Soft Merging of Experts with Adaptive Routing

Anonymous Authors[1]

Abstract

Sparsely-activated neural networks with conditional computation learn to route their inputs through different "expert" subnetworks, providing a strong structural prior and reducing computational costs. Despite their possible benefits, models with learned routing often underperform their parameter-matched densely-activated counterparts as well as models that use non-learned heuristic routing strategies. In this paper, we hypothesize that these shortcomings stem from the gradient estimation techniques used to train sparsely activated models with non-differentiable discrete routing decisions. To address this issue, we introduce **S**oft **M**erging of **E**xperts with **A**daptive **R**outing (SMEAR), which avoids discrete routing by using a single "merged" expert constructed via a weighted average of the experts' parameters. By routing activations through a single merged expert, SMEAR does not incur an increase in computational costs and facilitates standard gradient-based training. We empirically validate that the routing strategies learned via typical gradient estimation techniques underperform hand-designed heuristic strategies and that SMEAR outperforms both. Furthermore, we provide qualitative analysis demonstrating that the experts learned via SMEAR exhibit a significant amount of specialization.

1. Introduction

Neural networks typically use all of their parameters to process an example. This means that the computational cost of a neural network is often directly related to the number of parameters it has. However, there are cases where it might be appropriate to use a model architecture where different parts of the model are active for different inputs. Such an architecture can decouple the computational cost of a model from the number of parameters that it has. This possibility is increasingly useful given the current trend of scaling up models (Kaplan et al., 2020) because there may be cases where it is beneficial to train a model with more parameters but it is prohibitively expensive to train a typical densely-activated neural network (Fedus et al., 2021). Separately, specializing different parts of the model to different types of data may reduce interference and allocate capacity more effectively across downstream tasks (Sanh et al., 2021; Wei et al., 2021; Zamir et al., 2018; Bao et al., 2021) or languages (Pires et al., 2019; Liu et al., 2020; Xue et al., 2020).

Conditional computation techniques provide a possible way to attack these issues because they allow the network to selectively apply only a subset of its parameters to an input. A common way to use conditional computation is to introduce specialized subnetworks called *experts* that are controlled by *routers* that decide which experts should be active. As a result, a model with many experts can have a large number of parameters while incurring a lower computational cost by selecting a small number of experts to activate. When the model is trained with diverse data, this form of conditional computation can allow experts to specialize to different types of inputs while allowing flexible knowledge sharing across experts (Ma et al., 2019). However, because routing involves making a discrete decision as to which expert to use, the loss on final prediction cannot back-propagate though the routing decision to update the router. Consequently, models with conditional computation often require gradient estimation techniques for training (Clark et al., 2022; Fedus et al., 2021; Bengio et al., 2013).

In practice, past work has shown models with conditional computation do not always learn effective routing strategies. For example, Mittal et al. (2022) investigate models with a continuous router in a controlled setting and find the models do not route examples from the same group to the same experts, and perform poorly compared to models with oracle routing. However, models with task-specific modules (Gururangan et al., 2021; Kudugunta et al., 2021) provide evidence that it is possible to train performant models with specialized experts. As an extreme example, Roller et al. (2021) achieves results comparable to learned routing with a fixed random routing. Relatedly, Fedus et al. (2021) find the gain from scaling up parameters by 30× with a sparsely

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

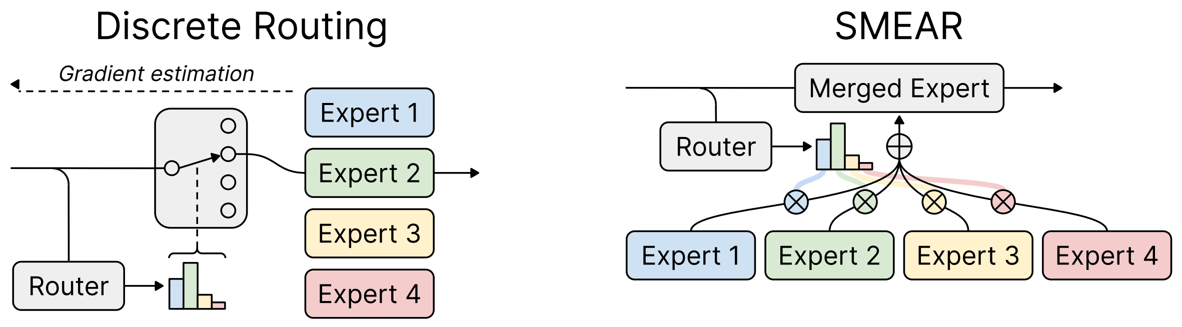


Figure 1. The discrete routing decisions commonly used in models that route activations among experts requires the use of gradient estimation (left). We propose SMEAR (right), which uses a given router's distribution to average the parameters of the corresponding experts and then routes the input through a single merged expert. SMEAR achieves better performance than models with discrete routing, can be trained with standard backpropagation, and does not incur significant additional computational costs compared to discrete routing.

activated model is smaller than scaling up both parameters and FLOPs by 3× in a dense model. As a possible explanation, Clark et al. (2022) characterize how models with conditional computation improve with scale and find a detrimental term that scales with the product of the log number of experts and active parameters. Consequently, increasing the number of experts yields limited returns and existing methods for training conditional computation models may only be helpful when the number of active parameters is moderate.

In this work, we hypothesize that issues with conditional computation stem from difficulties with gradient estimation. Specifically, we design experimental settings where we can compare learned routing to a performant hand-designed heuristic routing scheme. We find that all gradient estimation techniques that we consider produce models that underperform the heuristic routing, even in cases where a better routing strategy than the hand-designed one is possible. To address this shortcoming, we introduce **S**oft **M**erging of **E**xperts with **A**daptive **R**outing (SMEAR), a method for training models with specialized experts and learned routing. SMEAR works by using the router's distribution over experts to compute a weighted average of the parameters of the individual experts. Activations are then sent through the *merged* expert, which results in a similar computational cost to discrete routing to a single expert. However, the fact that all components of SMEAR are fully differentiable enables standard gradient-based training. Empirically, we show that SMEAR significantly outperforms discrete routing solutions found via gradient estimation as well as hand-designed heuristic routing schemes without incurring a significant increase in computational costs. We also qualitatively validate that the experts learned by SMEAR specialize to different types of inputs. Put together, our results show that SMEAR provides an effective alternative for models that use adaptive routing among expert subnetworks.

After providing the background on conditional computation models and gradient estimators in the following section, we define SMEAR in Section 3. We then describe our experimental findings in Section 4, discuss related works in Section 5 and conclude in Section 6.

2. Background

To provide the necessary background for our work, we first explain how sparsely activated neural networks use conditional computation, then discuss gradient estimators that enable learning routing strategies. In addition, we define the notion of "heuristic" routing strategies in settings where a performant routing can be hand-designed.

2.1. Routing Among Experts

In models that use discrete routing among experts (i.e. subnetwork modules), experts are typically organized into blocks and are incorporated into deep neural network architectures. An expert routing block B comprises a set of N experts $\{f_1, f_2, \dots f_N\}$ and a router R. Experts in the same block accept inputs of the same dimensionality. Given a hidden-state representation u, the output of the i-th expert with parameters θ_i is $f_i(u, \theta_i)$. In our work, the router chooses a single f_i to process the input of the block (though sparsely-activated models in other work may use more than one expert (Shazeer et al., 2017; Du et al., 2022)). Thus we can use the block B like any multi-layer building block in a neural network.

2.2. Gradient Estimators

In sparsely activated models that involve discrete adaptive routing, it is not possible to train the router's parameters with standard gradient-based learning. Fortunately, gradient estimators can provide approximate gradients to the router parameters. There are a few common designs shared by models that use gradient estimators to train routers. Their router R often applies a lightweight network to some intermediate hidden states v in the model rather than the original input to the full model. The lightweight routing network yields a probability distribution $P(v)$ over all the N experts. Different gradient estimators vary in how they make the routing decision from P and how they construct the output from the chosen expert. Additionally, some estimators may introduce additional loss terms.

REINFORCE Gradients can be estimated through discrete operations through reinforcement learning techniques (Schulman et al., 2015; Bengio et al., 2013). In reinforcement learning, a policy loss is used to train an agent to learn optimal actions in an environment. In this paper, we experiment with the REINFORCE algorithm which computes the policy loss as $\log(\pi)r$ where r denotes the received reward for taking an action whose assigned probability is π. When applied to models that use discrete routing among experts, the goal is to train the model to choose which expert to use to process a given input. Here, the router R acts an agent that samples an expert to use according to the routing probabiltiies. In order to train such a router, the router's assigned probability to the sampled expert is used as π and the negative of the model's loss is used as the reward r. The router is therefore trained to pick experts that maximize the reward which, in turn, minimizes the loss. The REINFORCE estimator often suffers from high variance because of the sampling operation. This motivates the use of baselines, which reduce variance without changing the optimal solution. In our work, we use a baseline introduced by Clark et al. (2022), where the baseline b is generated by a small neural network with a single hidden layer that takes as input v and is trained with Huber loss. The overall loss function is then

$$L = \mathbb{E}_{i \sim P(v)} \alpha \log P(v)_i (r - b) - \quad (1)$$

$$\beta P(v) \log P(v) + \gamma L_{\text{Huber}}(r, b) \quad (2)$$

where $P(v)$ is the routing probability distribution and α, β, and γ are hyperparameters that correspond to policy gradient weight, policy entropy weight, and value loss weight. Finally, the output of the block B is just $f_i(u, \theta_i)$.

