other hand, for Experts Choose dropping just means some other token got that spot in the expert buffer, thus the model just transferred compute from one unlucky token to another lucky one.

In this setup, for a small number of experts (16-32) it is common to observe a $\sim 15\%$ rate of dropping. On the other hand, we also experimented with a large number of experts (100-1000) where each expert selects very few tokens. In this case, the dropping rate for Experts Choose can grow above 40-50% in some layers: most experts select the very same tokens. Tokens Choose seems to completely drop up to ~25% of the tokens.

In Figures 13 and 14 we study how much a little bit of buffer slack ($C = 1.125$) can help in terms of performance and dropping to Experts Choose and Tokens Choose, respectively. Both plots are similar: the amount of dropping goes down around ~5% and performance slightly increases when the number of experts is large. Note that the step time also increases in these cases.

Finally, Figure 15 shows the effect of Batch Priority Routing (Riquelme et al., 2021) for Tokens Choose. By smartly selecting which tokens to drop we do not only uniformly reduce the amount of dropping, but we significantly bump up performance.

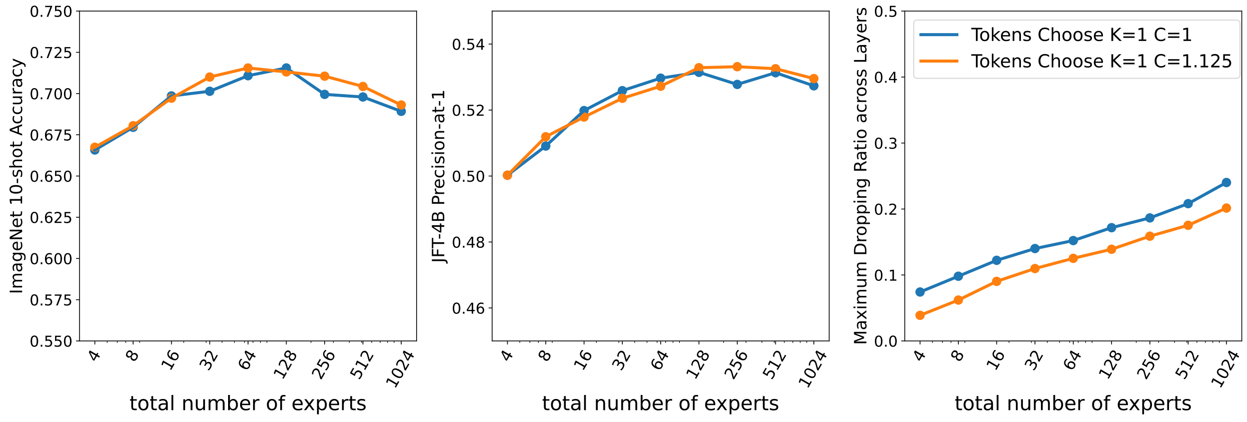


Figure 13: S/14. Performance and amount of token dropping for increasing experts for Tokens Choose with tight buffers ($K = 1$ and $C = 1$) and some amount of buffer slack ($K = 1$ and $C = 1.125$).

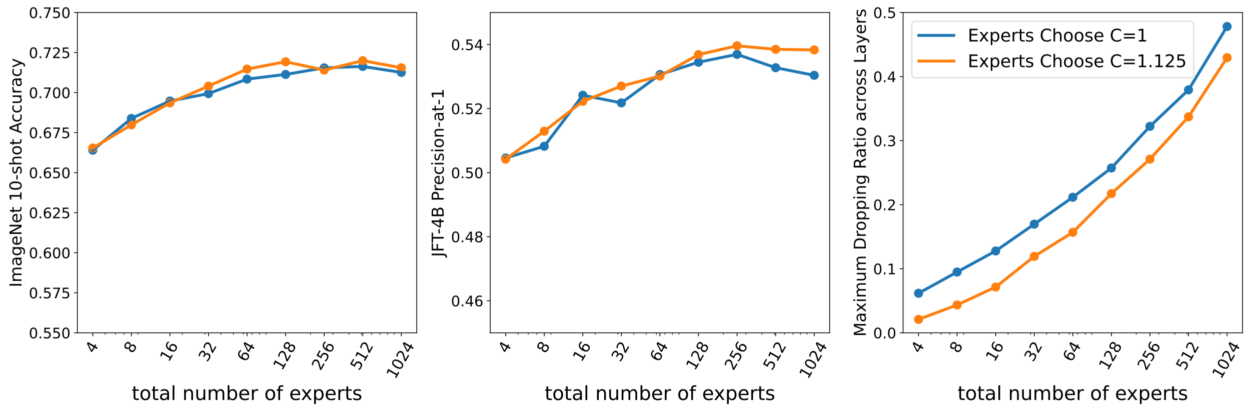


Figure 14: S/14. Performance and amount of token dropping for increasing experts for Experts Choose with tight buffers ($C = 1$) and slightly larger buffers ($C = 1.125$).

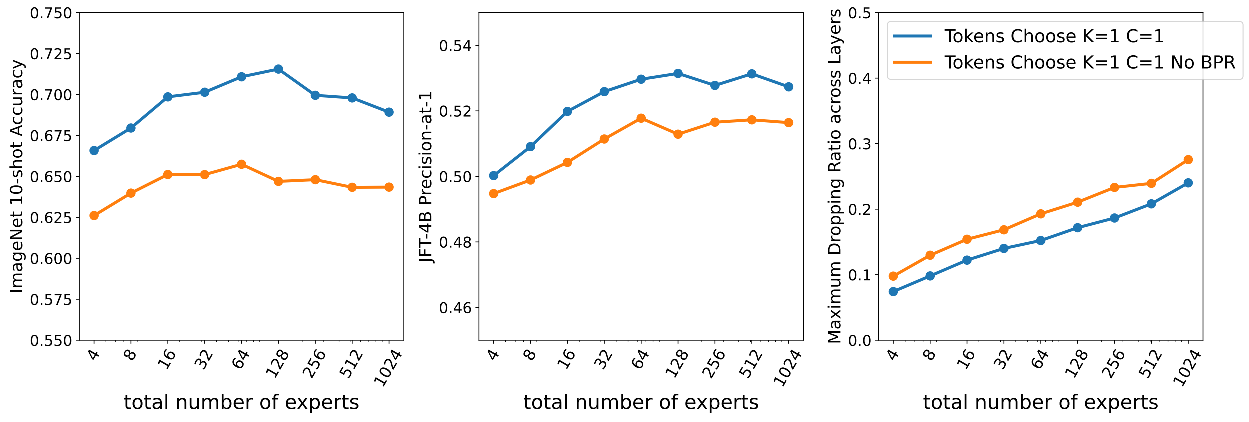


Figure 15: S/14. Performance and amount of token dropping for increasing experts with and without BPR for Tokens Choose.

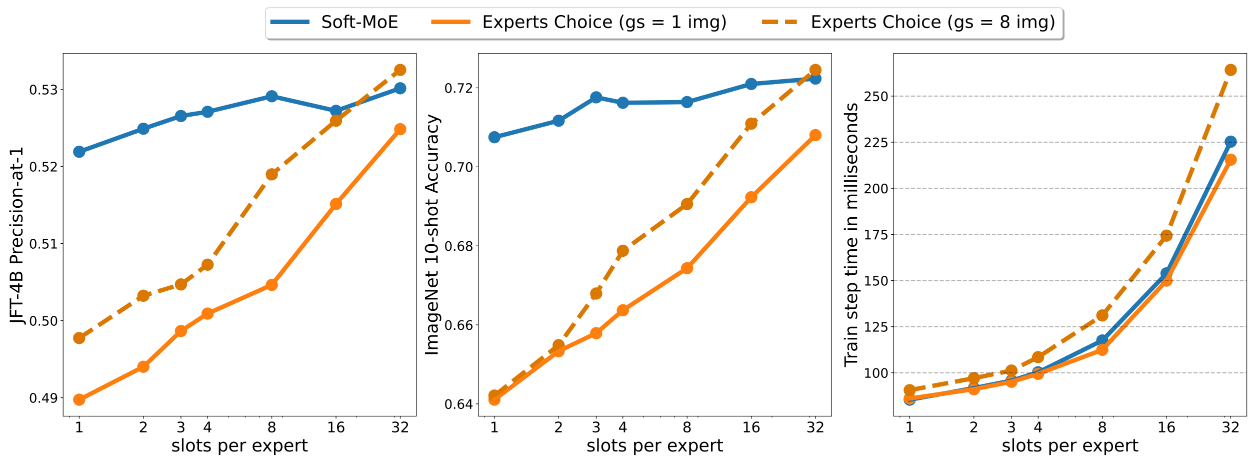


Figure 16: **Performance (left, center) and step time (right) of models with 32 experts, but increased slots, all trained for the same number of steps (300k).** Increasing the number of slots per expert only increases performance of Soft MoE a small amount, while increasing cost substantially.

C Soft MoE Increasing Slots

In this section we explore the following question: for a fixed number of experts, how much does Soft MoE routing benefit from having additional slots per expert? Figure 16 shows results for Soft MoE S/16 with 32 experts. We also show Experts Choice with group sizes of one and eight images. When increasing the number of slots, the performance grows only modestly, while cost increases quickly. Experts Choice benefits much more from increased slots, catching up at a large group size, but at a very large cost.

