Sparsely Activated Mixture-of-Experts are Robust Multi-Task Learners

Anonymous ACL submission

Abstract

Traditional multi-task learning (MTL) methods use dense networks that use the same set of shared weights across several different tasks. This often creates interference where two or more tasks compete to pull model parameters in different directions. In this work, we study whether sparsely activated Mixture-of-Experts (MoE) improve multi-task learning by specializing some weights for learning shared representations and using the others for learning task-specific information. To this end, we devise task-aware gating functions to route examples from different tasks to specialized experts which share subsets of network weights conditioned on the task. This results in a sparsely activated multi-task model with a large number of parameters, but with the same computational cost as that of a dense model. We demonstrate such sparse networks to improve multi-task learning along three key dimensions: (i) transfer to low-resource tasks from related tasks in the training mixture; (ii) sample-efficient generalization to tasks not seen during training by making use of task-aware routing from seen related tasks; (iii) robustness to the addition of unrelated tasks by avoiding catastrophic forgetting of existing tasks.

1 Introduction

The traditional mechanism of using large-scale pre-trained language models PLMs (Devlin et al., 2019; He et al., 2021) involve fine-tuning them for each task individually. This approach fails to benefit from interactions between tasks that could be related to each other. For instance, the task of predicting if one text entails or contradicts another can benefit from tasks that predict whether two texts are semantically similar or not. To address these limitations, Multi-Task Learning (MTL) methods like MT-DNN (Liu et al., 2019) and Muppet (Aghajanyan et al., 2021a) instead train a single model jointly on a multi-task mixture consisting of multiple tasks. The typical mechanism is to facilitate transfer between the tasks by encoding the examples using a task-agnostic network shared between all the tasks, and then using task-specific layers on top to optimize individual task objectives. The dominant choice for the network is a Transformer-based PLM such as BERT (Devlin et al., 2019). However, such dense (fully-connected) task-agnostic networks have the limitation that they use all the weights of the network for every example, including those coming from very different tasks. This creates interference among different tasks, e.g., the tug-of-war phenomenon (Hadsell et al., 2020) where two or more tasks pull the model parameters in different directions, thus impacting the multi-task learning performance.

A possible mechanism to alleviate this problem is to devise a task-aware network that can capture specialized information about individual tasks, as well as information that can be shared among multiple tasks. Mixture-of-Experts (MoE) framework (Shazeer et al., 2017; Fedus et al., 2021; Lepikhin et al., 2021) provides a way to model this mechanism. Such architectures are designed to support conditional computation in which only certain weights of the network are activated per input as governed by a gating mechanism. This sparse design has an additional advantage of providing additional capacity in terms of model parameters while keeping overall computational cost constant.

The above sparse MoE models have been typically trained from scratch using language modeling objectives for tasks like neural machine translation; or fine-tuned on NLU tasks in a single-task setting. In contrast, in this work *we study multi-task adaptation (as opposed to pre-training from scratch) of sparse MoE models on diverse NLU tasks when judiciously initialized with the weights of a pre-trained language model*. Our motivation for using MoEs is that the sparsity and conditional computation within MoEs will help to alleviate inter-task interference by specializing some weights for learn-

1

ing shared representations and using the others for learning task-specific information.

Multi-task adaptation for sparse MoE models that have been traditionally used in single-task settings require rethinking the gating mechanism. Existing sparse models use a single task-agnostic shared gate that learns to route inputs from all the tasks, leading to interference wherein different tasks compete for the shared gate.

Contributions: We (*Contribution 1*) first address this limitation by devising a task-aware gating mechanism within sparse MoEs to route the input (tokens from different tasks) to specialized experts conditioned on the task to support MTL.

Thereafter, (*Contribution 2.1*) we perform an extensive empirical study of the robustness of dense and sparse models to inter-task interference for multi-task learning on three key dimensions, (i) *transfer to low-resource tasks* from related tasks in the training mixture; (ii) *sample-efficient generalization to tasks not seen during training* from related seen tasks; (iii) *robustness to the addition of unrelated tasks* by avoiding catastrophic forgetting of existing tasks. We (*Contribution 2.2*) empirically demonstrate sparse MoE models with task-aware gating and routing to be more robust multi-task learners than their non-MoE dense counterparts on the above dimensions.

2 Sparse Mixture-of-Experts: Background

We adopt the popularly used Transformer architecture (Vaswani et al., 2017) as the basic encoder consisting of L repeated Transformer blocks, where each block consists of a self-attention sub-layer, a fully connected feed-forward network (FFN) and residual connections around the sub-layers followed by layer normalization.

The objective of sparse design of the above Transformer blocks is to support conditional computation and increase the parameter count while keeping the floating point operations (FLOPs) for each input example constant. Mixture-of-Experts (MoE) Transformer models (Shazeer et al., 2017; Fedus et al., 2021; Lepikhin et al., 2021; Zuo et al., 2021) achieve this by using N feed-forward networks (FFN), namely "experts" denoted as $\mathbb{E}_{i=1}^{N}$, each with its own set of learnable weights. In order to sparsify the network to keep the FLOPs constant, there is an additional gating network $\mathbb{G}$ whose output is a sparse N-dimensional vector to route each

token via a few of these experts. Note that, a sparse model with $N = 1$ corresponding to only one FFN layer in each Transformer block collapses to the traditional dense model.

Consider x_s as the input token representation in the sth position to the MOE layer comprising of the $\{\mathbb{E}\}_{i=1}^{N}$ expert FFNs. Also, consider w_i^{in} and w_i^{out} to be the input and output projection matrices for i^{th} expert. Expert output $\mathbb{E}_i(x_s)$ is given by:

$$\mathbb{E}_i(x_s) = w_i^{out} \cdot GeLU(w_i^{in} \cdot x_s) \qquad (1)$$

Consider $\mathbb{G}(x_s)$ to be output of the gating network. Output of the sparse MoE layer is given by:

$$h(x_s) = \sum_i \mathbb{G}(x_s)_i \, \mathbb{E}_i(x_s) \qquad (2)$$

where $\mathbb{G}(x_s)_i$ denotes the probability of selecting expert $\mathbb{E}_i$ for x_s.

3 Sparse Multi-task Learning with Mixture-of-Experts

We first highlight the shortcoming of existing sparse MoE models for multi-task learning and our architectural modifications to support the same along with an analysis of its impact on the model size and task scalability. We then present some details on the task formulation and optimization objectives to train sparse multi-task models.

3.1 Task-aware Sparse Routing to Experts

The sparse MoE design outlined in the previous section does not consider the underlying task (Figure 1(a)). Given the same input from different tasks, the task-agnostic gating mechanism routes tokens to the same experts, thereby generating similar hidden-state representations. This is an issue during multi-task learning, where it is beneficial to learn task-specific contextualized representation of the input. To address this shortcoming, we modify the gating function to be task-aware, such that inputs from a given task are routed to specialized experts that also share weights across related tasks.