Straight Through Gumbel-Softmax (ST-Gumbel) The Gumbel-Softmax trick (Jang et al., 2016) provides a continuous differentiable approximation to sampling from a categorical distribution like the one produced by a router. Specifically, Gumbel noise is added to the logits of the distribution and a temperature scale is applied in the softmax operation. Denoting $g_i \sim \text{Gumbel}(0, 1)$ and τ as the temperature, the Gumbel-Softmax trick produces the following modified distribution:

$$\hat{P}(v)_i = \frac{\exp((\log(P(v)_i) + g_i)/\tau)}{\sum_{j=1}^{N} \exp((\log(P(v)_i) + g_i)/\tau)} \quad (3)$$

The expert f_i with the highest assigned probability is chosen by applying an argmax operation over this distribution. In order to approximate gradients through the argmax operation, we use the Straight-Through estimator which replaces $f_i(u, \theta_i)$ with $(1 - \text{sg}[\hat{P}(v)_i] + \hat{P}(v)_i) f_i(u, \theta_i)$ where sg stands for the stop-gradient operator. During forward pass, the multiplier for $f_i(u, \theta_i)$ becomes 1 and the multiplier receives gradients for the term $\hat{P}(v)_i$ in the backward pass. In practice, the temperature τ is gradually annealed from a high to low value so that the approximated samples are more and more similar to discrete samples.

Top-k Shazeer et al. (2017) propose a gradient estimation scheme where the router sends the input through the k experts that are assigned the highest probability. Fedus et al. (2021) later found that this router could be used effectively when $k = 1$. Specifically, the estimator selects the subnetwork with the highest probability and scales its output using its corresponding routing probability. The output of the block is therefore $P(v)_i f_i(u, \theta_i)$, where $i = \text{argmax}_i(P(v))$.

2.3. Heuristic Routing

As a point of comparison for techniques that learn adaptive routing, we experiment with three baseline routing strategies that do not require a trained router.

Tag Routing If we have prior knowledge about the data that a model will be applied to, it can be possible to hand-design a heuristic routing strategy for choosing which experts to use for a given example based on data properties. Tag routing takes advantage of tags associated with the examples (such as its domain or task in multitask learning) and associates each expert in a given expert routing block with a particular tag. In this work, we assume each example has a single tag. As such, examples are routed to the expert corresponding to their tag.

Hash Routing Roller et al. (2021) propose hash routing, which uses a fixed hashing function to determine which expert to use for a given example. Specifically, each example is assigned a random expert choice in each expert routing block which is used consistently over the course of training. This approach disregards any shared characteristics across examples.

Monolithic Routing As a baseline, we consider models where each expert routing block only has a single expert. This provides an important point of comparison as it is a degenerate solution that can be found with learned routing by having the router always choose the same expert for all examples.

3. Soft Merging of Experts with Adaptive Routing

As we will later show in Section 4, the gradient estimation techniques used to train models with discrete routing among experts often fail to produce performant routing strategies. Our goal in this work is therefore to explore whether it is possible to train models with adaptive routing among experts without resorting to gradient estimation. Ideally, we would be able to design an expert and router architecture that facilitates standard gradient-based training so that the model could be trained end-to-end in a standard fashion.

Ensemble Routing One simple idea would be to pass the input of a given expert routing block through *every* expert, and then compute an average of the experts' outputs weighted according the router's distribution, i.e. exactly computing $\mathbb{E}_{i\sim P(v)} f_i(u, \theta_i)$. We refer to this approach as an *ensemble* routing strategy since it corresponds to using the ensemble prediction of the experts. Since the operations involved in computing the average are all differentiable, using an ensemble routing strategy would allow for exact computation of gradients and end-to-end-learning. Unfortunately, such an approach would incur a significant increase in computational costs because it requires computing the output of every expert rather than a single expert.

Merging Experts To explore an alternative fully-differentiable expert routing block, we take inspiration from recent work on *merging* models (Matena & Raffel, 2021; Wortsman et al., 2022b;c; Choshen et al., 2022; Don-Yehiya et al., 2022). These works have shown that averaging the parameters of models that share a common architecture can often produce an aggregate model that shares the capabilities of the individual models. For example, Wortsman et al. (2022b) found that averaging the weights of multiple fine-tuned models produced a single model that performs comparably to an ensemble of the models. Motivated by these findings, we propose **S**oft **M**erging of **E**xperts with **A**daptive **R**outing (SMEAR), which constructs a single merged expert whose parameters are computed as the weighted average of the experts within a routing block. Each expert's weight is set according to the corresponding routing probability generated by the router. In SMEAR, the input to the routing block is fed into the merged expert, whose output is used as the output of the block. SMEAR implicitly assumes that all experts in the routing block share an identical architecture (thereby inducing a natural one-to-one mapping between parameters in each expert). To the best of our knowledge, all past works focused on routing among experts use experts with identical architecture.

More explicitly, we define SMEAR as computing the output of an expert routing block using a merged expert computed as $\bar{f}(u, \sum_i P(v)_i \theta_i)$. The merged expert shares the same architecture with the individual experts f_i. Notably, the input of the routing block is only ever processed by $\bar{f}$; activations are never fed to any of the individual experts. To break symmetry, all experts are randomly initialized with different parameter values. Importantly, all operations in SMEAR are fully differentiable, SMEAR enables standard gradient-based end-to-end learning. In addition, SMEAR retains the ability to learn an adaptive routing strategy that can intelligently route different examples to different experts. We will later show qualitatively that this leads to meaningful specialization of different experts in real-world experiments (Section 4.3).

Computational Costs Importantly, SMEAR only ever computes the output of a single expert. This suggests that the computational costs of SMEAR could be comparable to using discrete routing and significantly lower than ensemble routing. However, we note that the averaging operation itself incurs a nontrivial computational cost. To quantify this cost, we focus on the common expert architecture comprising a dense layer that projects from d-dimensional activations to a m dimensional vector followed by a nonlinearity and finally an additional dense layer projecting from m dimensions back to d. For simplicity, we ignore the cost of the nonlinearity since it introduces a relatively small computational cost. We focus on the setting where the input is a sequence of length L so that the input to an expert routing block is a sequence of activations of size $L \times d$. In this case, computing the output of a single expert incurs a computational cost of approximately $L \times 4 \times d \times m$ FLOPs and ensemble routing with N experts would require $N \times L \times 4 \times d \times m$ FLOPs. SMEAR also requires only $L \times 4 \times d \times m$ to compute the output of the merged expert, but must average together the parameters of N experts. Computing this average once incurs an additional cost of approximately $N \times 2 \times d \times m$. Some past work on models with discrete routing has the router choose a different expert for each entry in the input sequence of activations (e.g. Fedus et al., 2021; Lewis et al., 2021; Roller et al., 2021). This would require computing the expert average L times, which would make the cost of SMEAR similar to that of ensemble routing. We therefore focus on settings where models make a *single* routing choice for an entire input example (e.g. Shazeer et al., 2017; Gururangan et al., 2021; Kudugunta et al., 2021; Ye et al., 2022). This results in a total cost of approximately $(L \times 4 + N \times 2) \times d \times m$ for SMEAR. Consequently, as long as $L \times 4 \gg N \times 2$, SMEAR and discrete routing have roughly the same computational costs. Furthermore, we would expect SMEAR to be approximately $\frac{N \times L}{N + L}$ times cheaper than ensemble routing. More concretely, we find in Section 4.2 that the wall-clock time required to process an example with SMEAR in real-world experiments is roughly the same as using discrete routing and significant faster than ensemble routing.

4. Experiments

In order to thoroughly evaluate the effectiveness of SMEAR in addressing the limitations of models that learn discrete routing through gradient estimation, we perform experiments in two real-world settings that differ in model architecture and modality. We are particularly interested in whether a given approach for learning routing outperforms the heuristic routing strategies describe in Section 2.3. As such, we focus on experimental settings where a performant "tag routing" baseline can be designed, i.e. where we have metadata that can be used to appropriate route examples. Specifically, we experiment with fine-tuning T5 1.1 Base (Raffel et al., 2020) on datasets from GLUE (Wang et al., 2018) (referred to as T5-GLUE) and fine-tuning a ResNet18 (He et al., 2016) on DomainNet (Peng et al., 2019) (ResNet-DomainNet). In these settings, we add experts to an existing pre-trained backbone in the same way that Adapters are used for parameter-efficient fine-tuning (Houlsby et al., 2019).