D Sparse Layers Placement

Soft MoE does unlock the effective use of a large number of experts. An important design choice for sparse models is the number and location of sparse layers, together with the number of experts per layer. Unfortunately, the large number of degrees of freedom in these choices has usually made thorough ablations and optimization unfeasible. In this section, we provide the results of a simple experiment that can help better design the configuration of sparse models. We fix a total number of experts ($E = 512$) with one slot per expert, thus leading to matched number of parameters (note in this case FLOPs may vary greatly depending on the number of sparse layers). Then, for an S/16 backbone architecture, we distribute those experts in various ways (all in one layer, half of them in two layers, etc) and compare their performance after 300k training steps. Table 5 shows the results. Again, we observe that a number of experts close to the number of input tokens (there are 196 tokens, given the 16x16 patch size for 224x224 images) split over the last few layers works best. Moreover, note these models are indeed cheaper than those in the comparison with 512 or 256 experts per layer. Table 6 offers results for Tokens Choose routing with $K = 1$ and BPR Riquelme et al. (2021). In this case, all algorithms use a comparable FLOPs count (ignoring slightly increasing routing costs with more experts). Results are essentially similar, thus suggesting optimal expert placement (including expert count and location) may not strongly depend on the routing algorithm.

Table 5: Expert placing ablation with a Soft MoE S/16 with 12 layers (indexed from 0 to 11).

Sparse Layers	Experts per Layer	Total Experts	IN/10shot	JFT prec1
11	512	512	70.0%	51.5%
10	512	512	70.1%	52.0%
10, 11	256	512	71.7%	52.2%
5, 11	256	512	70.4%	52.1%
8, 9, 10, 11	128	512	**72.8%**	**53.2%**
2, 5, 8, 11	128	512	71.1%	52.5%
4:11	64	512	**72.1%**	**53.1%**
1:4, 8:11	64	512	70.5%	52.1%

Table 6: Expert placing ablation with a V-MoE S/16 Tokens Choose $K = 1$ with 12 layers (indexed as 0:11).

Sparse Layers	Experts per Layer	Total Experts	IN/10shot	JFT prec1
11	512	512	64.4%	50.1%
10	512	512	67.2%	51.9%
10, 11	256	512	68.6%	51.3%
5, 11	256	512	65.3%	50.6%
8, 9, 10, 11	128	512	**69.1%**	**52.3%**
2, 5, 8, 11	128	512	67.3%	51.1%
4:11	64	512	**69.9%**	**52.2%**
1:4, 8:11	64	512	68.0%	51.2%

Table 7: Expert placing ablation with a V-MoE S/16 Experts Choose $C = 1$ with 12 layers (indexed as 0:11).

Sparse Layers	Experts per Layer	Total Experts	IN/10shot	JFT prec1
11	512	512	65.3%	50.3%
10	512	512	66.5%	51.7%
10, 11	256	512	68.8%	51.8%
5, 11	256	512	65.9%	51.1%
8, 9, 10, 11	128	512	**69.4%**	**52.2%**
2, 5, 8, 11	128	512	68.0%	51.7%
4:11	64	512	**69.0%**	**52.2%**
1:4, 8:11	64	512	67.4%	51.1%

E The collapse of softmax layers applied after layer normalization

E.1 Theoretical analysis

A softmax layer with parameters $\Theta \in \mathbb{R}^{n\times d}$ transforms a vector $x \in R^d$ into the vector softmax$(\Theta x) \in \mathbb{R}^n$, with elements:

$$\text{softmax}(\Theta x)_i = \frac{\exp((\Theta x)_i)}{\sum_{j=1}^n \exp((\Theta x)_j)} = \frac{\exp(\sum_{k=1}^d \theta_{ik} x_k)}{\sum_{j=1}^n \exp(\sum_{k=1}^d \theta_{jk} x_k)} \tag{4}$$

Layer normalization applies the following operation on $x \in \mathbb{R}^d$.

$$\text{LN}(x)_i = \alpha_i \frac{x_i - \mu(x)}{\sigma(x)} + \beta_i; \quad \text{where } \mu(x) = \frac{1}{d}\sum_{i=1}^d x_i \text{ and } \sigma(x) = \sqrt{\frac{1}{d}\sum_{i=1}^d (x_i - \mu(x_i))^2} \tag{5}$$

Notice that $\text{LN}(x) = \text{LN}(x - \mu(x))$, thus we can rewrite LayerNorm with respect to the centered vector $\tilde{x} = x - \mu(x)$, and the centered vector scaled to have unit norm $\hat{x}_i = \frac{\tilde{x}_i}{\|\tilde{x}\|}$:

$$\text{LN}(\tilde{x})_i = \alpha_i \frac{\tilde{x}_i}{\sqrt{\frac{1}{d}\sum_{j=1}^d \tilde{x}_j^2}} + \beta_i = \sqrt{d}\alpha_i \frac{\tilde{x}_i}{\|\tilde{x}\|} + \beta_i = \sqrt{d}\alpha_i \hat{x}_i + \beta_i \tag{6}$$

When a softmax layer is applied to the outputs of layer normalization, the outputs of the softmax are given by the equation:

$$\text{softmax}(\Theta\text{LN}(x))_i = \frac{\exp(\sum_{k=1}^d \theta_{ik}(\sqrt{d}\alpha_k \hat{x}_k + \beta_k))}{\sum_{j=1}^n \exp(\sum_{k=1}^d \theta_{jk}(\sqrt{d}\alpha_k \hat{x}_k + \beta_k))} \tag{7}$$

By setting $\vartheta_i = \sum_{k=1}^d \theta_{ik}\alpha_k \hat{x}_k$, and $\delta_i = \sum_{k=1}^d \theta_{ik}\beta_k$, the previous equation can be rewritten as:

$$\text{softmax}(\Theta\text{LN}(x))_i = \frac{\exp(\sqrt{d}\vartheta_i + \delta_i)}{\sum_{j=1}^n \exp(\sqrt{d}\vartheta_j + \delta_j)} \tag{8}$$

Define $m = \max_{i\in[n]} \sqrt{d}\vartheta_i - \delta_i$, $M = \{i \in [n] : \sqrt{d}\vartheta_i - \delta_i = m\}$. Then, the following equality holds:

$$\text{softmax}(\Theta\text{LN}(x))_i = \frac{\exp(\sqrt{d}\vartheta_i + \delta_i - m)}{\sum_{j=1}^n \exp(\sqrt{d}\vartheta_j + \delta_j - m)} \tag{9}$$

Given that $\lim_{d\to\infty} \exp(\sqrt{d}\vartheta_i + \delta_i - m) = \begin{cases} 1 : i \in M \\ 0 : i \notin M \end{cases}$ the output of the softmax tends to:

$$\lim_{d\to\infty} \text{softmax}(\Theta\text{LN}(x))_i = \begin{cases} \frac{1}{|M|} & i \in M \\ 0 & i \notin M \end{cases} \tag{10}$$

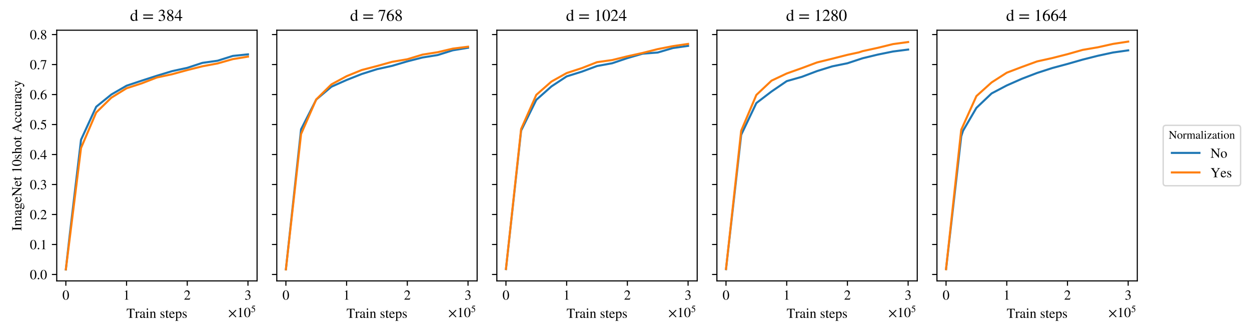
In particular, when the maximum is only achieved by one of the components (i.e. $|M| = 1$), the softmax collapses to a one-hot vector (a vector with all elements equal to 0 except for one).

E.2 Empirical analysis

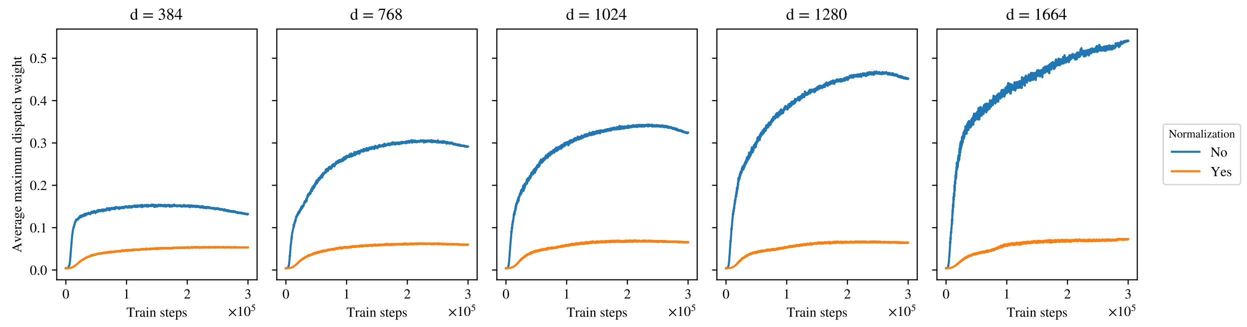
The previous theoretical analysis assumes that the parameters of the softmax layer are constants, or more specifically that they do not depend on d. One might argue that using modern parameter initialization techniques, which take into account $\frac{1}{\sqrt{d}}$ in the standard deviation of the initialization Glorot and Bengio (2010); He et al. (2015); Klambauer et al. (2017), might fix this issue. We found that they don't (in particular, we use the initialization from Glorot and Bengio (2010)).