Consider a set of T diverse tasks in the multi-task mixture and $x_{s,t}$ to be the token representation in the sth position of the input sequence from task $t \in T$, where each task is equipped with its own loss function. Consider trainable weight matrices $\mathcal{W}_{g,t} \in \mathcal{R}^{N \times d}$ corresponding to each task $t \in T$ where, N is the number of experts and d is the hidden state dimension. To incorporate task information in the gating mechanism, we multiply the
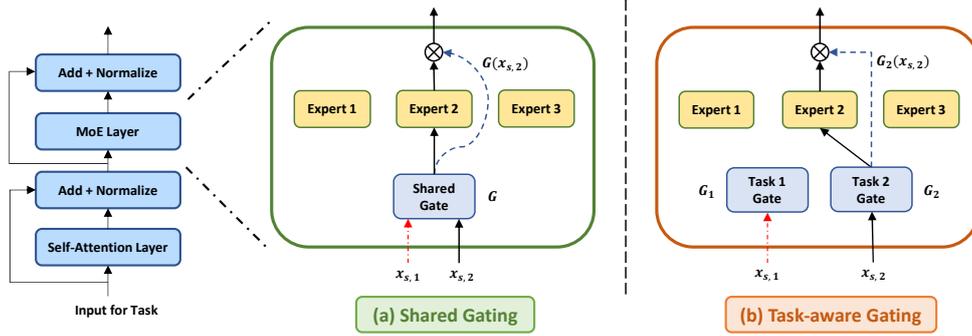
Figure 1: Sparse MoE layer with 3 Experts, 2 Tasks, and top-1 expert routing with (a) Shared Gating, and (b) Task-aware Gating. $x_{s,1}$ and $x_{s,2}$ are tokens from Task 1 and 2 respectively. They share the same gate G in sub-figure (a), and routed to respective task-specific gates in sub-figure (b). For simplicity, we only show the pathway for $x_{s,2}$ with a solid line, and show the gating behavior for $x_{s,1}$ with a dashed red line

input $x_{s,t}$ with the task-specific weight matrix $\mathcal{W}_{g,t}$ to obtain the routing logits:

$$l_t(x_{s,t}) = x_{s,t} \cdot \mathbb{W}_{g,t} \qquad (3)$$

We can further normalize them via a softmax distribution over the N experts in each MoE layer to obtain the corresponding routing probabilities. The gate-value for the ith expert is given by:

$$\mathbb{G}_t(x_{s,t})_i = \frac{e^{l_t(x_{s,t})_i}}{\sum_{j=1}^{N} e^{l_t(x_{s,t})_j}} \qquad (4)$$

We can now select the top-k gate values for routing the token. In order to keep the number of FLOPs in the sparse Transformer to be the same as that of a dense one, the gating mechanism is constrained to route each token to only the top-1 expert FFN selected as:

$$g_t^*(x_{s,t}) = max_i \, \mathbb{G}_t(x_{s,t})_i \qquad (5)$$

The output of the sparse MOE layer in Equation 2 can be modified with the task-specific gating function by linearly combining the selected top-1 expert's ($\mathbb{E}^*$) computation on $x_{s,t}$ and the probability of selecting the expert as:

$$h(x_{s,t}) = g_t^*(x_{s,t}) \, \mathbb{E}^*(x_{s,t}) \qquad (6)$$

where h denotes the task-specific representation of input $x_{s,t}$.

In the above formulation, the task-specific gating function $\mathbb{G}_t$ learns to route tokens from the input to specialized experts. Note that the experts themselves do not have explicit relationship with the task and are only dependent on input context so as to encourage information sharing among all experts. The expert selection is implicitly conditioned on the task id t (provided with the input)

via task-aware gating function $\mathbb{G}_t$. We refer our framework as **MT-TaG**, short for Multi-Task Task-aware Gating (Figure 1(b)).

3.2 Analysis of Sparsity and Task-scalability

We introduce the feed-forward networks (FFN) as experts in every layer of the Transformer. Consider N experts, L layers and P_f to be the number of parameters in each FFN expert. The number of expert parameters in the model is $L \times N \times P_f$. Since the experts are shared among all tasks, increasing the number of tasks does not impact expert parameters.

On the other hand, the gating network is task-aware which increases the number of parameters with more tasks. Considering H to be the hidden state dimension and T to be the number of tasks, the number of gating parameters is $L \times N \times H \times T$.

Since the hidden state dimension and number of tasks are much less than the number of FFN parameters (i.e., $H \times T \ll P_f$) in most practical settings, increasing tasks contribute very less parameters as compared to the parameters already contained in the standard feed-forward Transformer networks.

Consider the following as an illustration. Consider a 6-layer Transformer with 384 hidden dimension and $22M$ **encoder parameters** corresponding to a standard dense Transformer. Consider 4 experts and 8 tasks for MTL, where we introduce these experts in each Transformer layer. MT-TaG contains only $74K$ **gating parameters** in the task-specific gating networks for expert selection as compared to $21M$ **expert parameters**. In total, the sparse MT-TaG model doubles the number of parameters as compared to the dense model although incurring the same number of FLOPs with top-1 expert selection. This capacity coupled with task-awareness improves model performance in MTL as demonstrated in experiments.

3

3.3 Multi-task Training

We now outline multi-task objectives and protocol for training the MT-TaG model.

Task objectives: For a classification task t, we use a task-specific projection layer on top of the MTL encoder to obtain the class probability distribution for the contextualized representation of an input example x_t[1] from task t as:

$$P(c|x_t) = Softmax(\mathbb{U}_t \cdot h(x_t)) \quad (7)$$

where, $\mathbb{U}_t \in \mathbb{R}^{C_t \times d}$ is the task-specific parameter matrix with C_t representing the number of classes and d as the hidden state dimension.

For a regression task t (e.g., textual similarity), we obtain the output score for the contextualized representation of the input x_t as:

$$S(x_t) = \mathbb{V}_t \cdot h(x_t) \quad (8)$$

where, $\mathbb{V}_t \in \mathbb{R}^{1 \times d}$ is the task-specific parameter matrix and $S(x_t) \in \mathbb{R}(-\infty, \infty)$.

For classification tasks, we use cross-entropy loss, where we train the network to minimize the following objective in the MTL setup:

$$-\sum_{t \in \mathbb{T}} \sum_{x_t \in X_t} \sum_{c \in C_t} \mathbb{1}(x_t, c) \ log \ P(c|x_t) \quad (9)$$

where, X_t is the set of examples from task t, $\mathbb{1}(x, c)$ is the binary indicator which is 1 if c is the correct class label for x and 0 otherwise.

For regression tasks, we use mean-squared error loss, where we train the network to minimize the following objective in the MTL setup:

$$\sum_{t \in \mathbb{T}} \sum_{\langle x_t, y_t \rangle \in \langle X_t, Y_t \rangle} (y_t - S(x_t))^2 \quad (10)$$

where, $\langle X_t, Y_t \rangle$ is the set of examples from task t with corresponding ground-truth scores.

Joint optimization: We jointly optimize Equations 9 and 10 to train the entire model including the gating network by back-propagation, where the gradients back-propagate through the gating network to the inputs.

Loss scaling: In the MTL setup, the number of classes per task can vary. To ensure stability in the training, we leverage loss scaling to normalize the task-specific loss function in Equation 9 with respect to the number of classes in the task t as

$(\sum_{c \in C_t} \mathbb{1}(x_t, c) \ log \ P(c|x_t))/log(|C_t|)$, where $|.|$ denotes the cardinality of the set of classes.