Routing	T5-GLUE	ResNet-DomainNet
Tag	78.5	61.5
Tag+	79.8	–
Hash	67.4	52.5
Monolithic	77.5	59.0
Top-k	77.9	60.0
ST-Gumbel	76.1	58.3
REINFORCE	78.4	59.8
SMEAR	**82.5**	**62.0**
Expert ensemble	82.8	62.5

Table 1. Average performance of models trained using different routing strategies. Discrete routing strategies learned through gradient estimators (Top-k, ST-Gumbel, REINFORCE) tend to underperform tag-based heuristic routing but outperform degenerate strategies (Hash, Monolithic). SMEAR outperforms all other routing strategies with comparable computational cost – an expert ensemble routing strategy (greyed out) is significantly more expensive (cf. Figure 2).

T5-GLUE In this scenario, we focus on training a T5 model (Raffel et al., 2020) on the GLUE meta-benchmark (Wang et al., 2018) for natural language understanding. GLUE consists of nine datasets ranging across sentimental analysis (SST-2 (Socher et al., 2013)), acceptability judgement (CoLA (Warstadt et al., 2019)), natural language inference (MNLI (Williams et al., 2017), RTE (Bentivogli et al., 2009)), semantic similarity (QQP[1], MRPC (Dolan & Brockett, 2005), STS-B (Cer et al., 2017)), question answering (QNLI (Rajpurkar et al., 2016)), and commonsense reasoning (WNLI (Levesque et al., 2012)). Following convention, we exclude WNLI and use the remaining 8 datasets. We use the prompted form of these datasets available in PromptSource (Bach et al., 2022), maps each example into a natural language request-and-response form. During training, we randomly select a prompt template for each example from the PromptSource templates for the example's dataset and we evaluate each example using all of the datatset's templates. We base our implementation on Mahabadi et al. (2021) for splitting train, eval, and test sets of GLUE datasets. When using tag routing, we consider the dataset of each example as the tag. We use the pretrained T5 1.1 Base model as the backbone and adapt the model in a way similar to adding adapters (Houlsby et al., 2019) for a single task, i.e. we keep all pretrained parameters frozen except for layer normalization parameters and insert expert routing blocks after self-attention, feed-forward and cross-attention modules. The T5 1.1 Base model has 12 Transformer layers in both the encoder and decoder, resulting in a total of $12 \times 2 = 24$ blocks in the encoder and $12 \times 3 = 36$ blocks in the decoder, totally 60 expert routing blocks. In each block, we introduce eight experts (one for each dataset in GLUE). The router architecture comprises a layer normalization layer, a linear layer, and a softmax nonlinearity to generate the routing probability distribution. During the forward pass, each vector in the linear layer is rescaled using a separate layernorm to avoid unbounded growth which would result in the router consistently outputting a one-hot distribution. In the encoder, the router takes as input the preceding hidden states, which are averaged across the sequence and fed into the router. In the decoder, the routers receive the average of the encoder's final hidden states instead of the decoder hidden states to prevent information leakage from later target tokens to earlier target tokens. We also experimented with expert dropout of 0.1, following Komatsuzaki et al. (2022) where each expert is dropped with a probability of 0.1. Our results indicate that expert dropout is beneficial for SMEAR and Top-K methods, but not for other methods. This aligns with our observation that these two methods have less exploration than REINFORCE and ST-Gumbel, as the latter methods do sampling for exploration. Therefore, the results presented in Table 1 include expert dropout for SMEAR and the Top-K method. A detailed ablation of expert dropout can be found in Table 2.

ResNet-DomainNet In this scenario, we focus on adapting an ImageNet pre-trained ResNet18 (He et al., 2016) to datasets within DomainNet (Peng et al., 2019). DomainNet is a collection of object recognition datasets that cover six distinct domains that all share the same label space corresponding to 345 object categories. We treat the domain of each example as its tag. As in the T5-GLUE scenario, we freeze the pretrained model, and insert eight expert routing blocks after each of the eight residual layer groups in the

[1]https://quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs

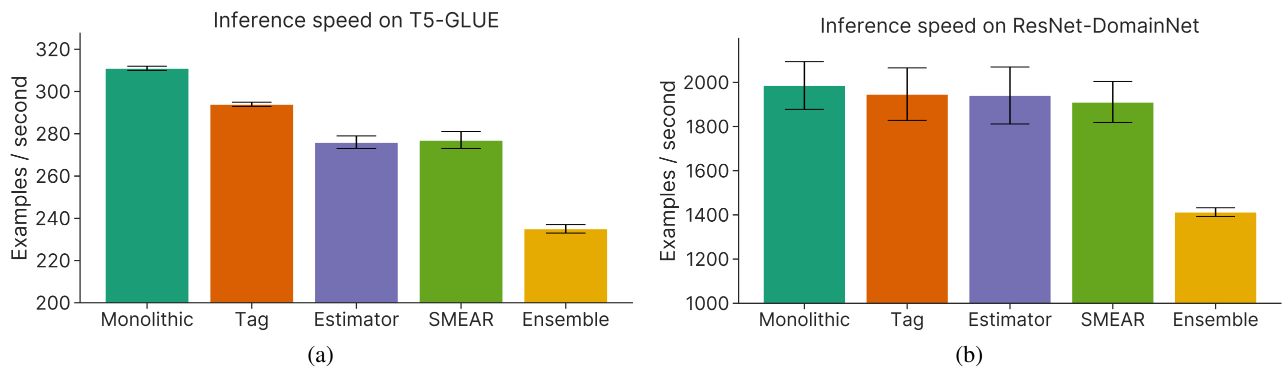


Figure 2. Comparison of inference speed for various routing strategies in T5-GLUE (a) and ResNet-DomainNet (b). SMEAR has comparable speed with that of discrete routing with estimators, whereas computing an ensemble of experts ("Ensemble") is the slowest.

model. Each block includes six experts corresponding to the number of domains. We use the same architecture for routers as in T5-GLUE and feed average-pooled hidden states into the router to compute the routing probability. Experts in this setting use batchnorm on their input instead of layer norm in the output, following (Rebuffi et al., 2017).

In each scenario, we compare SMEAR to learned routing using the gradient estimators from Section 2.2 and heuristic routing strategies from Section 2.3. Full details of hyperparameters and training timings for each scenario are presented in Appendix C. For scenarios with multiple datasets, we only provide the average performance across datasets in the main paper due to space limitations. Full results are provided in Appendix E.

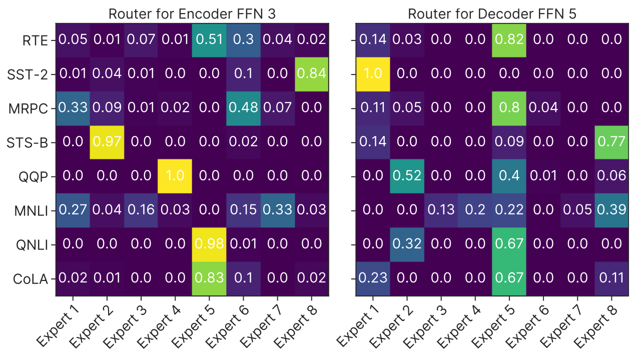
4.1. Results

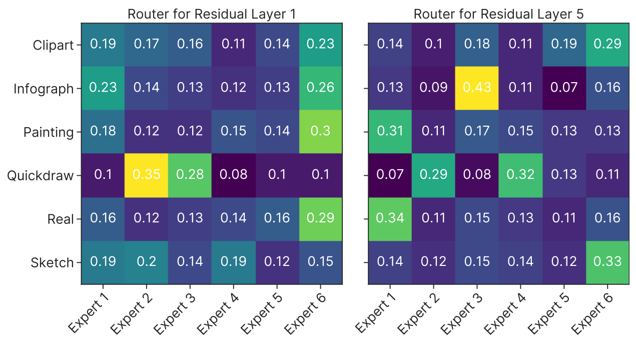
To assess the overall effectiveness of routing strategies learned with SMEAR, we compare to the performance attained by sparsely activated models trained with the gradient estimators described in Section 2.2 and models with different heuristic routing strategies described in Section 2.3. A summary of our results is shown in Table 1. First, we find that models using routing strategies learned through gradient estimators do not outperform tag routing in either settings. Specifically, the best-performing estimator (REINFORCE) in T5-GLUE slightly underperforms tag routing and the best performing estimator (Top-k) in ResNet-DomainNet underperforms tag routing by 1.5%. Learned routing with estimators except ST-Gumbel do better than hash and monolithic routing in both settings, which suggests that estimators are learning nontrivial routing strategies that are nevertheless less effective than tag routing. Pertinently, in all experimental settings, SMEAR outperforms every other routing strategy, including both routing learned by gradient estimators and all heuristic routing strategies. In particular, SMEAR achieves 4% improvement over tag routing in T5-GLUE and 0.5% improvement over tag routing in ResNet-DomainNet. As an upper bound on performance, we also compare SMEAR to expert ensembling ("Ensemble") which averages the outputs of all experts and incurs significantly higher computational cost. SMEAR underforms expert ensemble by 0.3% in T5-GLUE and 0.5% in ResNet-DomainNet. Whether expert ensembling corresponds to an attainable upper bound for other learned routing methods is unclear since it makes use of more computation.