Figure 17 shows different metric curves during the training of a small SoftMoE model with different model dimensions. The model dimensions are those corresponding to different standard backbones: S (384), B (768), L (1024), H (1280) and G (1664). The rest of the architecture parameters are fixed: 6 layers (3 dense layers followed by 3 MoE layers with 256 experts), 14x14 patches, and a MLP dimension of 1536. As the model dimension d increases, the figure shows that, if the inputs to the softmax in the SoftMoE layers are not normalized, the average maximum values of the dispatch and combine weights tend to grow (especially the former). When d is big enough, the ImageNet 10shot accuracy is significantly worse than that achieved by properly normalizing the inputs.

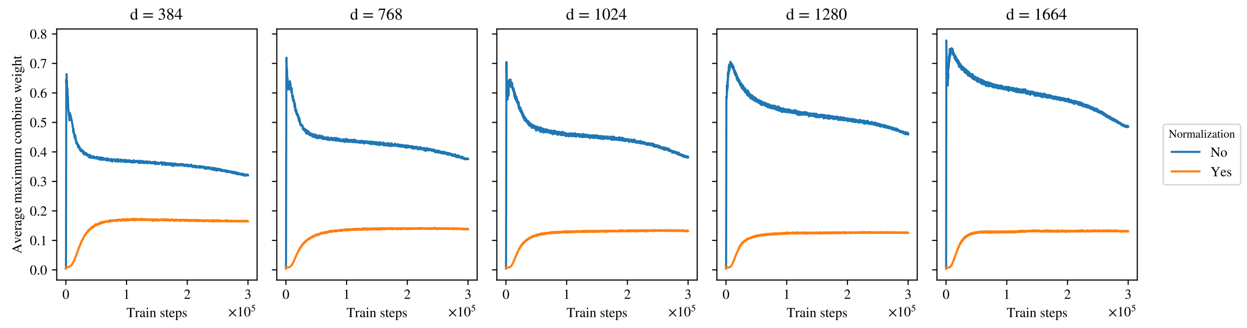
In the previous experiment, we trained our model with a linear decay schedule and a peak value of 10^{-3}. In addition, we also found that applying the softmax layer directly on the output of layer normalization is also very sensible to the learning rate's configuration. Once again, our recipe suggested in Section 2.3 gives equal or better quality, and is generally more stable. Figure 18 shows different metric curves during the training of the same small SoftMoE model as before, with a model dimension of $d = 1664$, using an inverse square root learning rate schedule, with a fixed timescale of 10^5, a linear warmup phase of 10^5 steps, and a linear cooldown of $5 \cdot 10^5$ steps, varying the peak learning rate value. In this figure, similarly to the results from the previous experiment, the average maximum values of the dispatch and combine weights grows to values approaching 1.0 (indicating a collapse in the softmax layers to a one-hot vector), when the inputs to the softmax in the SoftMoE layers are not normalized, which eventually severely hurts the accuracy of the model. However, using the normalization in Section 2.3 gives better accuracy and makes the model less sensible to the choice of the peak value of the learning rate.



(a) ImageNet 10shot accuracy.

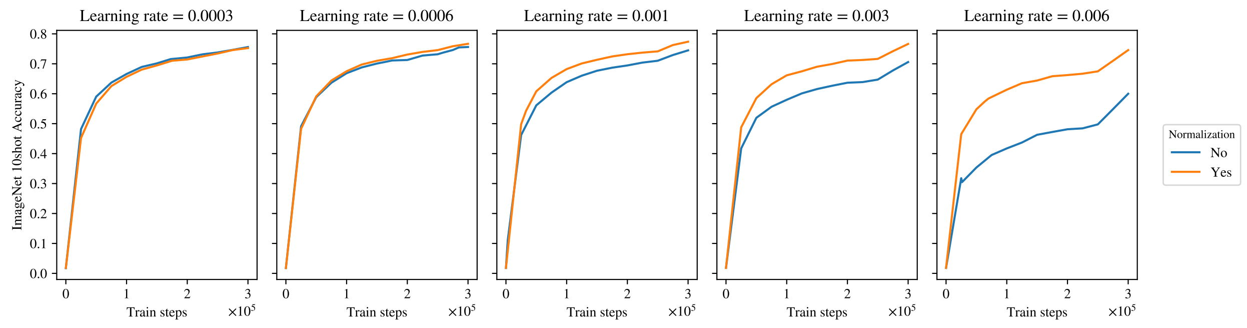


(b) Average value of the maximum dispatch weight per slot.

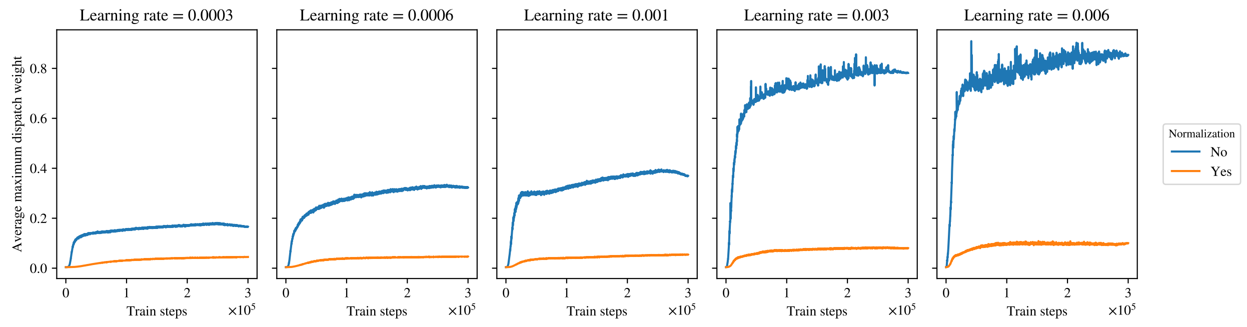


(c) Average value of the maximum combine weight per token.

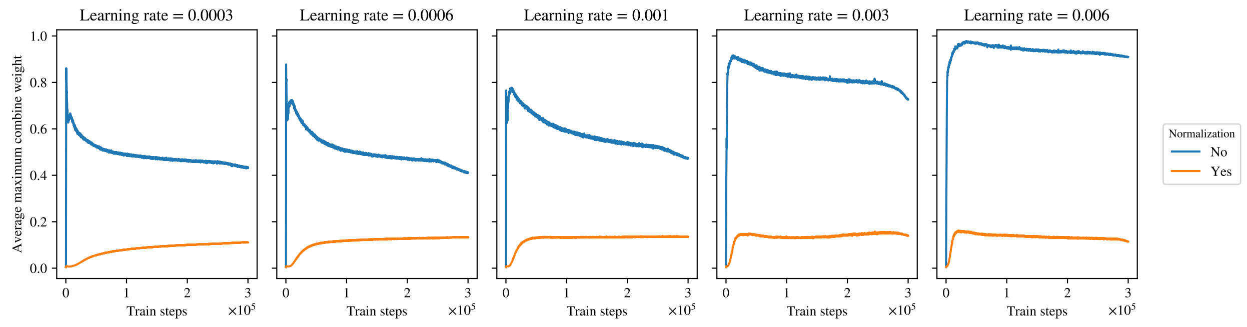
Figure 17: Training plots of the ImageNet 10shot accuracy (top), the average value of the maximum dispatch weight per slot (middle) and the average value of the maximum combine weight per token (bottom) for different model dimensions d. Observe that maximum values of the combine and (especially) the dispatch weights grow as the model dimension grows during training, as our theoretical analysis predicted. Although the ImageNet 10shot accuracy is similar for small model dimensions, applying the softmax layer directly on the output of layer normalization, without any further re-normalization, hurts the accuracy as the model dimension d grows. By normalizing the inputs to the softmax as suggested in Section 2.3 improves the performance for large values of d.



(a) ImageNet 10shot accuracy.



(b) Average value of the maximum dispatch weight per slot.



(c) Average value of the maximum combine weight per token.

Figure 18: Training plots of the ImageNet 10shot accuracy (top), the average value of the maximum dispatch weight per slot (middle) and the average value of the maximum combine weight per token (bottom) for different peak values of the learning rate, using a model dimension of $d = 1664$ (i.e. that of a G backbone).

F Additional Results

Table 8: Comparison between Top-K with and without BPR.

Model	Number of Experts	K	BPR	JFT prec@1	IN/10shot
V-MoE S/16	32	1	No	50.1%	64.5%
V-MoE S/16	32	1	Yes	51.2%	68.9%
V-MoE S/16	32	2	No	52.5%	71.0%
V-MoE S/16	32	2	Yes	52.8%	71.4%
V-MoE S/16	64	1	No	50.0%	64.4%
V-MoE S/16	64	1	Yes	51.5%	69.1%
V-MoE S/16	64	2	No	52.9%	70.9%
V-MoE S/16	64	2	Yes	52.9%	71.4%

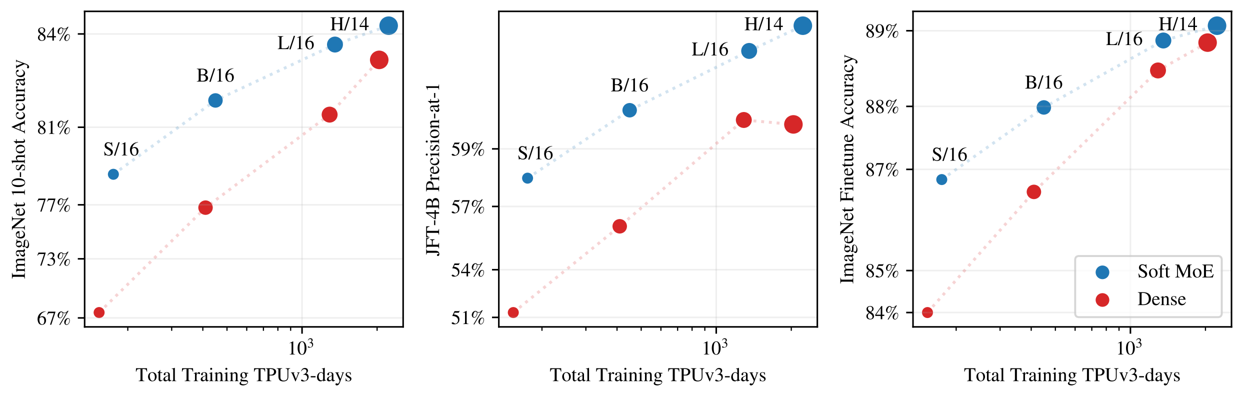


Figure 19: **Long runs.** Soft MoE and ViT models trained for 4 million steps with batch size 4096 (H/14 models trained for 2 million steps instead). Equivalent model classes (S/16, B/16, L/16, H/14) have similar training costs, but Soft MoE outperforms ViT on all metrics. We show ImageNet 10-shot (left), JFT precision at 1 (middle) and ImageNet accuracy after finetuning (right), versus total training FLOPs. See Table 2. We report training FLOPs in Figure 4.