Batching and sampling: The MTL training process optimizes several objectives which are often at loggerheads with each other. Recent work (Aghajanyan et al., 2021b) demonstrates *heterogeneous batching* to work better for MTL, where batches from different tasks are sampled to construct a super-batch, which is then used for jointly optimizing corresponding task-objectives. We follow similar principles along with employing a natural sampling of tasks, wherein we sample batches from tasks in proportion to their dataset sizes to reflect the complexity of the corresponding tasks.

4 Experimental Setup

4.1 Datasets

We use 8 diverse NLU datasets from the GLUE benchmark (Wang et al., 2018) for MTL training consisting of single-text classification tasks such as COLA and SST-2; paired-text classification tasks such as RTE, MRPC, QNLI, QQP, and MNLI; and paired-text regression tasks such as STS-B. These evaluate various NLU capabilities such as sentiment classification in SST-2; textual entailment in RTE, QNLI, and MNLI; paraphrase detection in MRPC and QQP; text similarity in STS-B; and text acceptability in CoLA. There are varying number of examples per dataset ranging from $2.5K$ examples in the smallest one (RTE) to $393K$ examples in the largest one (MNLI). This allows us to study the efficacy of MTL models in terms of transfer to low-resource tasks. The task mixture also consists of tasks like COLA and SST-2 that have low similarity with the rest, enabling us to study the robustness of MTL models in the presence of unrelated tasks. We provide more details about these datasets and their sizes in Appendix A.2 and Table 9.

4.2 Models for Comparison

We consider several models that are all FLOPs matched per token for comparison as follows.

(a) Single-Task: This baseline trains a dense model directly on individual end-tasks without MTL. Since there is no interaction across tasks, this baseline helps us evaluate the impact of MTL.

(b) MT-Dense: This baseline is created by training a dense MTL model. Note that this baseline is similar in flavor to the multi-task learning methods like MT-DNN (Liu et al., 2019) and Muppet (Aghajanyan et al., 2021b).

---

[1]For inputs with sequence pairs (x^1, x^2), we consider $x = x^1 \oplus x^2$, with $\oplus$ representing concatenation operation.

(c) MT-Switch: This is a sparse MTL Mixture-of-Experts model using a single shared gate for all tasks as depicted in Figure 1(a). Note that MT-Switch differs with MT-TaG only in its usage of a single task-agnostic shared gate, helping us evaluate the impact of task-aware gating.

(d) MT-TaG: This is the sparse MTL Mixture-of-Experts model outlined in Section 3.1 (depicted in Figure 1(b)) that uses task-aware gating.

All the models have similar FLOPs per token and all the MTL models are trained using the procedure outlined in Section 3.3. We use top-1 expert routing for both sparse MTL models.

4.3 Model Initialization and Setup

Dense models: As in prior multi-task learning works (Liu et al., 2019), we initialize the dense model using weights from pre-trained language models. In addition to using BERT_{Base} (12 layers, 768 hidden size, $110M$ params) and BERT_{Large} (24 layers, 1024 hidden size, $345M$ params) pre-trained models, we also consider MiniLM (Wang et al., 2021) (6 layers, 384 hidden size, $22M$ params) distilled from BERT_{Large} as its compressed variant. Unless otherwise stated, we use MiniLM as our default encoder to carry out an extensive study with limited compute resources.

Sparse models: For a fair comparison with the dense models, we create FLOPs matched sparse models, and initialize them using the weights of dense pre-trained language models. To this end, we replace the feed-forward layers (FFNs) in each transformer layer of the dense model with a MoE layer containing N experts and T gates ($T = 1$ for MT-Switch; $T =$ num. of tasks for MT-TaG). This results in as many MoE layers as the number of Transformer layers of the corresponding dense pre-trained language model used for initialization. To initialize the FFN weights of experts in any MoE layer, we simply make N copies of the FFN weights of the corresponding layer from the dense pre-trained language model[2].

4.4 Implementation Details

We use standard wordpiece tokenization ($30K$ vocabulary) and segmentation for the input sequences. We use $N = 4$ experts in all layers for our experiments[3], giving us sparse models with $44M, 280M$, and $940M$ parameters that are FLOPs matched to MiniLM, BERT_{Base}, and BERT_{Large} encoders, respectively. We initialize all gating weights using a normal distribution with 0 mean and 0.001 standard deviation. Similarly, we initialize task-specific parameter matrices $\mathbb{U}_t, \mathbb{V}_t$ using a normal distribution with 0 mean and 0.02 standard deviation. We initialize all layer normalization weights with 1, bias weights with 0, and use a dropout of 0.1.

We use Adam Optimizer (Kingma and Ba, 2015) with a linear learning rate decay schedule and warm-up. We use mixed-precision training, clip the norms of gradients to 1, and use 4 Nvidia V100 GPUs for distributed training. We utilize PyTorch and HuggingFace Transformers (Wolf et al., 2019) for our implementation[4].

4.5 Evaluation

MTL Training protocol: We follow a two-stage training protocol for MTL models. We first train the dense or sparse model (initialized from a pre-trained language model as outlined in Section 4.3) on a multitask mixture such as the GLUE dataset following the MTL training procedure (as outlined in Section 3.3) for a fixed number of steps, which gives us the corresponding MTL model. We then further fine-tune the MTL model on individual target datasets. This additional fine-tuning step has been shown to be beneficial for the model performance (Liu et al., 2019). Note that we use the same training protocol for all the MTL models.

Metrics: We use the standard train and dev splits for all GLUE datasets for training and evaluation. For the MTL models, we report the numbers obtained from the fine-tuning stage. We use Spearman correlation as our evaluation metric for STS-B, Matthews correlation coefficient (MCC) for COLA, and accuracy for the rest. For MNLI, we report the average accuracy on the matched (in-domain) and mismatched (cross-domain) splits. We additionally report two aggregate statistics: *All Tasks*, and *Small Tasks*, capturing the average performance on all tasks and just the small tasks respectively. We define Small Tasks as the tasks with $\leq 10k$ examples, which for GLUE includes RTE, MRPC, STS-B, and COLA. We provide more experimental details, including hyper-parameter tuning and values in Appendix A.3.2.

---

[2]Experiments with initializing expert weights differently by adding a small random noise did not show improvements.

[3]We provide results with varying #experts in Appendix.

[4]Our code and model checkpoints will be made public.

Model	RTE (2.5k)	MRPC (3.7k)	STS-B (5.7k)	CoLA (8.5k)	SST-2 (67.3k)	QNLI (105k)	QQP (364k)	MNLI (393k)	Small Tasks (Avg.)	All Tasks (Avg.)
Single-Task	70.7	88.7	88.9	<u>41.8</u>	**92.4**	**90.4**	90.6	**83.9**	72.5	80.9
MT-Dense	77.9	89.0	<u>90.5</u>	**42.1**	92.0	<u>90.3</u>	<u>90.8</u>	<u>83.8</u>	74.9	82.1
MT-Switch	<u>78.9</u>	90.0	<u>90.5</u>	40.7	92.0	<u>90.3</u>	**90.9**	83.6	<u>75.0</u>	<u>82.1</u>
MT-TaG	**81.1**	**90.7**	**90.6**	41.1	<u>92.1</u>	90.2	<u>90.8</u>	83.6	**75.9**	**82.5**

Table 1: Comparison of dense and sparse models on GLUE. Best task numbers are **boldfaced**, and second-best <u>underlined</u>. Sparse MoE with task-specific gating (MT-TaG) outperforms Single-Task and FLOPs matched dense and sparse MTL models with significant improvements for low-resource tasks. All models use MiniLM encoder.