To get a better sense of the ways that learned routing can outperform tag routing, we sought to design an improved tag routing strategy. Based on prior results in transfer learning on GLUE that show that intermediate- or multi-task training on MNLI and RTE can improve performance on RTE (Phang et al., 2018; Devlin et al., 2018; Pruksachatkun et al., 2020; Vu et al., 2020), we designed an additional tag-based routing scheme (called "Tag+"), where examples from RTE and MNLI are routed to the same expert in all the routing blocks of the encoder. We find that SMEAR outperforms the Tag+ strategy by 2.7%. This suggests that SMEAR may be able to uncover and exploit the beneficial commonality between the tasks for different examples without any supervision or metadata. We further explore the routing decisions and expert specialization in Section 4.3 and find evidence of emergent task-relevant structure in routing decisions.

4.2. Inference speed

To compare the inference speed of the various methods, we measure the number of examples processed per second during inference in both experimental settings as shown in Figure 2. Monolithic routing has the fastest inference speed as all examples utilize only one expert. Tag routing is slower than Monolithic routing as the examples in a batch must be routed to the corresponding experts based on tags. Learned routing with gradient estimators operate similarly during inference, selecting the expert with the highest probability. For the sake of comparison, we compute using only one estimator and refer to it as the Estimator method. The Estimator





(a) (b)

Figure 3. Average routing distributions produced by SMEAR for two routers from the T5-GLUE model (a) and two from the ResNet-DomainNet model (b). For a given router, we average all routing distributions across all examples from a given dataset.

method has a slightly slower speed than tag routing due to the additional computation required in the router. Despite the slight overhead of averaging the weights in SMEAR, we observe that its inference speed is almost identical to that of routing with estimators. The Ensemble method performs poorly in terms of speed, with a 1.2x slowdown in T5-GLUE and 1.4x slowdown in ResNet-DomainNet compared to SMEAR. In summary, the SMEAR method outperforms heuristic routing and learned routing with estimators with almost no significant change in inference speed. While the Ensemble method has high performance, its high inference cost makes it impractical for larger models.

4.3. Qualitative Analysis

In this section, we provide qualitative analysis of the routing learned by SMEAR by visualizing the average router distribution across all examples in a given dataset for every router in each model. Figure 3 illustrates the routing distributions for two routers from the model trained in T5-GLUE and two from ResNet-DomainNet. For the T5-GLUE routers, we observe significantly different behavior – one mainly follows a tag routing-style strategy whereas the other routes most datasets to the same expert. However, we note that the tag-style router utilizes shared experts for RTE, MRPC, and MNLI; notably, these tasks are somewhat similar in that they all involve determining similarity among pairs of sentences. In the monolitic-style router, STS-B (the only regression task) and SST-2 (which has a distinct output vocabulary) are given dedicated experts, and MNLI (a large and relatively challenging dataset) is routed through many different experts. More broadly, we highlight that there is generally a great deal of sparsity in the learned routing distributions, suggesting a significant amount of expert specialization. In ResNet-DomainNet, we can see that examples from the Quickdraw domain are routed to two specific experts in both cases. Additionally, we observe that the router distribution of the Painting and Real domains are highly correlated. Other domains such as Clipart, Sketch seem to evenly use experts. Interestingly, there is less expert specialization in the ResNet-DomainNet model, suggesting that there may be more similarities between the individual domains in DomainNet compared to the tasks in GLUE. Routing distributions for all routers can be found in Appendix F.

5. Related Work

Models with Conditional Computation Various works have investigated ways to circumvent the difficulties in routing in multi-task learning settings. For example, Deecke et al. (2020); Hazimeh et al. (2021); Dua et al. (2021) start training with most of the experts activated and gradually introduce sparsity. Kudugunta et al. (2021); Ponti et al. (2022); Ma et al. (2019); Gupta et al. (2022) group examples from the same task together and introduce task-specific parameters in the router. Some works avoid learned routing by hand-crafting heuristic routing strategies. Gururangan et al. (2021) built sparsely activated language models where different domains use separate experts. On an unknown domain, the model assesses the experts' fitness to combine the experts. Tang et al. (2022); Pfeiffer et al. (2022; 2020) specify assign experts based on task-related human knowledge. Li et al. (2022a) demonstrate that models structured as sparse mixture-of-experts generalize effectively to novel domains, as compared to other domain generalization algorithms in vision transformers. Our focus on settings where performant routing schemes can be hand-designed takes inspiration from this line of work.

Because sparsely activated models disentangle computation and parameter count, significant effort has gone into leveraging conditional computation to create massive pre-trained models with a feasible computation cost (Fedus et al., 2022; Shazeer et al., 2017; Fedus et al., 2021; Du et al., 2022; Zoph et al., 2022; Yu et al., 2022). Many works explore different routing methods in this setting, with a major focus on balancing the load across experts (Lewis et al., 2021; Zhou et al., 2022; Kool et al., 2021; Roller et al., 2021). Another line of work aims to introduce ways to convert trained dense models into similar-sized sparse models with a lower computational footprint (Lee-Thorp & Ainslie, 2022; Zhang et al., 2022; Komatsuzaki et al., 2022). We are interested in scaling up models using SMEAR in future work, but we currently lack the computational resources to do so.

Gradient Estimation Techniques Many gradient estimators have been proposed to produce approximate gradients for learning discrete representations that involve sampling. Our work uses a learned baseline from Clark et al. (2022) to reduce the variance of the REINFORCE estimator. The REBAR estimator (Tucker et al., 2017) adds a reparameterizable term to REINFORCE as a baseline that results in a more effective unbiased estimator. This additional term uses a relaxed sample similar to Gumbel-Softmax (Jang et al., 2016). RELAX (Grathwohl et al., 2017) is similar to REBAR but uses a learnable neural network for the reparameterizable term. Kool et al. (2019) uses additional samples from the policy as a built-in baseline for REINFORCE. Yin & Zhou (2018) and Dong et al. (2020) use the idea of coupling between multiple samples to reduce the variance of the gradient estimator and are designed for binary latent variables. Dong et al. (2021) improve upon Yin & Zhou (2018) and Dong et al. (2020) by extending the estimator to categorical variables. In preliminary experiments, we did not find any major gains from using more sophisticated gradient estimation techniques, but designing gradient estimators with discrete routing in mind could yield more performant routing strategies.

Issues with Conditional Computation Clark et al. (2022) study the scaling laws of sparse language models and discovered a computational cutoff above which no additional benefits are observed. Relatedly, Du et al. (2022) observe worse results when further scaling up the number of experts. Chi et al. (2022) and Dai et al. (2022) discover the representation collapse and routing fluctuation issue, respectively. Mittal et al. (2022) create a set of simple and modular data distributions, and show that systems with modular architecture can not find the most beneficial solution when trained end-to-end. Ye et al. (2022) experiment with various designs for multi-task learning with task-level routing and find that the performance still cannot surpass simple multi-task baselines. Our work demonstrates a possible way to avoid many of these issues by using a fully differentiable routing strategy that does not increase computational costs.

Weight Averaging Methods Our work takes inspiration from prior work that utilizes parameter averaging as an alternative to ensembling. For example, Wortsman et al. (2022c); Ilharco et al. (2022) average the weights of a pre-trained and a fine-tuned model to improve performance on target tasks as well as robustness to distribution shift. Choshen et al. (2022) similarly show that merging multiple models fine-tuned on different datasets can provide a better initialization than using the original pre-trained model for further fine-tuning on new unseen datasets.

Model averaging is also a common step in distributed optimization, where it is widely used in federated learning McMahan et al. (2017) and has recently been used for distributed fine-tuning (Wortsman et al., 2022a), multi-domain training (Li et al., 2022b), and multitask training (Don-Yehiya et al., 2022). There are also works that utilize different styles of merging instead of weight averaging of parameters, such as reweighting parameters in accordance with their approximate Fisher information (Matena & Raffel, 2021), aligning features by fitting a linear projection (Jin et al., 2022), and permuting columns to account for permutation symmetries (Ainsworth et al., 2022). We are interested in applying these more sophisticated merging methods to SMEAR in future work.