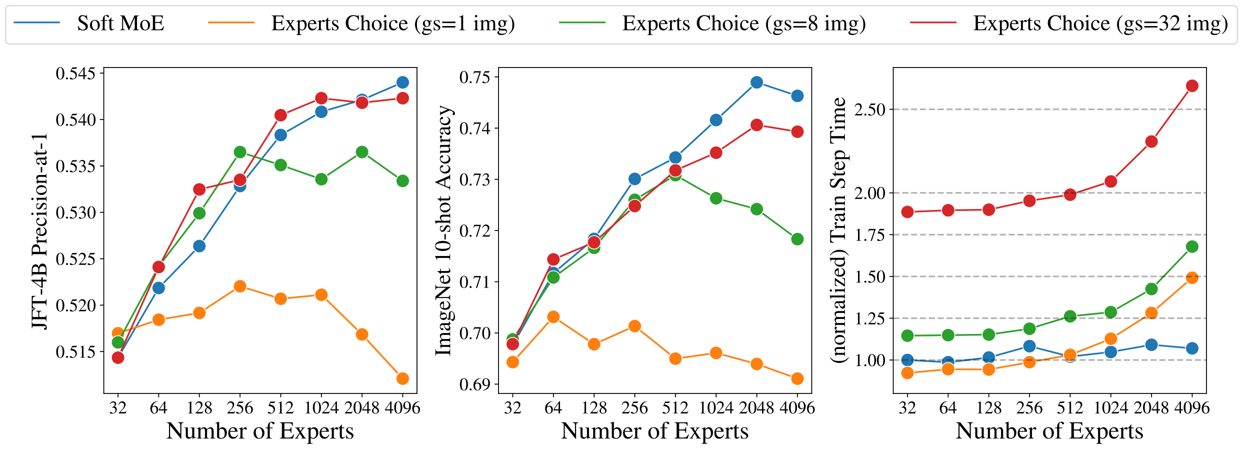


Figure 20: JFT precision-at-1, ImageNet 10-shot accuracy, and normalized training step time when increasing the total number of experts while keeping the total amount of slots fixed. Soft MoE achieves consistently better results with more experts, whereas cost is kept roughly constant. Adding too many experts to Experts Choice hurt performance and significantly increases the cost. Experts Choice can perform well with many experts if we increase the group size up to 32 images per group. The normalized train step time is computed with respect to Soft MoE with 32 experts. Experts Choice with 32 images per group and 4096 experts requires more than 2.5x its cost.

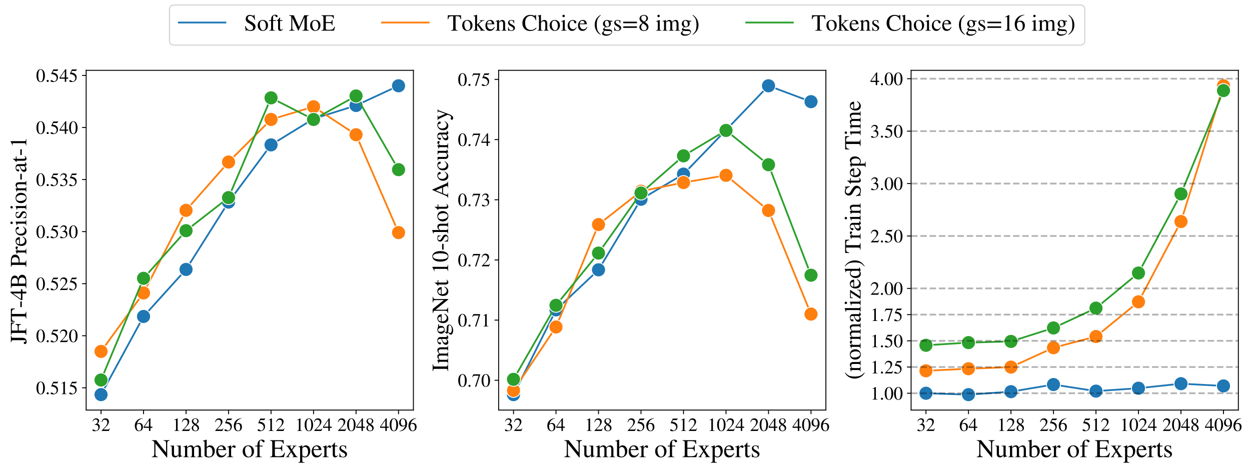
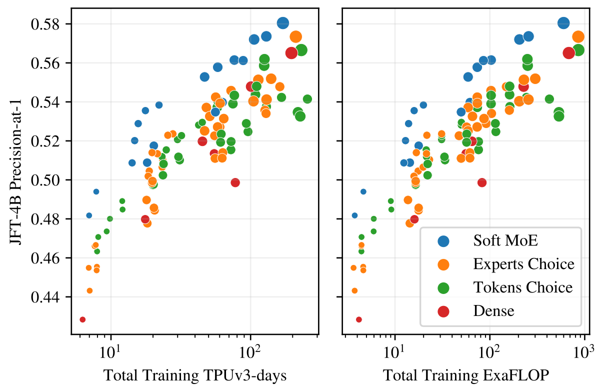
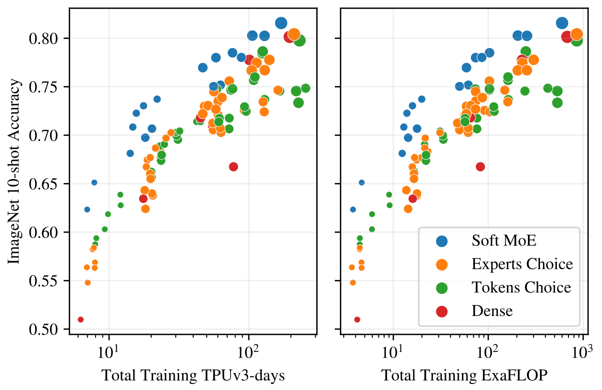


Figure 21: JFT precision-at-1, ImageNet 10-shot accuracy, and normalized training step time when increasing the total number of experts while keeping the total amount of slots fixed. Soft MoE achieves consistently better results with more experts, whereas cost is kept roughly constant. Adding too many experts to Tokens Choice hurt performance and significantly increases the cost. Even with a large group size (16 images), Tokens Choice struggles to perform well with a few thousand experts. The normalized train step time is computed with respect to Soft MoE with 32 experts. Tokens Choice with 8 or 16 images per group and 4096 experts requires almost 4x its cost.

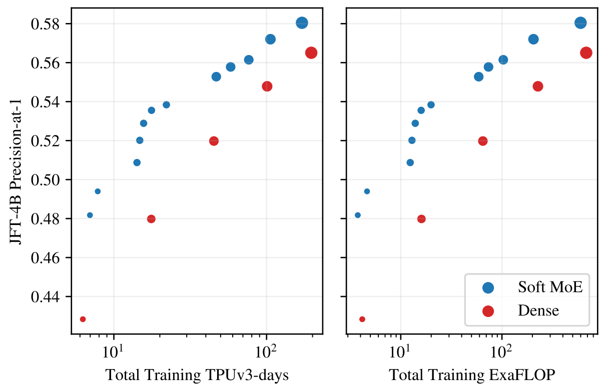


(a) JFT-4B Precision-at-1

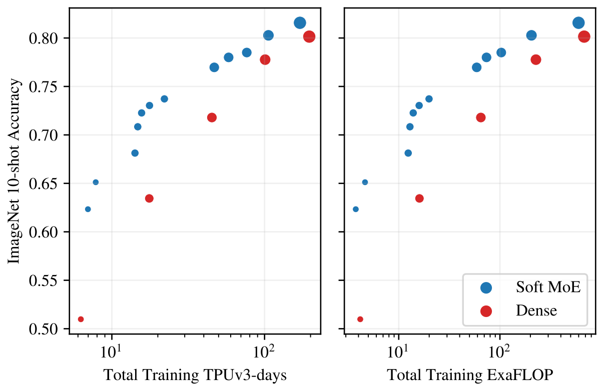


(b) ImageNet 10-shot Accuracy

Figure 22: JFT-4B Precision-at-1 and ImageNet 10-shot accuracy on short runs (300k steps). The size of the marker depends on the backbone size: S/32, S/16, B/32, B/16, L/16 and H/14. Colors represent different methods: Soft MoE (blue), Sparse MoEs with Experts Choice (orange) and Tokens Choice routing (green), and a Dense (red) model. MoE runs include different configurations.