5 Robustness Analysis

We perform an extensive empirical study of the robustness of sparse and dense MTL models along key dimensions with the following desiderata:

❶ **Transfer to low-resource tasks**: A robust model should be able to alleviate task interference in the training mixture and improve performance on the low-resource tasks through transfer from other related tasks.

❷ **Sample-efficient generalization to unseen related tasks**: A robust model should be able to retain information from individual tasks in its training mix, and generalize in a sample-efficient manner to new related tasks that are not seen during training.

❸ **Robustness to the addition of unrelated tasks**: A robust model should be better at weathering the interference introduced by the addition of unrelated tasks in its training mixture, and avoid catastrophic forgetting of existing tasks.

5.1 Low-resource Task Transfer

We first evaluate the ability of MTL models to leverage task-level similarities in the multitask mixture to improve performance on low-resource tasks. To this end, we train and evaluate all models on GLUE. Table 1 shows that all MTL models obtain improvements on low-resource tasks over Single-Task baseline, while maintaining similar performance on relatively high-resource tasks. This demonstrates the benefit of multi-task learning in utilizing inherent similarities between tasks. Furthermore, we observe that both the sparse MoE models (MT-Switch and MT-TaG) outperform the non-MoE dense one (MT-Dense), demonstrating the benefit of inducing sparsity for MTL. Finally, we observe the sparse MoE model with task-aware gating (MT-TaG) to outperform all baselines, including single-gate sparse MoE (MT-Switch), demonstrating improved ability to mitigate interference between tasks during multi-task learning.

Model	SciTail		IMDB	
	1% (235)	10% (2.4k)	1% (250)	10% (2.5k)
Single-Task	81.9	90.6	86.1	90.6
MT-Dense	86.8	**93.3**	<u>89.8</u>	**91.2**
MT-Switch	<u>89.3</u>	92.9	<u>89.8</u>	91.1
MT-TaG	**90.0**	<u>92.9</u>	**90.3**	**91.2**

Table 2: Generalization performance on low-resource unseen related tasks. MT-TaG delivers large gains over Single-Task, and outperforms other MTL models in extremely low-resource settings demonstrating superior sample-efficiency. All models use MiniLM encoder.

5.2 Sample-efficient Generalization to Unseen Related Tasks

Section 5.1 demonstrates the benefit of sparse models on improving the MTL model performance on low-resource tasks. In this experiment, we want to evaluate their ability to generalize to related tasks that were not encountered during MTL training in a sample-efficient manner.

To study this generalization ability, we leverage SciTail and IMDB as the unseen tasks for the GLUE-trained MTL models. Note that these tasks have some similarity to a subset of the GLUE tasks. For instance, SciTail is an NLI dataset with similarities to RTE, QNLI, and MNLI in GLUE; whereas IMDB is a sentiment classification dataset with similarities only to SST-2. This variation in similarity helps us study the degree of transferability from the multi-task training mixture to the new unseen tasks. We simulate low-resource settings by creating 1% and 10% samples from these datasets to study sample-efficiency, yielding datasets with roughly 250 and 2.5k examples respectively. We use accuracy as the metric for both datasets. We provide more details about these datasets and their task formulation in Appendix A.2 and Table 9.

We only fine-tune the GLUE-trained MTL models on these datasets, and compare against corre-

sponding Single-Task baselines. For fine-tuning MT-TaG, we exploit task-specific gates, and re-use the gate corresponding to SST-2 for IMDB, and the gate corresponding to MNLI for SciTail due to their task-level similarities.

Table 2 shows that all MTL models obtain improvements over the Single-Task baselines, demonstrating generalization ability of the MTL models. Furthermore, we observe that MT-TaG outperforms all baselines on extremely low-resource settings on unseen datasets **demonstrating superior sample-efficiency** of sparse models. MT-TaG shows improvements even on IMDB which has only one related dataset in GLUE demonstrating improved task transfer from related tasks. *We attribute these capabilities to the re-use of MT-TaG's task-specific gates and routing that help it to better transfer information from related tasks in a sample-efficient manner.* We further found re-using unrelated task gates and randomly initializing the gates to perform significantly worse (results in Appendix A.1.1).

5.3 Robustness to Unrelated Tasks

Section 5.2 demonstrates the improved performance of sparse MTL models to transfer information from even a single task of its kind (referred to as *singleton tasks* henceforth) in the multi-task mixture. In this section, we further evaluate the robustness of MTL models on adding several diverse singleton tasks. Specifically, we evaluate if the singleton task addition has an adversarial affect on the performance of existing tasks in the multi-task mixture due to catastrophic forgetting.

To study this, we remove CoLA and SST-2 singleton datasets from the GLUE multi-task mixture, and refer to this new clean multi-task mixture as C-GLUE (short for Clean-GLUE). We evaluate the robustness by training all MTL models on both GLUE and C-GLUE, and comparing their performance on the common tasks: RTE, MRPC, STS-B, QNLI, QQP, and MNLI. We report the average performance on the common *Small Tasks* and *All Tasks* in Table 3, and provide the corresponding task-level results in Table 10 of Appendix A.5.1.

We observe performance of dense MTL model (MT-Dense) to decrease from C-GLUE to GLUE, demonstrating its lack of robustness to unrelated datasets in the multi-task mixture. Both sparse MTL models show better robustness because of their capability to specialize experts for unrelated tasks. MT-TaG performs the best, further demon-

Dataset	Small Tasks	All Tasks
	MT-Dense	
C-GLUE	86.27	87.18
GLUE	85.80 (-0.47)	87.05
	MT-Switch	
C-GLUE	86.27	87.22
GLUE	86.47 (+0.20)	87.37
	MT-TaG	
C-GLUE	86.50	87.32
GLUE	87.47 (+0.97)	87.83

Table 3: Model performance on GLUE (containing several diverse tasks) and C-GLUE (as a subset of GLUE containing only related tasks) evaluated on the common tasks in both. Sparse MTL models demonstrate robustness in the presence of unrelated tasks in GLUE, with MT-TaG with task-specific routing being the most robust. All models use MiniLM encoder.

strating the usefulness of combining expert specialization in sparse MoE with task-specific routing.

This result, combined with the findings in Section 5.2 demonstrate that *MT-TaG is not only better at transfer from singleton tasks, but is also more robust to their presence in the multi-task mixture*. This motivates scaling MT-TaG to a large number of diverse tasks as demonstrated in Section 6.2.

6 Scaling Analysis

6.1 Encoder Size Scaling

We study the sensitivity of the MT-TaG model performance with change in the encoder size. To this end, we train MT-TaG using MiniLM, $BERT_{Base}$ and $BERT_{Large}$ encoders of varying number of parameters. From Table 4, we observe that MT-TaG significantly outperforms single-task baselines across different encoder sizes.