6. Conclusion

In this work, we sought to address shortcomings of models with discrete routing among experts that lead them to produce worse performance than heuristic non-learned routing. We hypothesized that these issues stem from the gradient estimation techniques required to propagate gradients through discrete routing decisions, and therefore focused on designing an expert routing architecture that faciliated exact gradients to be calculated. Our approach, called SMEAR, works by computing a weighted average of expert parameters where the weighting is set according to the output of a learned router. We compared the performance of models using SMEAR to models that were trained via various gradient estimation techniques to perform discrete routing. In experimental settings covering different modalities and model architectures, we found that SMEAR outperformed all models with discrete routing as well as performant heursitic routing strategies. Notably, this performance boost comes with no increase in computational costs. Through qualitative analysis, we further confirmed that the experts learned in a model using SMEAR specialize to different types of inputs while the router learns a nontrivial strategy that exploits commonalities across different examples. In future work, we are interested in exploring different expert architectures (Liu et al., 2022; hu2) and improved merging methods (Matena & Raffel, 2021; Ainsworth et al., 2022; Jin et al., 2022). Given the poor scaling properties of models with discrete routing (Clark et al., 2022), we would also be excited to try out SMEAR in the same large-scale settings where discrete routing has been used (Fedus et al., 2021; Zoph et al., 2022; Du et al., 2022).

References

Ainsworth, S. K., Hayase, J., and Srinivasa, S. Git re-basin: Merging models modulo permutation symmetries. *arXiv preprint arXiv:2209.04836*, 2022.

Bach, S. H., Sanh, V., Yong, Z.-X., Webson, A., Raffel, C., Nayak, N. V., Sharma, A., Kim, T., Bari, M. S., Fevry, T., et al. Promptsource: An integrated development environment and repository for natural language prompts. *arXiv preprint arXiv:2202.01279*, 2022.

Bao, H., Dong, L., and Wei, F. Beit: Bert pre-training of image transformers. *arXiv preprint arXiv:2106.08254*, 2021.

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

Bentivogli, L., Clark, P., Dagan, I., and Giampiccolo, D. The fifth pascal recognizing textual entailment challenge. In *TAC*, 2009.

Cer, D., Diab, M., Agirre, E., Lopez-Gazpio, I., and Specia, L. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*, 2017.

Chi, Z., Dong, L., Huang, S., Dai, D., Ma, S., Patra, B., Singhal, S., Bajaj, P., Song, X., and Wei, F. On the representation collapse of sparse mixture of experts. *arXiv preprint arXiv:2204.09179*, 2022.

Choshen, L., Venezian, E., Slonim, N., and Katz, Y. Fusing finetuned models for better pretraining. *arXiv preprint arXiv:2204.03044*, 2022.

Clark, A., de Las Casas, D., Guy, A., Mensch, A., Paganini, M., Hoffmann, J., Damoc, B., Hechtman, B., Cai, T., Borgeaud, S., et al. Unified scaling laws for routed language models. In *International Conference on Machine Learning*, pp. 4057–4086. PMLR, 2022.

Dai, D., Dong, L., Ma, S., Zheng, B., Sui, Z., Chang, B., and Wei, F. Stablemoe: Stable routing strategy for mixture of experts. *arXiv preprint arXiv:2204.08396*, 2022.

Deecke, L., Hospedales, T., and Bilen, H. Latent domain learning with dynamic residual adapters. *arXiv preprint arXiv:2006.00996*, 2020.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Dolan, B. and Brockett, C. Automatically constructing a corpus of sentential paraphrases. In *Third International Workshop on Paraphrasing (IWP2005)*, 2005.

Don-Yehiya, S., Venezian, E., Raffel, C., Slonim, N., Katz, Y., and Choshen, L. Cold fusion: Collaborative descent for distributed multitask finetuning. *arXiv preprint arXiv:2212.01378*, 2022.

Dong, Z., Mnih, A., and Tucker, G. Disarm: An antithetic gradient estimator for binary latent variables. *Advances in neural information processing systems*, 33:18637–18647, 2020.

Dong, Z., Mnih, A., and Tucker, G. Coupled gradient estimators for discrete latent variables. *Advances in Neural Information Processing Systems*, 34:24498–24508, 2021.

Du, N., Huang, Y., Dai, A. M., Tong, S., Lepikhin, D., Xu, Y., Krikun, M., Zhou, Y., Yu, A. W., Firat, O., et al. Glam: Efficient scaling of language models with mixture-of-experts. In *International Conference on Machine Learning*, pp. 5547–5569. PMLR, 2022.

Dua, D., Bhosale, S., Goswami, V., Cross, J., Lewis, M., and Fan, A. Tricks for training sparse translation models. *arXiv preprint arXiv:2110.08246*, 2021.

Fedus, W., Zoph, B., and Shazeer, N. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity, 2021.

Fedus, W., Dean, J., and Zoph, B. A review of sparse expert models in deep learning. *arXiv preprint arXiv:2209.01667*, 2022.

Grathwohl, W., Choi, D., Wu, Y., Roeder, G., and Duvenaud, D. Backpropagation through the void: Optimizing control variates for black-box gradient estimation. *arXiv preprint arXiv:1711.00123*, 2017.

Gupta, S., Mukherjee, S., Subudhi, K., Gonzalez, E., Jose, D., Awadallah, A. H., and Gao, J. Sparsely activated mixture-of-experts are robust multi-task learners. *ArXiv*, abs/2204.07689, 2022.

Gururangan, S., Lewis, M., Holtzman, A., Smith, N. A., and Zettlemoyer, L. Demix layers: Disentangling domains for modular language modeling. *arXiv preprint arXiv:2108.05036*, 2021.

Hazimeh, H., Zhao, Z., Chowdhery, A., Sathiamoorthy, M., Chen, Y., Mazumder, R., Hong, L., and Chi, E. Dselect-k: Differentiable selection in the mixture of experts with applications to multi-task learning. *Advances in Neural Information Processing Systems*, 34:29335–29347, 2021.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., and Gelly, S. Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pp. 2790–2799. PMLR, 2019.

Ilharco, G., Wortsman, M., Gadre, S. Y., Song, S., Hajishirzi, H., Kornblith, S., Farhadi, A., and Schmidt, L. Patching open-vocabulary models by interpolating weights. *arXiv preprint arXiv:2208.05592*, 2022.

Jang, E., Gu, S., and Poole, B. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Jin, X., Ren, X., Preotiuc-Pietro, D., and Cheng, P. Dataless knowledge fusion by merging weights of language models. *arXiv preprint arXiv:2212.09849*, 2022.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Komatsuzaki, A., Puigcerver, J., Lee-Thorp, J., Ruiz, C. R., Mustafa, B., Ainslie, J., Tay, Y., Dehghani, M., and Houlsby, N. Sparse upcycling: Training mixture-of-experts from dense checkpoints. *arXiv preprint arXiv:2212.05055*, 2022.

Kool, W., van Hoof, H., and Welling, M. Buy 4 reinforce samples, get a baseline for free! 2019.

Kool, W., Maddison, C. J., and Mnih, A. Unbiased gradient estimation with balanced assignments for mixtures of experts. *arXiv preprint arXiv:2109.11817*, 2021.

Kudugunta, S., Huang, Y., Bapna, A., Krikun, M., Lepikhin, D., Luong, M.-T., and Firat, O. Beyond distillation: Task-level mixture-of-experts for efficient inference. *arXiv preprint arXiv:2110.03742*, 2021.

Lee-Thorp, J. and Ainslie, J. Sparse mixers: Combining moe and mixing to build a more efficient bert. *arXiv preprint arXiv:2205.12399*, 2022.

Levesque, H., Davis, E., and Morgenstern, L. The winograd schema challenge. In *Thirteenth international conference on the principles of knowledge representation and reasoning*, 2012.

Lewis, M., Bhosale, S., Dettmers, T., Goyal, N., and Zettlemoyer, L. Base layers: Simplifying training of large, sparse models. In *International Conference on Machine Learning*, pp. 6265–6274. PMLR, 2021.

Li, B., Yang, J., Ren, J., Wang, Y., and Liu, Z. Sparse fusion mixture-of-experts are domain generalizable learners. *arXiv preprint arXiv:2206.04046*, 2022a.

Li, M., Gururangan, S., Dettmers, T., Lewis, M., Althoff, T., Smith, N. A., and Zettlemoyer, L. Branch-train-merge: Embarrassingly parallel training of expert language models. *ArXiv*, abs/2208.03306, 2022b.

Liu, H., Tam, D., Muqeeth, M., Mohta, J., Huang, T., Bansal, M., and Raffel, C. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *arXiv preprint arXiv:2205.05638*, 2022.

Liu, Y., Gu, J., Goyal, N., Li, X., Edunov, S., Ghazvininejad, M., Lewis, M., and Zettlemoyer, L. Multilingual denoising pre-training for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8: 726–742, 2020.