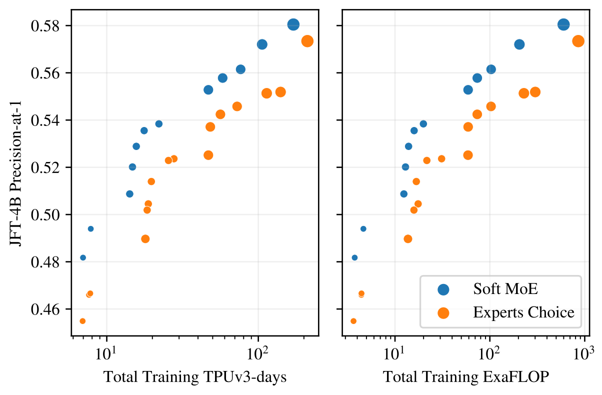


(a) JFT-4B Precision-at-1

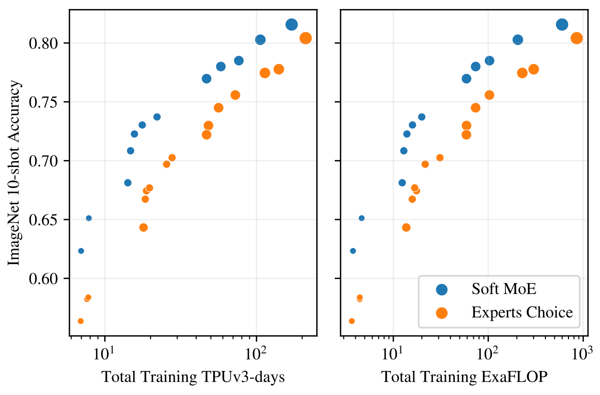


(b) ImageNet 10-shot Accuracy

Figure 23: JFT-4B Precision-at-1 and ImageNet 10-shot accuracy on short runs (300k training steps). The size of the marker depends on the backbone size: S/32, S/16, B/32, B/16, L/16 and H/14. Colors represent different methods: Soft MoE (blue) and Dense (red) models. MoE runs include different configurations. We only show the runs that are not dominated by another model using the same method (S/8 and L/32 were always dominated).

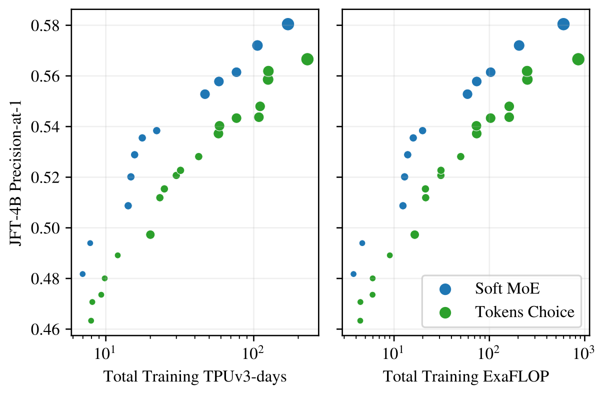


(a) JFT-4B Precision-at-1

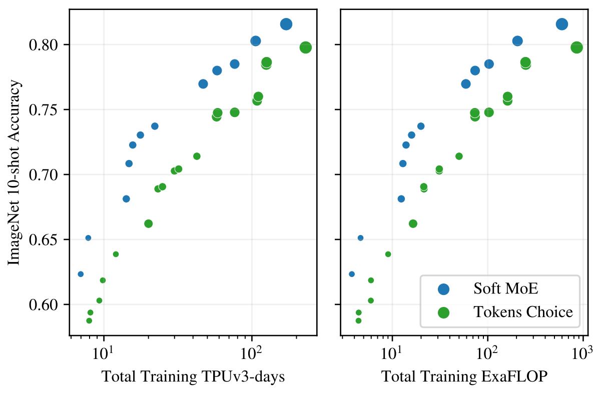


(b) ImageNet 10-shot Accuracy

Figure 24: JFT-4B Precision-at-1 and ImageNet 10-shot accuracy on short runs (300k training steps). The size of the marker depends on the backbone size: S/32, S/16, B/32, B/16, L/16 and H/14. Colors represent different methods: Soft MoE (blue) and Sparse MoEs with Experts Choice (orange) models. MoE runs include different configurations. We only show the runs that are not dominated by another model using the same method (S/8 and L/32 were always dominated).



(a) JFT-4B Precision-at-1



(b) ImageNet 10-shot Accuracy

Figure 25: JFT-4B Precision-at-1 and ImageNet 10-shot accuracy on short runs (300k training steps). The size of the marker depends on the backbone size: S/32, S/16, B/32, B/16, L/16 and H/14. Colors represent different methods: Soft MoE (blue) and Sparse MoEs with Tokens Choice (green) models. MoE runs include different configurations. We only show the runs that are not dominated by another model using the same method (S/8 and L/32 were always dominated).

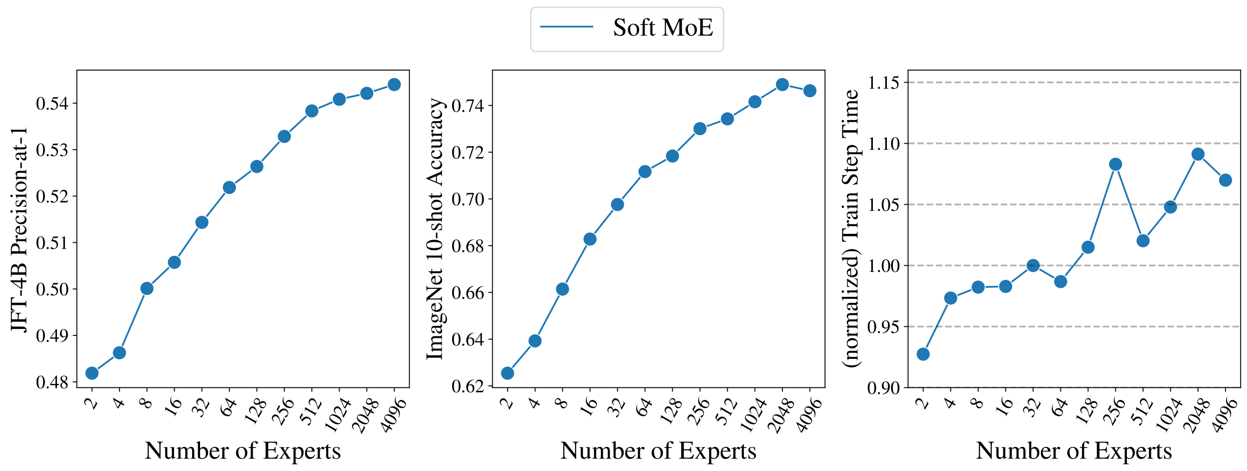


Figure 26: **JFT Precision-at-1, ImageNet 10-shot Accuracy, and normalized Training Step time when increasing the total number of experts while keeping the total amount of slots fixed (4096).** Soft MoE achieves consistently better results with more experts, whereas cost is kept roughly constant (same FLOPs but communication costs vary due to higher topologies needed for larger models). The normalized train step time is computed with respect to Soft MoE with 32 experts. Model sizes range from 38M (2 experts) to 9.7B parameters (4096 experts).

G Additional analysis

G.1 Cumulative sum of dispatch and combine weights

Figure 27 shows the distribution over slots of the cumulative sum (over tokens) of their corresponding dispatch weights. For each slot we compute the cumulative sum of the dispatch weights over tokens sorted in decreasing order. This indicates how many tokens are necessary to cover a given percentage of the total mass of the weighted average. We compute this cumulative sum for all slots over all the 50 000 ImageNet validation images, across all layers of the Soft MoE H/16 model after finetuning. In the plot, we represent with a solid line the average (over all slots and images) cumulative sum, and the different colored areas represent the central 60%, 80%, 90%, 95% and 99% of the distribution (from darker to lighter colors) of cumulative sums.

This tells us, for instance, how uniform is the weighted average over tokens used to compute each input slot. In particular, each slot in the last two layers is close to a uniform average of all the tokens (a completely uniform average would be represented by a straight line). This tells us that in these layers, every expert processes roughly the same inputs, at least after the model is trained. However, this weighted average is far from uniform in the rest of the layers, meaning that there are tokens that contribute far more than others. For example, in layer 28, a few tens of tokens already cover 80% of the weighted average mass. Finally, given the width of the colored areas, we can also see that there's a significant difference on the weighted averages depending on the slot, across all layers (except maybe the last two). This indicates that the dispatch weights vary across different slots and images.

Similarly, Figure 28 shows the corresponding plots for the cumulative sum of the combine weights. In this case, for each output token we compute the cumulative sum of the combine weights over slots sorted in decreasing order. Notice that, although the dispatch weights in the last two layers were almost uniform, the combine weights are not. This indicates that some slots (and thus, experts) are more important than others in computing the output tokens, and thus their corresponding expert parameters are not redundant. Of course, the identity of the "important" slots may vary depending on the input token.