We also compare against the multi-task MT-DNN model from Liu et al., 2019, which is similar in flavor to our MT-Dense model. Our sparse MTL MoE model MT-TaG shows impressive gains over the dense MT-DNN model[5], especially on low-resource tasks. We provide task-level results for comparison in Table 11 of Appendix A.5.2.

6.2 Number of Tasks Scaling

In this experiment, we evaluate if MT-TaG can continue to leverage similarities between tasks in the presence of a large number of tasks in its multi-task mixture. To this end, we expand our GLUE

---

[5]MT-DNN only provides numbers for $BERT_{Large}$.

Model	Small Tasks	All Tasks
	MiniLM	
Single-Task	72.53	80.93
MT-TaG	**75.88**	**82.53**
	BERT$_{Base}$	
Single-Task	76.53	83.34
MT-TaG	**80.73**	**85.45**
	BERT$_{Large}$	
Single-Task	78.85	84.93
MT-TaG	**82.73**	**86.94**
MT-DNN	81.25	86.04

Table 4: Performance of models with different encoder sizes. MT-TaG shows consistent gains across encoders of different sizes. MT-TaG also outperforms the dense MTL baseline MT-DNN (Liu et al., 2019).

Model	Small Tasks	All Tasks
Single-Task	81.28	85.00
MT-TaG	**83.56**	**86.46**

Table 5: Performance comparison on GLUE++ using BERT$_{Large}$. MT-TaG demonstrates impressive gains on scaling to a large number of diverse tasks.

multi-task mixture to 16 tasks with the addition of NLI datasets such as CB; QA datasets such as COPA, MultiRC, and BoolQ; Sentiment datasets such as IMDB, Rotten Tomatoes, and Yelp Polarity; and Word-sense disambiguation datasets such as WiC. For simplicity, we refer to this multitask mixture as GLUE++. We provide more details about these datasets in Appendix A.2 and Table 9. We train and evaluate MT-TaG on this dataset using BERT$_{Large}$ encoder, and compare with corresponding Single-Task baselines on aggregate average performance metrics, *Small Tasks* and *All Tasks*. For GLUE++, *Small Tasks* includes RTE, MRPC, STS-B, COLA, Rotten Tomatoes, WiC, CB, BoolQ, and COPA. Table 5 shows that MT-TaG obtains impressive gains, demonstrating the model's ability in scaling to a large number of diverse tasks.

7 Related Work

Mixture-of-Experts models have recently achieved promising results by introducing an outrageously large number of parameters while keeping a fixed computation cost via gating mechanism. Shazeer et al., 2017 first proposed the MoE layer with a single gating network with Top-k routing and load balancing across experts. Fedus et al., 2021 propose initialization and training schemes for Top-1 routing. Zuo et al., 2021 propose a consistency reg-

ularizer loss for random routing; Yang et al., 2021 propose k Top-1 routing with expert-prototypes, and Roller et al., 2021; Lewis et al., 2021 address other load balancing issues. All the above works study sparse MoE with pre-training from scratch in single-task settings. In contrast, we study multi-task adaptation of such sparse models and devise task-aware gating networks to support MTL. A contemporary work (Kudugunta et al., 2021) studies routing for multi-task sequence-to-sequence training for machine translation, where they route *all* tokens from a task to the same experts with a shared gate. In contrast, we study multi-task adaptation for NLU tasks where we make routing decisions at token-level using task-specific gates. In the non-Transformer space, an earlier work Ma et al., 2018 studied MTL for tabular classification and content recommendation. In contrast to all above works, we study multi-task adaptation of sparse MoE and analyze its robustness for diverse NLU tasks.

Multi-task learning and adaptation has been studied extensively for dense models (Caruana, 1997; Crawshaw, 2020), with recent works like UnifiedQA (Khashabi et al., 2020), MT-DNN (Liu et al., 2019) and Muppet (Aghajanyan et al., 2021a) showing impressive transfer and low-resource generalization ability. MT-DNN with BERT encoder performs multi-task adaptation on a mixture of GLUE tasks and is used as our baseline. While Muppet also follows similar principles, it uses RoBERTa and much larger number of tasks (50). For a fair comparison, with limited compute, we only compare against MT-DNN with the same encoder and same set of MTL tasks. We contrast our MTL setup against the above dense MTL models and demonstrate our sparse design to be more robust on three key transferability aspects.

8 Conclusion

In this work, we studied multi-task adaptation of sparse MoE models on diverse NLU tasks when initialized with the weights of a pre-trained language model. To support multi-task learning with sparse MoE, we devised task-aware gating networks to route input tokens from different tasks to specialized experts conditioned on the task. We demonstrated such sparse design to be more robust multi-task learners than their non-MOE dense counterparts on several key dimensions including transferability, sample-efficient generalizability, and avoiding catastrophic forgetting.

Ethical Considerations and Broader Impact

In this work, we develop an efficient multi-task deep neural network model that performs well across several diverse natural language understanding tasks. One of the benefits of a multi-task model is parameter efficiency, where the same model can be used across several different tasks, thereby, saving storage cost and memory footprint. We also demonstrate improved robustness of the multi-task model that further reduces risks of deploying such models in the wild. Furthermore, improved generalization, transferability and sample-efficiency of our model is beneficial for sensitive application domains including finance, legal and healthcare.

However, our model also has the risk of echoing the biases from the pre-trained language model it is based on. Furthermore, a considerable risk with multi-task learning is that it can facilitate the propagation of biases from individual datasets from its training mixture to the rest. Sparse models like MT-TaG with their increased capability to transfer information from just a single task from its training mixture poses increased risk of retaining and transferring such biases to the unseen tasks. Sparse models also massively increase the number of parameters, which can lead to significant storage cost in the absence of customized hardware and optimized implementations, leading to a negative impact on the carbon footprint from training and deploying such models.

References

Armen Aghajanyan, Anchit Gupta, Akshat Shrivastava, Xilun Chen, Luke Zettlemoyer, and Sonal Gupta. 2021a. Muppet: Massive multi-task representations with pre-finetuning. *ArXiv*, abs/2101.11038.

Armen Aghajanyan, Anchit Gupta, Akshat Shrivastava, Xilun Chen, Luke Zettlemoyer, and Sonal Gupta. 2021b. Muppet: Massive multi-task representations with pre-finetuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5799–5811, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Roy Bar Haim, Ido Dagan, Bill Dolan, Lisa Ferro, Danilo Giampiccolo, Bernardo Magnini, and Idan Szpektor. 2006. The second pascal recognising textual entailment challenge. In *Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment*.

Luisa Bentivogli, Peter Clark, Ido Dagan, and Danilo Giampiccolo. 2009. The fifth pascal recognizing textual entailment challenge. In *TAC*.

Rich Caruana. 1997. Multitask learning. *Machine learning*, 28(1):41–75.

Daniel Matthew Cer, Mona T. Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *SemEval@ACL*.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. Boolq: Exploring the surprising difficulty of natural yes/no questions. *ArXiv*, abs/1905.10044.