Ma, J., Zhao, Z., Chen, J., Li, A., Hong, L., and Chi, E. H. Snr: Sub-network routing for flexible parameter sharing in multi-task learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 216–223, 2019.

Mahabadi, R. K., Ruder, S., Dehghani, M., and Henderson, J. Parameter-efficient multi-task fine-tuning for transformers via shared hypernetworks. *arXiv preprint arXiv:2106.04489*, 2021.

Matena, M. and Raffel, C. Merging models with fisher-weighted averaging. *arXiv preprint arXiv:2111.09832*, 2021.

McMahan, B., Moore, E., Ramage, D., Hampson, S., and Arcas, B. A. y. Communication-efficient learning of deep networks from decentralized data. In *Artificial Intelligence and Sstatistics*, 2017.

Mittal, S., Bengio, Y., and Lajoie, G. Is a modular architecture enough? *arXiv preprint arXiv:2206.02713*, 2022.

Peng, X., Bai, Q., Xia, X., Huang, Z., Saenko, K., and Wang, B. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1406–1415, 2019.

Pfeiffer, J., Vulic, I., Gurevych, I., and Ruder, S. Mad-x: An adapter-based framework for multi-task cross-lingual transfer. In *EMNLP*, 2020.

Pfeiffer, J., Goyal, N., Lin, X. V., Li, X., Cross, J., Riedel, S., and Artetxe, M. Lifting the curse of multilinguality by pre-training modular transformers. In *NAACL*, 2022.

Phang, J., Févry, T., and Bowman, S. R. Sentence encoders on stilts: Supplementary training on intermediate labeled-data tasks. *arXiv preprint arXiv:1811.01088*, 2018.

Pires, T., Schlinger, E., and Garrette, D. How multilingual is multilingual bert? *arXiv preprint arXiv:1906.01502*, 2019.

Ponti, E. M., Sordoni, A., and Reddy, S. Combining modular skills in multitask learning. *arXiv preprint arXiv:2202.13914*, 2022.

Pruksachatkun, Y., Phang, J., Liu, H., Htut, P. M., Zhang, X., Pang, R. Y., Vania, C., Kann, K., and Bowman, S. R. Intermediate-task transfer learning with pretrained models for natural language understanding: When and why does it work? *arXiv preprint arXiv:2005.00628*, 2020.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P. J., et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67, 2020.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

Rebuffi, S.-A., Bilen, H., and Vedaldi, A. Learning multiple visual domains with residual adapters. *Advances in neural information processing systems*, 30, 2017.

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

Sanh, V., Webson, A., Raffel, C., Bach, S. H., Sutawika, L., Alyafeai, Z., Chaffin, A., Stiegler, A., Scao, T. L., Raja, A., et al. Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*, 2021.

Schulman, J., Heess, N., Weber, T., and Abbeel, P. Gradient estimation using stochastic computation graphs. *Advances in Neural Information Processing Systems*, 28, 2015.

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., and Dean, J. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A. Y., and Potts, C. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pp. 1631–1642, 2013.

Tang, D., Zhang, F., Dai, Y., Zhou, C., Wu, S., and Shi, S. One model, multiple tasks: Pathways for natural language understanding. *ArXiv*, abs/2203.03312, 2022.

Tucker, G., Mnih, A., Maddison, C. J., Lawson, J., and Sohl-Dickstein, J. Rebar: Low-variance, unbiased gradient estimates for discrete latent variable models. *Advances in Neural Information Processing Systems*, 30, 2017.

Vu, T., Wang, T., Munkhdalai, T., Sordoni, A., Trischler, A., Mattarella-Micke, A., Maji, S., and Iyyer, M. Exploring and predicting transferability across nlp tasks. *arXiv preprint arXiv:2005.00770*, 2020.

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S. R. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018.

Warstadt, A., Singh, A., and Bowman, S. R. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641, 2019.

Wei, J., Bosma, M., Zhao, V. Y., Guu, K., Yu, A. W., Lester, B., Du, N., Dai, A. M., and Le, Q. V. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*, 2021.

Williams, A., Nangia, N., and Bowman, S. R. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*, 2017.

Wortsman, M., Gururangan, S., Li, S., Farhadi, A., Schmidt, L., Rabbat, M., and Morcos, A. S. lo-fi: distributed fine-tuning without communication. *arXiv preprint arXiv:2210.11948*, 2022a.

Wortsman, M., Ilharco, G., Gadre, S. Y., Roelofs, R., Gontijo-Lopes, R., Morcos, A. S., Namkoong, H., Farhadi, A., Carmon, Y., Kornblith, S., et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International Conference on Machine Learning*, pp. 23965–23998. PMLR, 2022b.

Wortsman, M., Ilharco, G., Kim, J. W., Li, M., Kornblith, S., Roelofs, R., Lopes, R. G., Hajishirzi, H., Farhadi, A., Namkoong, H., et al. Robust fine-tuning of zero-shot models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7959–7971, 2022c.

Xue, L., Constant, N., Roberts, A., Kale, M., Al-Rfou, R., Siddhant, A., Barua, A., and Raffel, C. mt5: A massively multilingual pre-trained text-to-text transformer. *arXiv preprint arXiv:2010.11934*, 2020.

Ye, Q., Zha, J., and Ren, X. Eliciting transferability in multi-task learning with task-level mixture-of-experts. *arXiv preprint arXiv:2205.12701*, 2022.

Yin, M. and Zhou, M. Arm: Augment-reinforce-merge gradient for stochastic binary networks. *arXiv preprint arXiv:1807.11143*, 2018.

Yu, P., Artetxe, M., Ott, M., Shleifer, S., Gong, H., Stoyanov, V., and Li, X. Efficient language modeling with sparse all-mlp. *arXiv preprint arXiv:2203.06850*, 2022.

Zamir, A. R., Sax, A., Shen, W., Guibas, L. J., Malik, J., and Savarese, S. Taskonomy: Disentangling task transfer learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3712–3722, 2018.

Zhang, Z., Lin, Y., Liu, Z., Li, P., Sun, M., and Zhou, J. Moefication: Transformer feed-forward layers are mixtures of experts. In *Findings of the Association for Computational Linguistics: ACL 2022*, pp. 877–890, 2022.

Zhou, Y., Lei, T., Liu, H., Du, N., Huang, Y., Zhao, V., Dai, A., Chen, Z., Le, Q., and Laudon, J. Mixture-of-experts with expert choice routing. *arXiv preprint arXiv:2202.09368*, 2022.

Zoph, B., Bello, I., Kumar, S., Du, N., Huang, Y., Dean, J., Shazeer, N., and Fedus, W. Designing effective sparse expert models. *arXiv preprint arXiv:2202.08906*, 2022.

A. Appendix.

B. Compute resources used

We provide details on the compute resources used in our experiments. All models were trained using a combination of either 48GB A6000s or 24GB A5000s. The training time for each T5-GLUE experiment was approximately 72 hours with Ensemble method taking 125 hours, while each ResNet-DomainNet experiment required approximately 11 hours of training.

C. Experiment Details

We provide details on the experimental setup and hyperparameter choices for the T5-GLUE and ResNet-DomainNet experiments described in the main text.

C.1. T5-GLUE

In the T5-GLUE experiments, all T5 models were trained for $400k$ steps using a learning rate of $3e^{-4}$, with $2k$ warmup steps, and batch size of 128. The AdamW optimizer was used with its default settings. We ran ST-Gumbel estimator with a τ value of 10 and an anneal rate of $1e^{-6}$. For the REINFORCE estimator in Equation 1, we used the same values as in (Clark et al., 2022), $\alpha = 1e^{-2}$, $\beta = 5e^{-4}$, and $\gamma = 1e^{-2}$. The adapters here use $swish$ non-linearity in between. We concatenated all 8 datasets of GLUE and perform multitask training.

C.2. ResNet-DomainNet

In the ResNet-DomainNet experiments, all ResNet models were trained for $100k$ steps with batch size of 128 and a learning rate of $1e^{-3}$, with no warm up, using Adam optimizer. We used τ value of 10 and anneal rate of $1e^{-4}$ for the ST-Gumbel estimator. The values of α, β, and γ for the REINFORCE estimators in Equation 1 are same as in T5-GLUE experiments. The adapters also used $swish$ non-linearity in between. All the domains from DomainNet were concatenated to perform multitask training similar to T5-GLUE.

D. Expert dropout

Table 2 illustrates the impact of expert dropout on different learned routing methods. It is evident that SMEAR benefits from an improvement of 3.8% on T5-GLUE, and Top-K benefits from an improvement of 0.3% and 0.2% on T5-GLUE and ResNet-DomainNet respectively. Expert dropout is included for these two methods when discussed in the main text. However, expert dropout negatively impacts the performance of ST-Gumbel and REINFORCE methods and thus, it is excluded for these two methods in the main text.