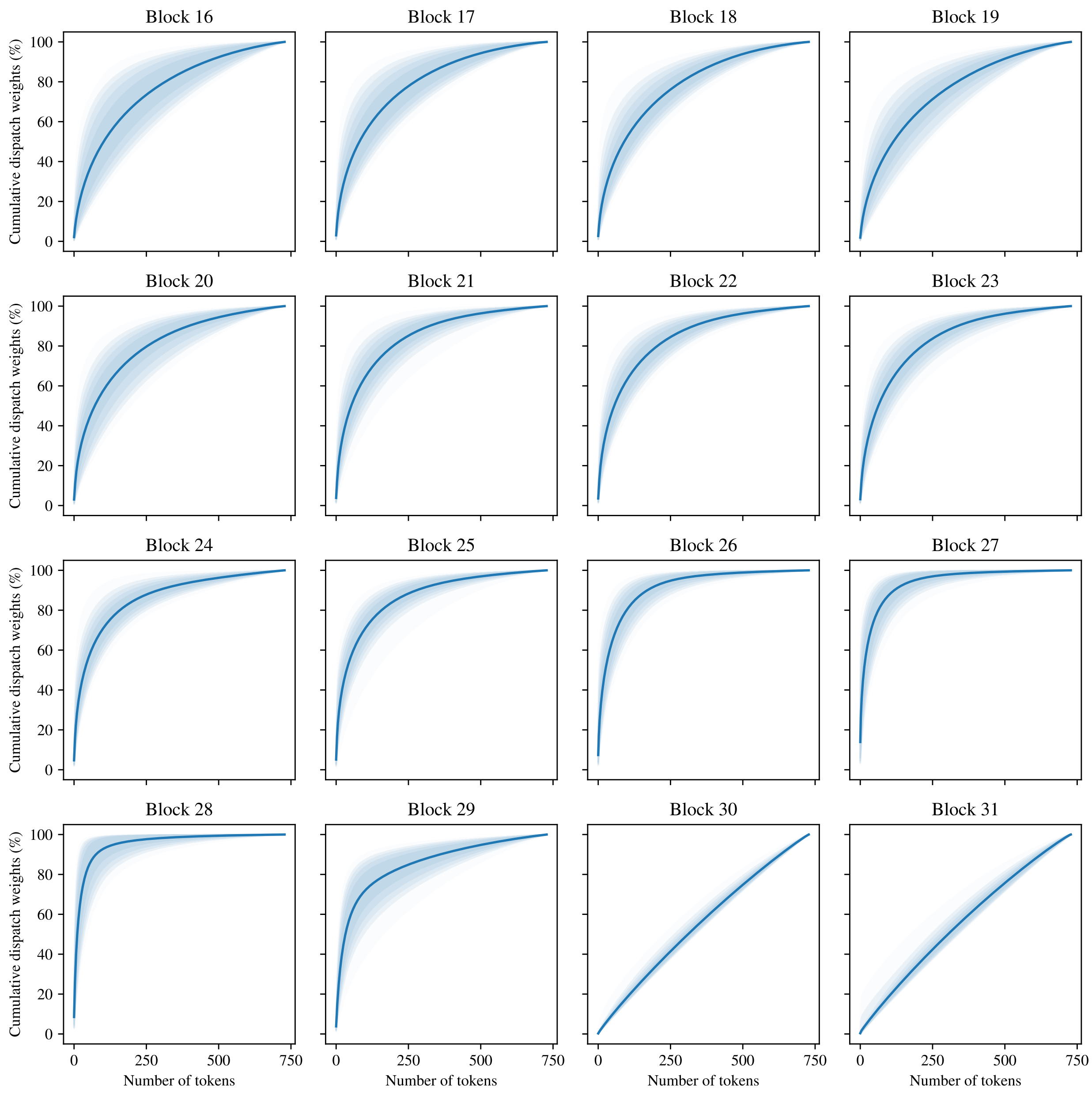


Figure 27: **Distribution of the cumulative sum of dispatch weights.** For each input slot, we compute the cumulative sum of its corresponding dispatch weights (sorted by decreasing value). This indicates over how many input tokens a certain cumulative weight is distributed over. The line in each plot represents the average computed over all slots and ImageNet validation images of the given block in the SoftMoE H/14 model. The colored areas represent the central 60%, 80%, 90%, 95% and 99% of the distribution (from darker to lighter, better seen in color).

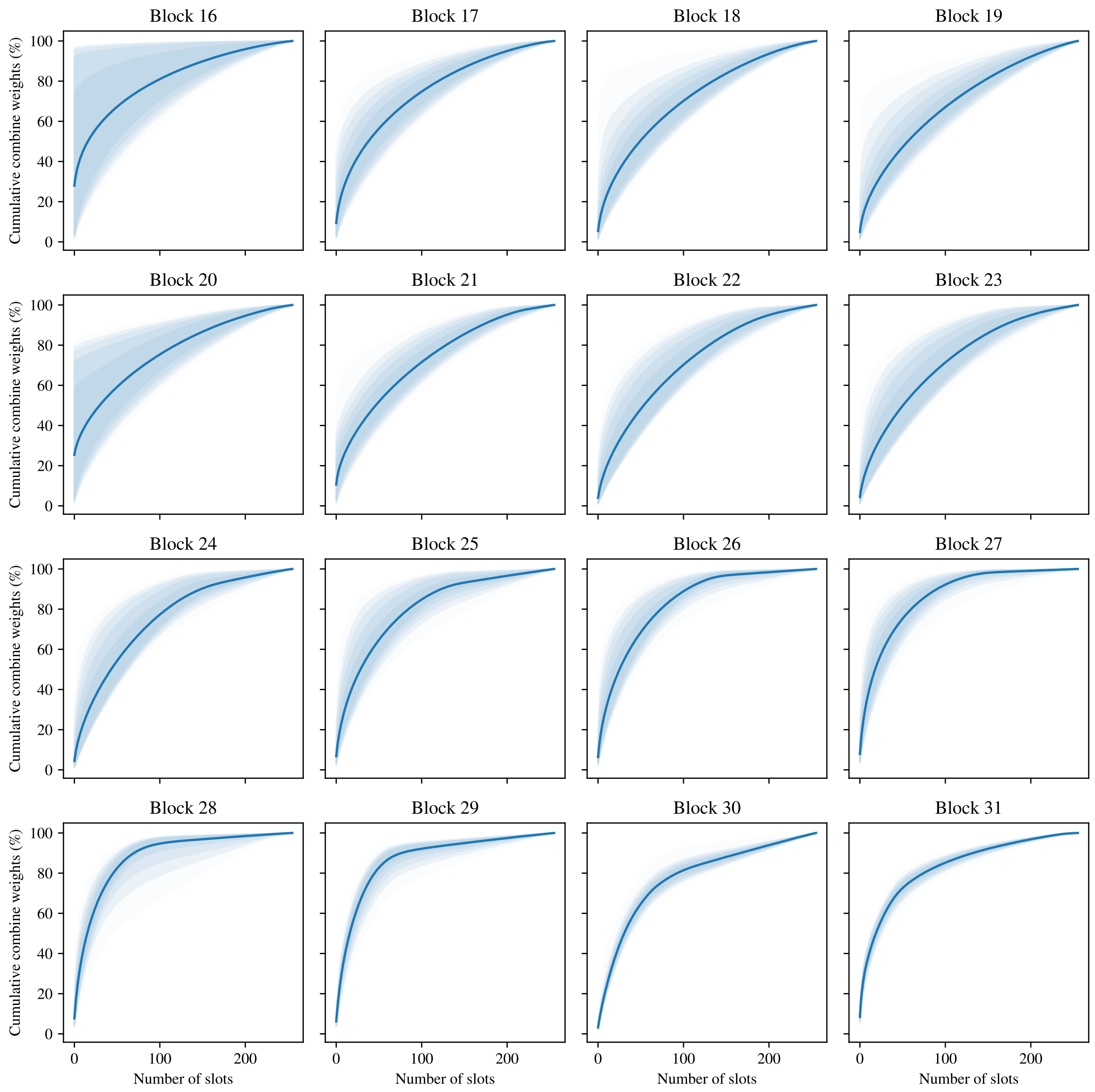


Figure 28: **Distribution of the cumulative sum of combine weights.** For each output token, we compute the cumulative sum of its corresponding combine weights (sorted by decreasing value). This indicates over how many output slots a certain cumulative weight is distributed over. The line in each plot represents the average computed over all tokens and ImageNet validation images of the given block in the SoftMoE H/14 model. The colored areas represent the central 60%, 80%, 90%, 95% and 99% of the distribution (from darker to lighter, better seen in color).

H Slot Correlation

In this section we explore the correlation between the different slot *parameters* that Soft MoE learns, and its relationship with the number of slots per expert. Figures 29 to 31 show for each of 6 layers in a Soft MoE S/16 the inner product between each pair of (normalized) slot parameter vectors.

While Figure 29 shows no clear relationship between slots from different experts (as each expert only has one slot), we observe in Figures 30 and 31 how consecutive slots (corresponding to the same expert) are extremely aligned. This confirms our hypothesis that adding more slots to experts does not work very well as these slots end up aligning their value, and computing somewhat similar linear combinations. Therefore, these projections do not add too much useful information to the different tokens to be processed by the experts (in the extreme, these slots would be identical).

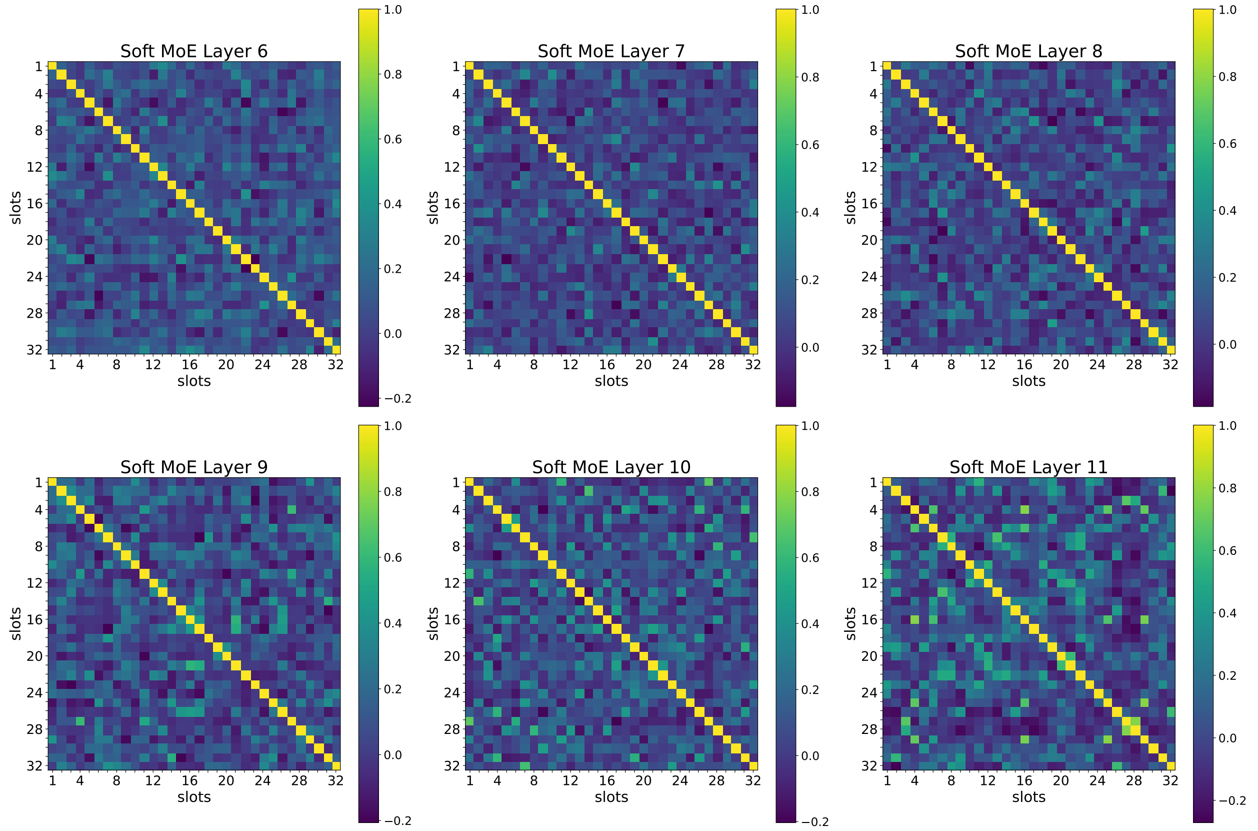


Figure 29: Soft MoE S/16 with 1 slot per expert.