Michael Crawshaw. 2020. Multi-task learning with deep neural networks: A survey. *ArXiv*, abs/2009.09796.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The pascal recognising textual entailment challenge. In *Machine Learning Challenges. Evaluating Predictive Uncertainty, Visual Object Classification, and Recognising Tectual Entailment*, pages 177–190. Springer.

Marie-Catherine De Marneffe, Mandy Simons, and Judith Tonhauser. 2019. The commitmentbank: Investigating projection in naturally occurring discourse. In *proceedings of Sinn und Bedeutung*, volume 23, pages 107–124.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *ArXiv*, abs/1810.04805.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *IJCNLP*.

William Fedus, Barret Zoph, and Noam M. Shazeer. 2021. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *ArXiv*, abs/2101.03961.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The third pascal recognizing textual entailment challenge. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*.

Raia Hadsell, Dushyant Rao, Andrei A. Rusu, and Razvan Pascanu. 2020. Embracing change: Continual learning in deep neural networks. *Trends in Cognitive Sciences*, 24:1028–1040.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. Deberta: Decoding-enhanced bert with disentangled attention. *ArXiv*, abs/2006.03654.

Shankar Iyer, Nikhil Dandekar, and Kornel Csernai. 2017. First quora dataset release: Question pairs.

Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. 2018. Looking beyond the surface: A challenge set for reading comprehension over multiple sentences. In *NAACL*.

Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. 2020. UNIFIEDQA: Crossing format boundaries with a single QA system. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1896–1907, Online. Association for Computational Linguistics.

Tushar Khot, Ashish Sabharwal, and Peter Clark. 2018. Scitail: A textual entailment dataset from science question answering. In *AAAI*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Sneha Kudugunta, Yanping Huang, Ankur Bapna, Maxim Krikun, Dmitry Lepikhin, Minh-Thang Luong, and Orhan Firat. 2021. Beyond distillation: Task-level mixture-of-experts for efficient inference. *ArXiv*, abs/2110.03742.

Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. 2021. Gshard: Scaling giant models with conditional computation and automatic sharding. In *International Conference on Learning Representations*.

Mike Lewis, Shruti Bhosale, Tim Dettmers, Naman Goyal, and Luke Zettlemoyer. 2021. Base layers: Simplifying training of large, sparse models. In *ICML*.

Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, Joe Davison, Mario vSavsko, Gunjan Chhablani, Bhavitvya Malik, Simon Brandeis, Teven Le Scao, Victor Sanh, Canwen Xu, Nicolas Patry, Angelina McMillan-Major, Philipp Schmid, Sylvain Gugger, Clement Delangue, Th'eo Matussiere, Lysandre Debut, Stas Bekman, Pierric Cistac, Thibault Goehringer, Victor Mustar, Franccois Lagunas, Alexander M. Rush, and Thomas Wolf. 2021. Datasets: A community library for natural language processing. In *EMNLP*.

Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2019. Multi-task deep neural networks for natural language understanding. In *ACL*.

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

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, A. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *ACL*.

Brian W Matthews. 1975. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure*, 405(2):442–451.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *ACL*.

Mohammad Taher Pilehvar and José Camacho-Collados. 2019. Wic: the word-in-context dataset for evaluating context-sensitive meaning representations. In *NAACL*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. In *EMNLP*.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S Gordon. 2011. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *2011 AAAI Spring Symposium Series*.

Stephen Roller, Sainbayar Sukhbaatar, Arthur D. Szlam, and Jason Weston. 2021. Hash layers for large sparse models. *ArXiv*, abs/2106.04426.

Noam M. Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc V. Le, Geoffrey E. Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *ArXiv*, abs/1701.06538.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, A. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *ArXiv*, abs/1804.07461.

Wenhui Wang, Hangbo Bao, Shaohan Huang, Li Dong, and Furu Wei. 2021. Minilmv2: Multi-head self-attention relation distillation for compressing pretrained transformers. In *FINDINGS*.

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

10

Adina Williams, Nikita Nangia, and Samuel R. Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *NAACL*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Yonghui Wu, Mike Schuster, Z. Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason R. Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Gregory S. Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *ArXiv*, abs/1609.08144.

An Yang, Junyang Lin, Rui Men, Chang Zhou, Le Jiang, Xianyan Jia, Ang Wang, Jie Zhang, Jiamang Wang, Yong Li, et al. 2021. M6-t: Exploring sparse expert models and beyond. *arXiv preprint arXiv:2105.15082*.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *ArXiv*, abs/1509.01626.

Simiao Zuo, Xiaodong Liu, Jian Jiao, Young Jin Kim, Hany Hassan, Ruofei Zhang, Tuo Zhao, and Jianfeng Gao. 2021. Taming sparsely activated transformer with stochastic experts. *ArXiv*, abs/2110.04260.

11

A Appendix

A.1 Analysis

A.1.1 Re-using task gates for generalization

In Table 6, we provide results for fine-tuning the GLUE-trained MT-TaG model on unseen SciTail dataset with different task gates. We observe that re-using the gates corresponding to the related tasks (RTE, MNLI) outperforms the random initialization of the gate, as well as re-using the gate from an unrelated task (SST-2). This demonstrates MT-TaG's ability in learning task-specific routing in its gates, and efficiently re-using it for generalizing to unseen related tasks in a sample efficient manner.

Task Gate	Accuracy
Random	91.2
SST-2	91.8
RTE	92.6
MNLI	**92.9**

Table 6: Performance of MT-TaG when fine-tuned with different task gates on the 10% sample of the unseen SciTail dataset. Gates corresponding to tasks with similarity to SciTail (RTE and MNLI) perform superior to random and unrelated task gates (SST-2). All results are with the MiniLM encoder.

A.1.2 Task Sampling

In Table 7, we provide results for using different task sampling strategies while training MT-TaG with heterogeneous batches. We observe that maintaining the natural distributions of tasks during MTL training outperforms uniformly sampling all tasks. We thus use natural sampling of tasks for the MTL models in our experiments.

Sampling	Small Tasks	All Tasks
Uniform	80.60	85.75
Natural	**82.73**	**86.94**

Table 7: Comparison of task sampling strategies in MT-TaG with the BERT$_{Large}$ encoder on GLUE. Maintaining the natural distribution of tasks (*Natural Sampling*) outperforms uniformly sampling tasks (*Uniform Sampling*).

A.1.3 Number of Experts

In Table 8, we provide results for using different number of experts in MT-TaG. We observe 4 experts to perform the best, and thus use 4 experts for all sparse model experiments.

#experts	Small Tasks	All Tasks
2 experts	80.78	85.79
4 experts	**82.73**	**86.94**
6 experts	80.60	85.76

Table 8: MT-TaG's performance comparison on GLUE with different number of experts (#experts) using the BERT$_{Large}$ encoder. 4 experts performs the best.

A.2 Datasets

Below, we provide details about all the datasets that we used. We also summarize the key information about these datasets in Table 9.

RTE: Recognizing Textual Entailment are datasets collected from a series of annual textual entailment challenges. The authors combine the data from RTE1 (Dagan et al., 2006), RTE2 (Bar Haim et al., 2006), RTE3 (Giampiccolo et al., 2007), and RTE5 (Bentivogli et al., 2009). All datasets are converted to two-class classification: entailment and not entailment.