Routing	T5-GLUE	ResNet-DomainNet
Top-k	77.6	59.8
w/ Expert dropout 0.1	77.9 (+0.3)	60.0 (+0.2)
ST-Gumbel	76.1	58.3
w/ Expert dropout 0.1	75.5 (-0.6)	57.9 (-0.4)
REINFORCE	78.4	59.8
w/ Expert dropout 0.1	77.2 (-1.2)	59.8 (+0.0)
SMEAR	78.7	62.0
w/ Expert dropout 0.1	82.5 (+3.8)	62.0 (+0.0)

Table 2. Performance comparision of different learned routing strategies w and w/o dropout. The results indicate that SMEAR and Top-k method benefit from the expert dropout, while ST-Gumbel and REINFORCE are negatively affected.

E. Full results on T5-GLUE and ResNet-DomainNet

We show the full results of T5-GLUE in Table 3 and ResNet-DomainNet in Table 4.

Routing	RTE acc	SST-2 acc	MRPC f1	MRPC acc	STS-B pearson	STS-B spearman	QQP f1	QQP acc	MNLI acc	QNLI acc	CoLA mcc	Average
Tag	52.9	92.1	88.1	82.8	84.1	83.9	86.1	89.6	84.9	89.3	29.9	78.5
Tag+	69.0	92.0	87.6	82.1	83.3	83.1	86.7	89.8	84.8	88.4	30.9	79.8
Hash	57.0	87.3	77.3	69.0	67.8	67.2	77.0	83.4	73.6	80.3	1.4	67.4
Monolithic	59.4	91.6	90.0	85.7	86.6	87.0	85.4	88.9	84.2	89.7	3.9	77.5
Top-k	64.6	92.4	87.2	81.5	89.1	88.9	85.1	88.8	84.0	89.1	6.3	77.9
ST-Gumbel	63.2	92.3	85.2	79.3	87.0	86.6	84.7	88.4	83.7	88.4	-1.8	76.1
REINFORCE	63.2	92.8	88.0	83.0	88.2	88.1	86.0	89.3	85.1	90.2	9.0	78.4
SMEAR	68.1	91.0	92.9	90.1	89.6	89.4	86.1	89.6	85.7	91.0	34.4	82.5
Ensemble	73.0	92.4	91.1	87.7	86.6	85.9	87.0	90.1	85.3	90.4	40.8	82.8

Table 3. Full T5-GLUE results.

Routing	Clipart	Infograph	Painting	Quickdraw	Real	Sketch	Average
Tag	63.3	30.7	58.2	61.8	74.2	54.8	61.5
Hash	53.5	23.2	50.3	48.5	68.6	46.1	52.5
Monolithic	60.6	28.0	54.7	58.9	72.2	52.5	59.0
Top-k	61.9	29.4	55.3	60.1	73.2	53.4	60.0
ST-Gumbel	59.8	27.2	54.6	57.7	72.0	51.9	58.3
REINFORCE	61.2	28.7	55.6	60.0	72.8	53.9	59.8
SMEAR	64.2	31.5	57.7	62.4	74.3	56.0	62.0
Ensemble	65.0	31.3	58.3	63.2	74.3	57.0	62.5

Table 4. Full ResNet-DomainNet results.

F. Routing distribution in all routing blocks

Here we put the routing distribution in all routing blocks in both T5-GLUE and ResNet-DomainNet learnt by SMEAR in Figure 4, Figure 5, Figure 6, and Figure 7.

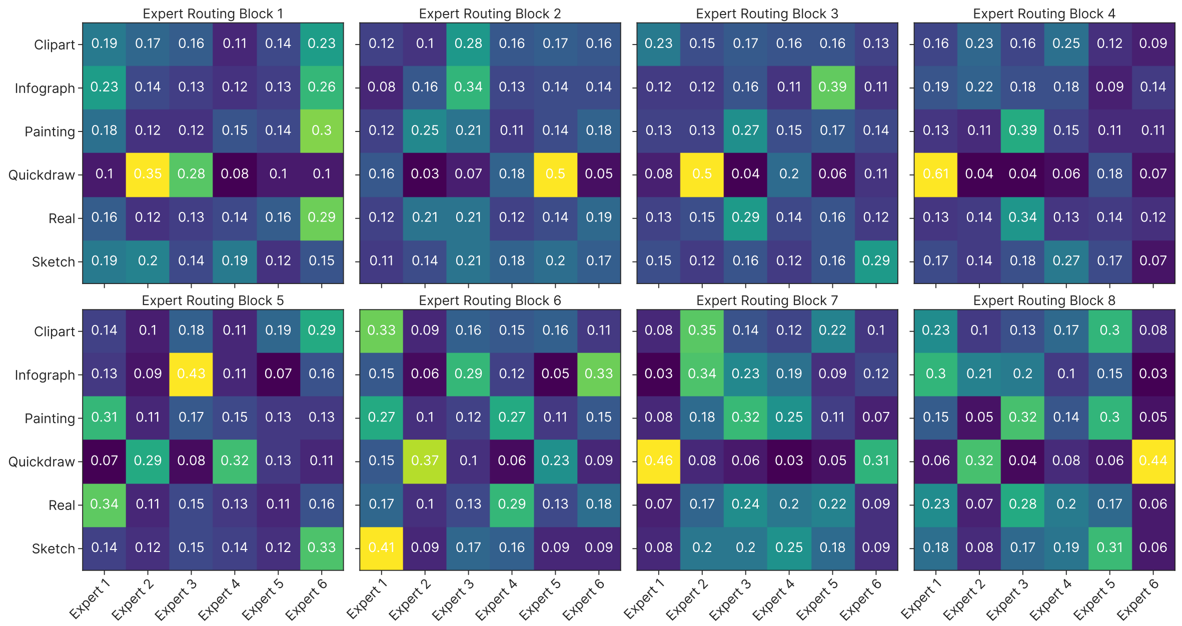


Figure 4. Routing distribution learnt by SMEAR in all routing blocks of ResNet-DomainNet

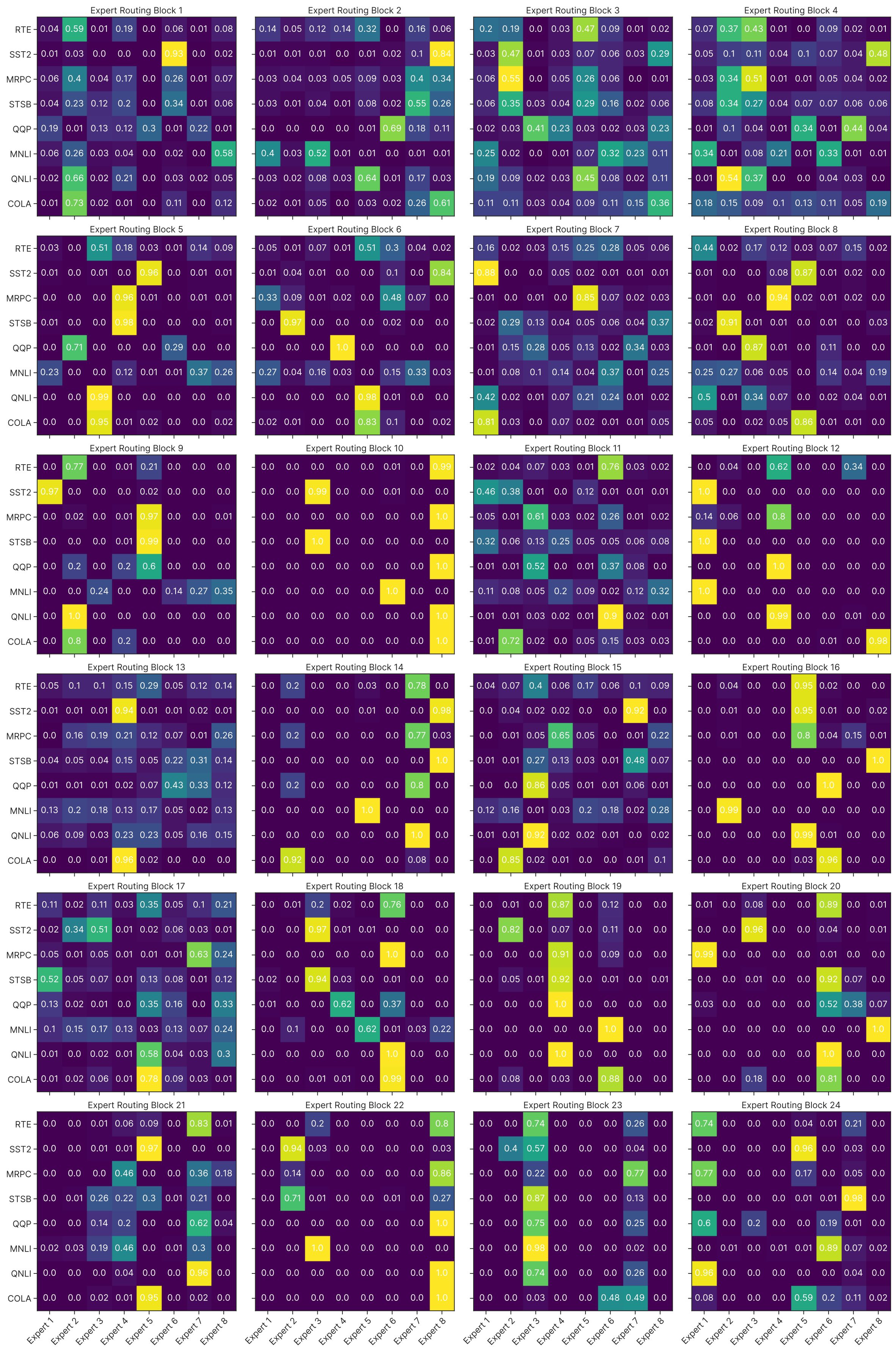


Figure 5. Routing distribution learnt by SMEAR in the encoder routing blocks (1-24) of T5-GLUE