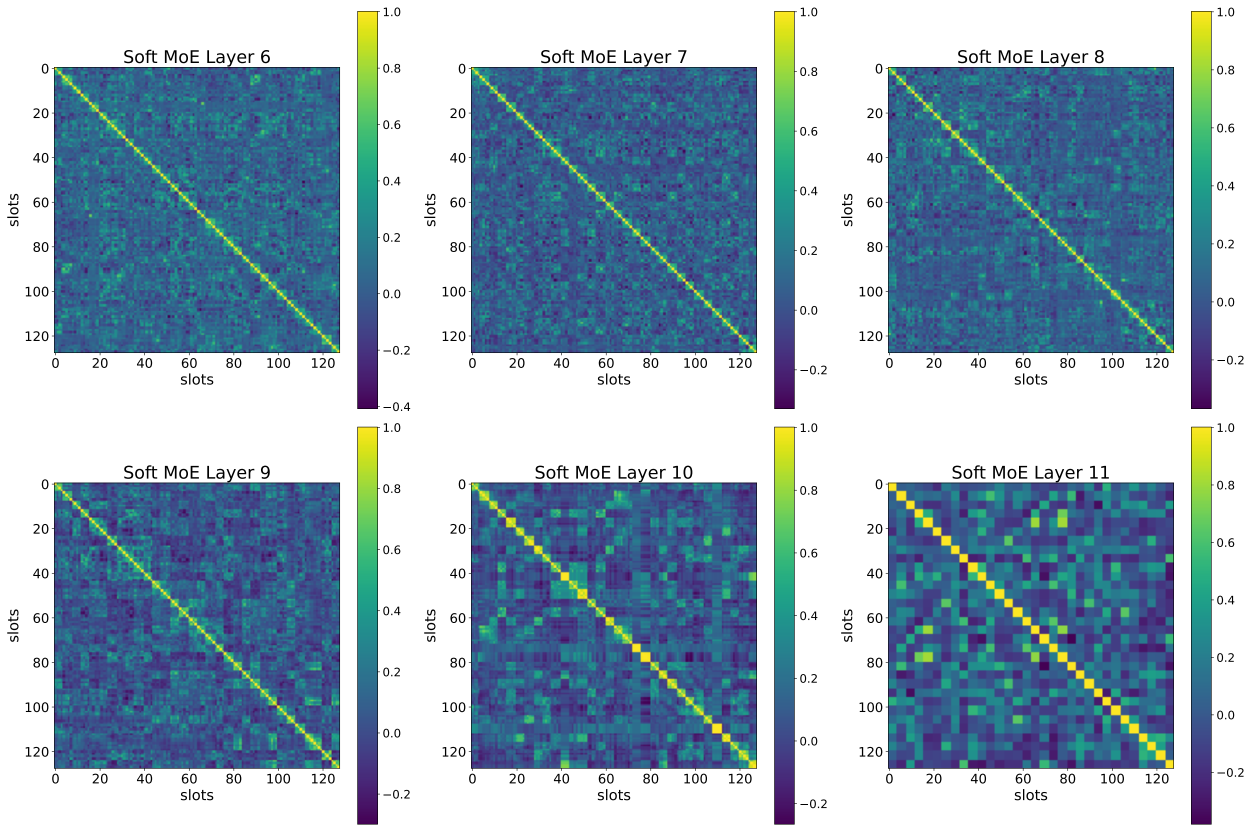


Figure 30: Soft MoE S/16 with 4 slots per expert.

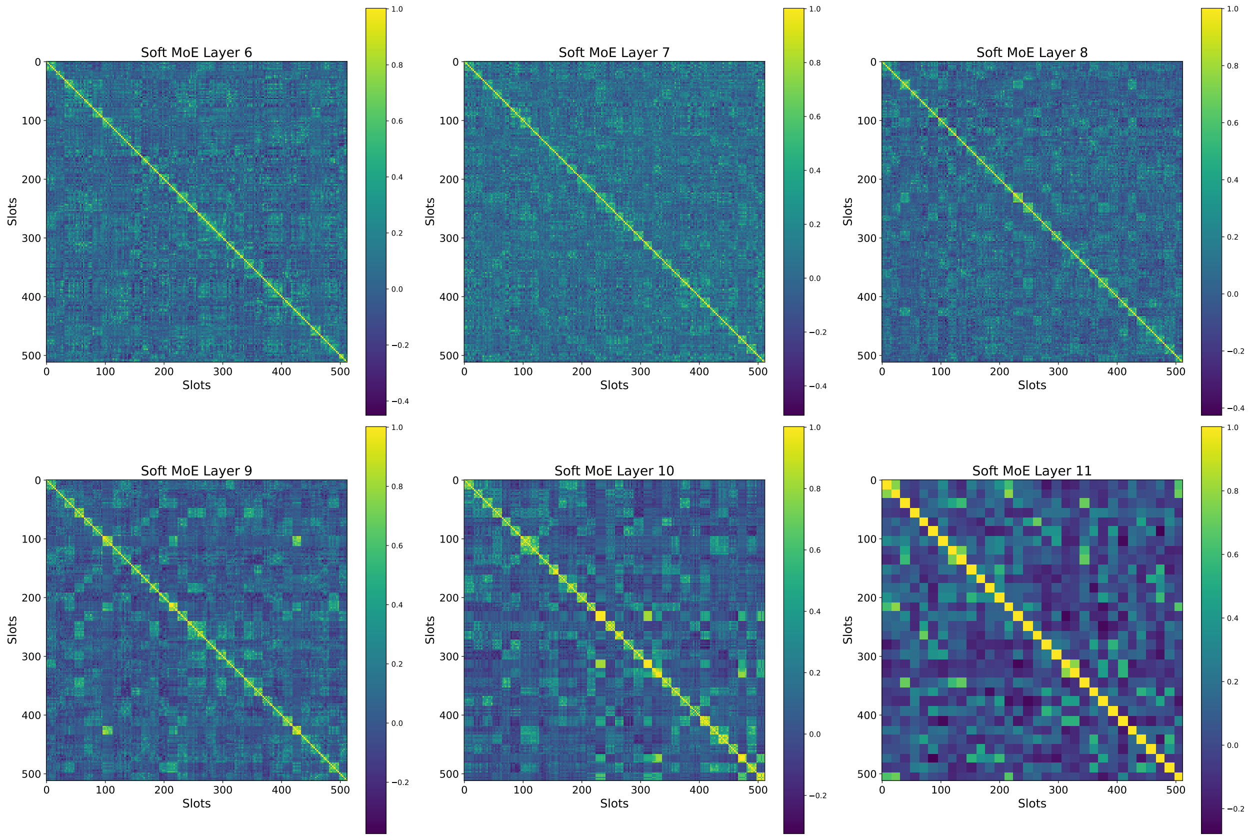


Figure 31: Soft MoE S/16 with 16 slots per expert.

I Pareto Models

Table 9: Model runs from Section 3.3 (shown in Pareto plot) trained for 300k steps on JFT with inverse square root decay and 50k steps cooldown. We trained dense and MoE (Soft MoE, Tokens Choice, Experts Choice) models with sizes S/32, S/16, S/8, B/32, B/16, L/32, L/16 and H/14. Sorted by increasing training TPUv3 days.