MRPC: Microsoft Research Paraphrase Corpus (Dolan and Brockett, 2005) is a corpus of sentence pairs automatically extracted from online news sources, with human annotations for whether the sentences in the pair are semantically equivalent.

STS-B: Semantic Textual Similarity Benchmark (Cer et al., 2017) is a collection of sentence pairs drawn from news headlines, video and image captions, and natural language inference data. Each pair is human-annotated with a similarity score from 1 to 5.

CoLA: Corpus of Linguistic Acceptability (Warstadt et al., 2019) consists of English acceptability judgments drawn from books and journal articles on linguistic theory. Each example is a sequence of words annotated with whether it is a grammatical English sentence.

SST-2: Stanford Sentiment Treebank (Socher et al., 2013) consists of sentences from movie reviews and human annotations of their sentiment. The task is to predict the sentiment of a given sentence. It uses the two-way (positive/negative) class split,

with only sentence-level labels.

QNLI: Stanford Question Answering Dataset (Wang et al., 2018; Rajpurkar et al., 2016) is a question-answering dataset consisting of question-paragraph pairs, where one of the sentences in the paragraph (drawn from Wikipedia) contains the answer to the corresponding question (written by an annotator). The authors of the benchmark convert the task into sentence pair classification by forming a pair between each question and each sentence in the corresponding context, and filtering out pairs with low lexical overlap between the question and the context sentence. The task is to determine whether the context sentence contains the answer to the question. This modified version of the original task removes the requirement that the model select the exact answer, but also removes the simplifying assumptions that the answer is always present in the input and that lexical overlap is a reliable cue.

QQP: Quora Question Pairs2 dataset (Iyer et al., 2017) is a collection of question pairs from the community question-answering website Quora. The task is to determine whether a pair of questions are semantically equivalent.

MNLI: Multi-Genre Natural Language Inference Corpus (Williams et al., 2018) is a crowdsourced collection of sentence pairs with textual entailment annotations. Given a premise sentence and a hypothesis sentence, the task is to predict whether the premise entails the hypothesis (entailment), contradicts the hypothesis (contradiction), or neither (neutral). The premise sentences are gathered from ten different sources, including transcribed speech, fiction, and government reports. The authors of the benchmark use the standard test set, for which they obtained private labels from the RTE authors, and evaluate on both the matched (in-domain) and mismatched (cross-domain) section.

CB: Commitment Bank (De Marneffe et al., 2019) is a corpus of short texts in which at least one sentence contains an embedded clause. Each of these embedded clauses is annotated with the degree to which it appears the person who wrote the text is committed to the truth of the clause. The resulting task framed as three-class textual entailment on examples that are drawn from the Wall Street Journal, fiction from the British National Corpus, and Switchboard. Each example consists of a premise containing an embedded clause and the corresponding hypothesis is the extraction of that clause.

BoolQ: Boolean Questions (Clark et al., 2019) is a QA task where each example consists of a short passage and a yes/no question about the passage. The questions are provided anonymously and unsolicited by users of the Google search engine, and afterwards paired with a paragraph from a Wikipedia article containing the answer.

MultiRC: Multi-Sentence Reading Comprehension (Khashabi et al., 2018) is a QA task where each example consists of a context paragraph, a question about that paragraph, and a list of possible answers. The system must predict which answers are true and which are false. Each question can have multiple possible correct answers, so each question-answer pair must be evaluated independent of other pairs. The questions are also designed such that answering each question requires drawing facts from multiple context sentences. The paragraphs are drawn from seven domains including news, fiction, and historical text.

WiC: Word-in-Context (Pilehvar and Camacho-Collados, 2019) is a word sense disambiguation task cast as binary classification of sentence pairs. Given two text snippets and a polysemous word that appears in both sentences, the task is to determine whether the word is used with the same sense in both sentences.

COPA: Choice of Plausible Alternatives (Roemmele et al., 2011) is a causal reasoning task in which a system is given a premise sentence and must determine either the cause or effect of the premise from two possible choices. All examples are handcrafted and focus on topics from blogs and a photography-related encyclopedia.

IMDB: Large Movie Review Dataset (Maas et al., 2011) built from reviews from IMDb (Internet Movie Database). This is a dataset for binary sentiment classification containing highly polar movie reviews.

Yelp Polarity: Large Yelp Review Dataset (Zhang et al., 2015). This is a dataset for binary sentiment classification constructed from highly polar Yelp reviews.

Rotten Tomatoes: Movie Review Dataset (Pang and Lee, 2005). This is a dataset of containing positive and negative processed sentences from Rotten Tomatoes movie reviews.

SciTail: SciTail (Khot et al., 2018) dataset is an entailment dataset created from multiple-choice science exams and web sentences. Each question and the correct answer choice are converted into an

13

Dataset	#Train	#Dev	#Labels	Formulation	Metrics
WSD					
WiC	5.4k	638	2	Pairwise-text Classification	Accuracy
Similarity					
STS-B	5.7k	1.5k	1	Pairwise-text Regression	Spearman corr
Acceptability					
CoLA	8.5k	1k	2	Single-text Classification	Matthews corr
Sentiment					
Rotten Tomatoes	8.5k	1k	2	Single-text Classification	Accuracy
IMDB	25k	25k	2	Single-text Classification	Accuracy
SST-2	67.3k	872	2	Single-text Classification	Accuracy
Yelp Polarity	560k	38k	2	Single-text Classification	Accuracy
Paraphrase					
MRPC	3.7k	408	2	Pairwise-text Classification	Accuracy
QQP	364k	40k	2	Pairwise-text Classification	Accuracy
NLI					
CB	250	56	3	Pairwise-text Classification	Accuracy
RTE	2.5k	277	2	Pairwise-text Classification	Accuracy
SciTail	23.6k	1.3k	2	Pairwise-text Classification	Accuracy
QNLI	105k	5.5k	2	Pairwise-text Classification	Accuracy
MNLI	393k	9.8k	3	Pairwise-text Classification	Accuracy
QA					
COPA	400	100	2	Pairwise-text Ranking	Accuracy
MultiRC	27k	4.8k	2	Pairwise-text Classification	F1a
BoolQ	9.4k	3.3k	2	Pairwise-text Classification	Accuracy

Table 9: Key information about all the datasets used.

assertive statement to form the hypothesis. Information retrieval is used to obtain relevant text from a large text corpus of web sentences, and use these sentences as a premise. Premise-hypothesis pair are annotated as supports (entails) or not (neutral).

We obtained all of these datasets from Hugging-Face's datasets library (Lhoest et al., 2021).

A.3 Implementation Details

A.3.1 Task formulations

In this section, we group all the tasks into different categories, and provide details about their formulation. All model variants followed BERT-like architectures (Devlin et al., 2019) with a [CLS] token added to the beginning of the input.

Single-text Classification

CoLA, SST-2, IMDB, Yelp Polarity, and Rotten Tomatoes belong to this category. The task is to perform binary classification based on a single sequence of concatenated sentences. A classifier head is used on top of the output representation of the [CLS] token for the classification. We use Matthews correlation coefficient (Matthews, 1975) as the evaluation metric for CoLA, and use accuracy for the rest.