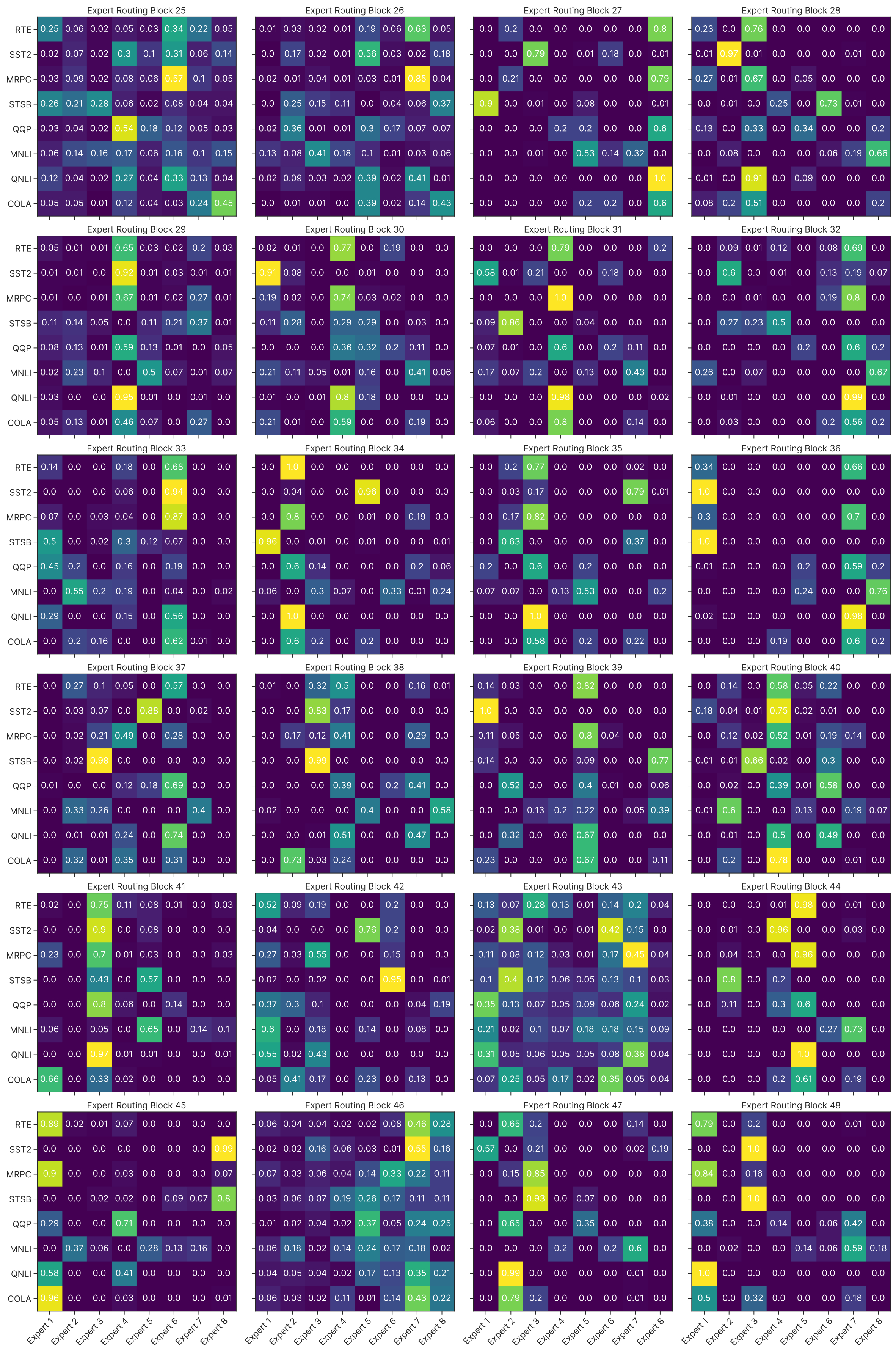


Figure 6. Routing distribution learnt by SMEAR in the decoder routing blocks (25-48) of T5-GLUE

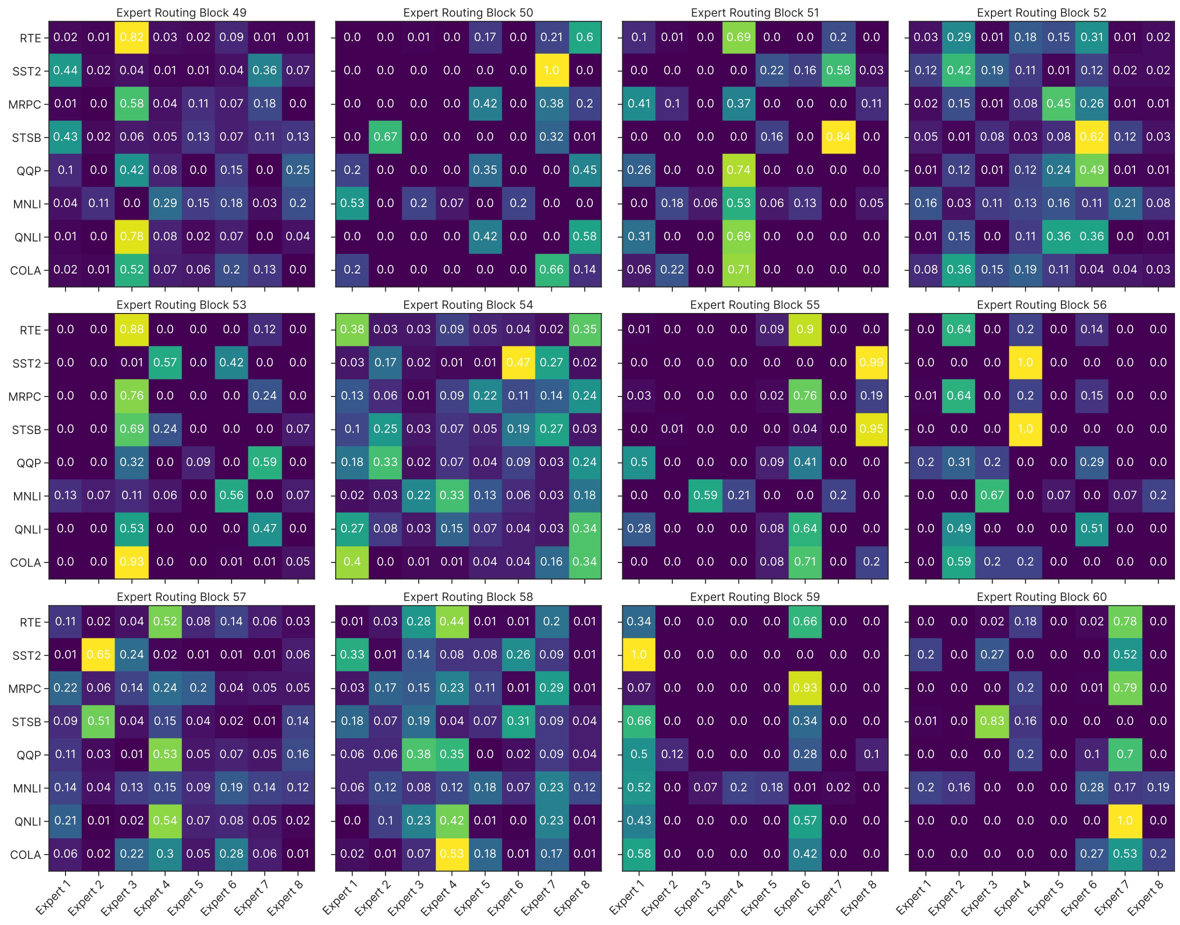


Figure 7. Routing distribution learnt by SMEAR in the decoder routing blocks (49-60) of T5-GLUE