Ref	Model	Routing	Experts	Group Size	K	C	JFT P@1	IN/10shot	Train exaFLOP	Train Days
1	S/32	Dense	–	–	–	–	42.8	51.0	4.2	6.3
2	S/32	Experts Choice	32	392	–	0.5	45.5	56.4	3.7	6.9
3	S/32	Soft MoE	32	49	–	–	48.2	62.3	3.8	7.0
4	S/32	Experts Choice	32	49	–	0.5	44.3	54.8	3.8	7.0
5	S/32	Experts Choice	32	392	–	1.0	46.6	58.2	4.4	7.6
6	S/32	Experts Choice	64	392	–	1.0	46.7	58.4	4.4	7.8
7	S/32	Soft MoE	64	49	–	–	49.4	65.1	4.6	7.8
8	S/32	Experts Choice	64	49	–	1.0	45.4	56.3	4.6	7.9
9	S/32	Experts Choice	32	49	–	1.0	45.5	56.9	4.6	7.9
10	S/32	Tokens Choice	32	392	1	1.0	46.3	58.7	4.4	8.0
11	S/32	Tokens Choice	64	392	1	1.0	47.1	59.4	4.4	8.1
12	S/32	Tokens Choice	32	392	2	1.0	47.4	60.3	6.0	9.3
13	S/32	Tokens Choice	64	392	2	1.0	48.0	61.9	6.0	9.8
14	B/32	Dense	–	–	–	–	48.0	63.4	16.0	11.7
15	B/32	Soft MoE	32	49	–	–	50.9	69.7	14.3	11.7
16	S/32	Tokens Choice	64	392	2	2.0	48.9	63.9	9.0	12.0
17	S/32	Tokens Choice	32	392	2	2.0	48.5	62.8	9.1	12.1
18	S/16	Soft MoE	16	196	–	–	50.9	68.1	12.4	14.2
19	S/16	Soft MoE	32	196	–	–	52.0	70.8	12.9	14.8
20	S/16	Dense	–	–	–	–	47.9	60.8	17.0	15.3
21	S/16	Soft MoE	64	196	–	–	52.9	72.3	13.9	15.7
22	S/16	Soft MoE	128	196	–	–	53.6	73.0	15.9	17.6
23	B/32	Experts Choice	32	392	–	0.5	49.0	64.3	13.7	18.0
24	B/32	Experts Choice	32	49	–	0.5	47.8	62.4	14.3	18.2
25	S/16	Experts Choice	128	196	–	0.5	50.2	66.7	15.8	18.5
26	S/16	Experts Choice	32	196	–	1.0	50.5	67.4	17.5	18.8
27	S/16	Experts Choice	128	1568	–	0.5	51.4	67.7	16.8	19.7
28	B/32	Experts Choice	32	392	–	1.0	49.9	66.0	16.5	19.7
29	B/32	Experts Choice	64	392	–	1.0	49.9	65.5	16.5	19.8
30	B/32	Tokens Choice	32	392	1	1.0	49.7	66.2	16.5	20.0
31	B/32	Tokens Choice	64	392	1	1.0	49.8	65.6	16.5	20.2
32	B/32	Soft MoE	64	49	–	–	51.8	70.7	17.8	20.3
33	B/32	Experts Choice	64	49	–	1.0	48.6	64.0	17.7	20.3
34	B/32	Experts Choice	32	49	–	1.0	48.4	63.8	17.7	20.5
35	S/16	Experts Choice	32	1568	–	1.0	51.3	68.7	21.5	21.5
36	S/16	Soft MoE	256	196	–	–	53.8	73.7	19.9	22.1
37	S/16	Tokens Choice	32	1568	1	1.0	51.2	68.9	21.5	23.2
38	S/16	Experts Choice	256	196	–	1.0	50.7	67.7	19.8	23.3
39	S/16	Experts Choice	32	196	–	2.0	51.0	68.3	23.1	23.5
40	B/32	Tokens Choice	32	392	2	1.0	50.2	67.4	22.0	23.6
41	B/32	Tokens Choice	64	392	2	1.0	50.8	68.0	22.1	23.8
42	S/16	Tokens Choice	64	1568	1	1.0	51.5	69.1	21.3	24.9
43	S/16	Experts Choice	256	1568	–	1.0	52.3	69.7	21.7	25.5
44	S/16	Experts Choice	32	1568	–	2.0	52.4	70.3	31.0	27.8
45	S/16	Tokens Choice	32	1568	2	1.0	52.1	70.3	31.0	30.0
46	B/32	Tokens Choice	64	392	2	2.0	51.2	70.0	33.2	30.4

Table 9: Model runs from Section 3.3 (shown in Pareto plot) trained for 300k steps on JFT with inverse square root decay and 50k steps cooldown. We trained dense and MoE (Soft MoE, Tokens Choice, Experts Choice) models with sizes S/32, S/16, S/8, B/32, B/16, L/32, L/16 and H/14. Sorted by increasing training TPUv3 days.

Ref	Model	Routing	Experts	Group Size	K	C	JFT P@1	IN/10shot	Train exaFLOP	Train Days
47	B/32	Tokens Choice	32	392	2	2.0	51.0	69.5	33.6	31.1
48	S/16	Tokens Choice	64	1568	2	1.0	52.3	70.4	31.1	32.0
49	S/16	Tokens Choice	32	1568	2	2.0	52.8	71.4	50.0	42.5
50	S/16	Tokens Choice	64	1568	2	2.0	52.9	71.4	50.1	45.1
51	B/16	Dense	–	–	–	–	52.0	71.8	64.8	45.2
52	B/16	Soft MoE	128	196	–	–	55.3	77.0	59.0	46.8
53	B/16	Experts Choice	128	1568	–	0.5	53.7	73.0	59.0	48.2
54	B/16	Experts Choice	32	196	–	1.0	53.3	73.0	65.6	51.0
55	B/16	Experts Choice	128	196	–	0.5	52.5	72.2	58.8	52.6
56	L/32	Dense	–	–	–	–	51.3	70.9	55.9	54.9
57	L/32	Experts Choice	32	392	–	0.5	52.3	71.2	47.4	55.2
58	L/32	Experts Choice	32	49	–	0.5	51.1	70.6	49.8	55.7
59	L/32	Soft MoE	32	49	–	–	53.5	75.0	49.8	56.0
60	B/16	Experts Choice	32	1568	–	1.0	54.2	74.5	73.6	56.2
61	B/16	Tokens Choice	32	1568	1	1.0	53.7	74.4	73.6	57.8
62	B/16	Experts Choice	256	196	–	1.0	52.7	72.7	73.4	58.1
63	B/16	Soft MoE	256	196	–	–	55.8	78.0	73.7	58.2
64	B/16	Tokens Choice	64	1568	1	1.0	54.0	74.8	73.2	58.7
65	L/32	Experts Choice	64	392	–	1.0	52.7	72.1	56.9	60.4
66	B/16	Experts Choice	256	1568	–	1.0	53.9	73.5	73.8	60.5
67	L/32	Experts Choice	32	392	–	1.0	52.7	71.7	56.8	60.6
68	L/32	Tokens Choice	64	392	1	1.0	51.9	71.4	56.9	61.0
69	L/32	Tokens Choice	32	392	1	1.0	52.3	71.7	57.1	61.6
70	L/32	Experts Choice	64	49	–	1.0	51.1	70.7	61.6	62.6
71	L/32	Soft MoE	64	49	–	–	54.0	75.2	61.7	62.8
72	L/32	Experts Choice	32	49	–	1.0	51.4	70.3	61.5	63.2
73	B/16	Experts Choice	32	196	–	2.0	53.1	73.9	86.8	64.2
74	L/32	Tokens Choice	32	392	2	1.0	51.5	70.7	76.0	72.2
75	B/16	Experts Choice	32	1568	–	2.0	54.6	75.6	102.9	72.5
76	L/32	Tokens Choice	64	392	2	1.0	52.0	71.8	76.0	72.5
77	B/16	Tokens Choice	32	1568	2	1.0	53.9	74.7	102.9	74.7
78	B/16	Soft MoE	512	196	–	–	56.1	78.5	103.1	76.5
79	B/16	Tokens Choice	64	1568	2	1.0	54.3	74.8	103.0	76.5
80	S/8	Dense	–	–	–	–	49.9	66.7	82.7	77.7
81	S/8	Soft MoE	512	784	–	–	56.1	78.0	85.6	88.5
82	S/8	Experts Choice	32	784	–	1.0	52.9	72.6	91.3	93.0
83	L/32	Tokens Choice	64	392	2	2.0	52.9	72.9	114.3	93.2
84	L/32	Tokens Choice	32	392	2	2.0	52.5	72.5	115.7	95.8
85	L/16	Dense	–	–	–	–	54.8	77.8	226.9	100.9
86	L/16	Experts Choice	128	196	–	0.5	54.0	76.7	204.6	104.9
87	L/16	Soft MoE	128	196	–	–	57.2	80.3	205.0	106.0
88	B/16	Tokens Choice	32	1568	2	2.0	54.4	75.7	161.4	108.4
89	B/16	Tokens Choice	64	1568	2	2.0	54.8	76.0	161.5	110.5
90	L/16	Experts Choice	32	196	–	1.0	55.1	77.5	228.6	113.6
91	L/16	Tokens Choice	32	1568	1	1.0	55.9	78.5	250.4	125.1
92	L/16	Tokens Choice	64	1568	1	1.0	56.2	78.6	248.8	125.7
93	S/8	Experts Choice	32	6272	–	1.0	53.6	73.4	160.6	126.6
94	S/8	Experts Choice	512	784	–	1.0	53.4	72.4	104.1	129.0

Table 9: Model runs from Section 3.3 (shown in Pareto plot) trained for 300k steps on JFT with inverse square root decay and 50k steps cooldown. We trained dense and MoE (Soft MoE, Tokens Choice, Experts Choice) models with sizes S/32, S/16, S/8, B/32, B/16, L/32, L/16 and H/14. Sorted by increasing training TPUv3 days.

Ref	Model	Routing	Experts	Group Size	K	C	JFT P@1	IN/10shot	Train exaFLOP	Train Days
95	L/16	Soft MoE	256	196	–	–	57.4	80.2	256.0	129.6
96	S/8	Tokens Choice	32	6272	1	1.0	53.8	73.7	162.5	129.8
97	L/16	Experts Choice	256	196	–	1.0	54.1	76.7	255.2	130.1
98	L/16	Experts Choice	32	196	–	2.0	55.2	77.8	301.0	140.3
99	S/8	Experts Choice	512	6272	–	1.0	54.8	74.6	149.3	161.9
100	S/8	Tokens Choice	32	6272	2	1.0	54.2	74.6	243.4	166.6
101	H/14	Soft MoE	128	256	–	–	58.0	81.6	599.2	170.5
102	H/14	Dense	–	–	–	–	56.5	80.1	680.5	196.2
103	H/14	Experts Choice	64	2048	–	1.25	57.3	80.4	855.9	210.9
104	L/16	Tokens Choice	32	1568	2	2.0	53.5	74.6	534.5	218.5
105	L/16	Tokens Choice	64	1568	2	2.0	53.3	73.3	535.1	226.9
106	H/14	Tokens Choice	64	2048	1	1.25	56.7	79.8	857.0	230.7
107	S/8	Tokens Choice	32	6272	2	2.0	54.1	74.8	424.4	255.4