Pairwise-text Classification

RTE, MRPC, QNLI, QQP, MNLI, CB, BoolQ, MultiRC, WiC, and SciTail belong to this category. The task is to perform binary or multi-class classification based on a pair of sequence inputs. We concatenate the input sequence pairs separated by a [SEP] token following (Devlin et al., 2019), and feed the fused sequence to the model. In the case of MultiRC, which contains three sequences (paragraph, question, and answer), the paragraph and question are concatenated to form the first sequence, and the answer is used as the second sequence. For all tasks except WiC, a classifier head which sees the output representation corresponding to the [CLS] token is used to select the predicted class. For WiC a span classification head is used, which extracts the output representations of the word of interest (from both input sentences) and concatenates them with the representation of the [CLS] token. This fused representation is then fed to a classifier head to predict the binary output. Following the authors, we use $F1_a$ as the metric for MultiRC, which evaluates binary decisions on all the answer-options in the dataset independently. $F1_a$ is the harmonic mean of precision and recall

across all answer-option pairs, without grouping by question or paragraph. For all other tasks, accuracy is used as the evaluation metric.

Pairwise-text Ranking

COPA belongs to this category. The task is to choose between a pair of sequences given a premise-question context. We join the premise-question sequence pair into a single context sequence, and evaluate each pair of choice alternatives independently by concatenating context, [SEP] token, and answer choice to form a pair of input sequences for the model. The task is then cast as a binary classification task for each input pair, for which we feed the output representation to a classifier head, and retrieve the positive (True) class logits for each input. Whichever input returns the largest positive-class logit is then taken as the answer choice, and we calculate accuracy as the evaluation metric.

Pairwise-text Regression

STS-B belongs to this category. The task is to perform regression from a pair of input sequences. The input sequences are concatenated together with a [SEP] token and fed to the model. A regression head is used to learn the similarity score and we calculate Spearman's rank correlation as the evaluation metric.

A.3.2 Model details

We use a Wordpiece Tokenizer (Wu et al., 2016) with 30k vocabulary size to tokenize all the examples. We truncate the examples on the right using a maximum length of 512 for QNLI and MNLI, and 128 for the rest of GLUE datasets. We use a batch size of 128 for MTL Training, and 32 for fine-tuning.

For training of Sparse models, we do not add any additional load balancing loss, input jitter, or additional dropout in the experts[6]. *Unlike existing work, we did not encounter a load-imbalance in the utilization of the experts, potentially due to the multi-task objective that pushes the network to specialize weights in different experts.*

Model selection

For MTL training, we train the model for a fixed number of steps, and select the checkpoint at the end of training. For fine-tuning, we use early stopping using the dev set. We tune the learning rate,

---

[6]Early experiments resulted in a drop in the performance.

15

warmup proportion, and the number of training steps for both MTL Training and fine-tuning. For fine-tuning, tuning is only done for the small tasks ($< 10k$ examples)[7]. For every task, we run 3 fine-tuning experimental runs for each model with different seeds, and report the max number obtained across runs for the model.

Hyper-parameters and Tuning

For the Adam optimizer, we used β_1 and β_2 values of 0.9 and 0.999 respectively, and an ϵ of $1e - 8$. For MTL Training, we ran tuning runs with a grid search of the learning rate in $[5e - 06, 1e - 05, 2e - 05, 5e - 05, 1e - 04]$, warmup rate in $[0.1, 0.2]$, and number of steps in $[30k, 50k]$. For fine-tuning, we tuned the learning rate in $[5e - 06, 1e - 05, 2e - 05, 5e - 05, 1e - 04]$, used a warmup of 0.1, and tuned the number of epochs in $[5, 10, 15, 20, 25, 30]$.

A.4 Limitations and Future Work

Using a separate gate for each task allows us to learn task-specific routing in the gates, however, it has the limitation that individual gates are only updated via the examples corresponding to their target task. This can lead to the gates for the smallest tasks being under-trained under a natural sampling of tasks. In the future, we will experiment with a training schedule in which we use uniform sampling at the beginning of training to allow all gates to train sufficiently, and then revert back to natural sampling. Our method also has the limitation that gates of related tasks only share information via the experts. To tackle this, we will experiment with incorporating task embeddings to allow the network to share routing information by learning similar task embeddings for related tasks. Lastly, we will experiment with further scaling up the number and diversity of tasks in our multitask mixture to obtain a general model for a wide-range of downstream tasks.

A.5 Task-level Results

A.5.1 Robustness to unrelated tasks

We provide the task-level results corresponding to the robustness experiments from Section 5.3 in Table 10.

A.5.2 Encoder Scaling

We provide the task-level results corresponding to the encoder scaling experiments from Section 6.1 in Table 11.

---

[7]Bigger tasks showed indifference to the choice of hyper-parameters.

16

Dataset	RTE (2.5k)	MRPC (3.7k)	STS-B (5.7k)	QNLI (105k)	QQP (364k)	MNLI (393k)	Small Tasks (Avg.)	All Tasks (Avg.)
				MT-Dense				
C-GLUE	**78.6**	**89.7**	90.5	89.8	**90.9**	83.6	**86.27**	**87.18**
GLUE	77.9	89	90.5	**90.3**	90.8	**83.8**	85.80	87.05
				MT-Switch				
C-GLUE	78.9	89.5	90.4	90.1	90.9	83.5	86.27	87.22
GLUE	78.9	**90**	**90.5**	**90.3**	90.9	**83.6**	**86.47**	**87.37**
				MT-TaG				
C-GLUE	78.2	**90.9**	90.4	90	90.8	83.6	86.50	87.32
GLUE	**81.1**	90.7	**90.6**	**90.2**	90.8	83.6	**87.47**	**87.83**

Table 10: Task-level model performance on GLUE (containing several diverse tasks) and C-GLUE (as a subset of GLUE containing only related tasks) evaluated on the common tasks in both. Sparse MTL models demonstrate robustness in the presence of unrelated tasks in GLUE, with MT-TaG with task-specific routing being the most robust. All models use MiniLM encoder.

Model	RTE (2.5k)	MRPC (3.7k)	STS-B (5.7k)	CoLA (8.5k)	SST-2 (67.3k)	QNLI (105k)	QQP (364k)	MNLI (393k)	Small Tasks (Avg.)	All tasks (Avg.)
					$BERT_{Base}$					
Single-Task	71.4	84.8	89.1	<u>60.8</u>	**92.9**	**91.9**	**91.4**	84.4	76.53	83.34
MT-TaG	**81.1**	**90.7**	**90.4**	<u>60.7</u>	92.9	91.8	**91.4**	84.6	**80.73**	**85.45**
					$BERT_{Large}$					
Single-Task	74.6	88.2	89.9	62.7	93.3	<u>92.7</u>	**91.7**	86.3	78.85	84.93
MT-TaG	**86.4**	**89.2**	**90.8**	**64.5**	<u>94.2</u>	<u>92.3</u>	**91.7**	<u>86.4</u>	**82.73**	**86.94**
MT-DNN	83.4	87.5	90.6	63.5	**94.3**	**92.9**	89.2	**86.9**	81.25	86.04

Table 11: Task-level performance of models with different BERT encoder sizes. MT-TaG shows consistent gains across encoders of different sizes. MT-TaG also outperforms the dense MTL baseline MT-DNN (Liu et al., 2019).