Outrageously Fast LLMs: Faster Inference and Fine-Tuning with Moe*fication* and LoRA

Stanford CS224N Custom Project
Mentor: Tony Wang

Chi Tsai[*]
Department of Computer Science
Stanford University
chiyotsai@stanford.edu

Jay Martin[†]
Department of Computer Science
Stanford University
jaydm@stanford.edu

Abstract

In recent years, transformer large language models (LLMs) have achieved wide success in a variety of downstream tasks, in large part by dramatically scaling up parameter counts. This, however, greatly increases the computational burden of training and running models. In this project, we propose a method that balances performance, inference complexity, and training complexity through a two-stage process. First, we reduce the inference complexity by converting the feed-forward networks in a pretrained LLM into a Mixture of Experts (MoE) through a process called MoE*fication*. This leads to reductions in performance on downstream tasks, however, so we utilize Low-Rank Adapatation (LoRA) matrices to efficiently fine-tune the MoE*ified* model. Our results demonstrate that we can recover nearly all of the performance loss, resulting in a pipeline for reducing inference time with minimal loss in model quality.

1 Introduction

The remarkable recent successes of language models come in no small part from exploiting scale in both training data and model size (Han et al., 2021). Model performance on downstream tasks is highly correlated with the number of model parameters and size of the dataset used for training (Kaplan et al., 2020; Aghajanyan et al., 2023), and as a result the principal driver of recent accomplishments in language modeling has been the continuous release of ever-larger models, and the path forward seen by a number of noted researchers is not merely large models but *outrageously* large models (Shazeer et al., 2017).

This trend, however, greatly increases the computational resources needed to train and run language models.

Consequently, a great deal of effort has gone into making transformer models more efficient, utilizing techniques like model pruning and attention approximation (Tay et al., 2022; Pope et al., 2023). In particular, one approach that has seen a recent resurgence since it was first proposed in the 1990s is Mixture of Experts (MoE) (Jacobs et al., 1991), in which a model consists of distinct "experts", only a subset of which are activated on a given input. This enables training very large models—and benefiting from the enhanced performance brought by scale—while reducing the computational burden of running such large models by only activating parameters that are most relevant to the input (Masoudnia and Ebrahimpour, 2014).

But a problem still remains: *training* large MoE models can still be prohibitively expensive and resource intensive. This is particularly the case for those with limited GPUs available, putting state-of-the-art models out of reach for many researchers.

[*]Responsible for implementing the training and inferencing frameworks, hyperparameter sweeping.
[†]Responsible for project scoping, initial implementation of LoRA, benchmarking.

To address this, we develop a pipeline for converting outrageously large language models into outrageously fast ones with minimal reduction in performance. We start from the view that the feed-forward networks (FFNs) of transformers act as *stores of knowledge* (Dai et al., 2021). In particular, the FFNs can be viewed as mixtures-of-experts, i.e. particular clusters of neurons hold information on particular topics or knowledge categories and each cluster is largely independent of the others (Suau et al., 2020). This enables a process of MoE*fication*, proposed by Zhang et al. (2022), that splits the FFNs of a model into a mixture-of-experts. This process is imperfect, however, and there is a reduction in performance on downstream tasks as a result of the process. We combat this by utilizing low-rank adaption (LoRA) matrices (Hu et al., 2021) to fine-tune the MoE*fied* modes. We show that we can recover nearly all of the performance loss in a parameter-efficient way.

2 Related Work

2.1 Model Acceleration

Model acceleration is a lively area of research that seeks to reduce the time and/or space complexity of models (Choudhary et al., 2020). Important approaches include knowledge distillation, where knowledge is transferred from a larger, more complex model (the "teacher") to a smaller, simpler model (the "student") (Sanh et al., 2019; Jiao et al., 2020); model pruning, a compression technique where parameters are systematically removed from a model utilizing strategies that minimize loss in capability (Zhu and Gupta, 2017; Voita et al., 2019; Lin et al., 2020; Zhang et al., 2021); attention approximation, where attention is approximated to reduce computational overhead from attention computations (Wang et al., 2020; Kitaev et al., 2020; Ham et al., 2020; Choromanski et al., 2020); model quantization, where floating-point precision is reduced by mapping continuous floating-point values to a finite set of discrete values (Rokh et al., 2023; Polino et al., 2018; Bai et al., 2021); and dynamic inference, where the computational graph of a model adjusts dynamically based on the input at inference time (Xia et al., 2022; Xin et al., 2020; Hou et al., 2020).

2.2 Mixture of Experts

One approach to model acceleration that this project is based around is Mixture of Experts (MoE), part of an avenue of research known as "conditional computation" that seeks to take advantage of the greater capabilities of very large models while reducing training and inference speed by only activating a subset of a model's parameters on any given input (Bengio, 2013; Bengio et al., 2015). MoE is an approach that precedes modern deep learning, having first been proposed in 1991 by Jacobs et al. (1991) to speed up supervised training and inference. It has since become an active area of research in deep learning and has shown success at reducing computational resources for very large models (Masoudnia and Ebrahimpour, 2014). Notable recent MoE models include the Switch-Transformer (Fedus et al., 2021), BASELayer (Lewis et al., 2021), Hash-Layer (Roller et al., 2021) and GLaM (Du et al., 2022). There is ongoing research into architectural enhancements of MoE models, like improved expert routing (Zhou et al., 2015), differentiable gating (Hazimeh et al., 2021), improved token-expert allocation (Lewis et al., 2021), and hardware optimization (He et al., 2021).

2.3 Efficient Fine-Tuning

A related area of research that has become particularly active in this era of scale is efficient fine-tuning (Chen et al., 2023). A number of approaches have been proposed, e.g. adapter tuning (Houlsby et al., 2019), prefix and prompt tuning (Li and Liang, 2021; Lester et al., 2021), and BitFit (Zaken et al., 2021). One approach that has taken on particular significance and which is utilized by this project is low-rank adaptation (LoRA), which reduces fine-tuning to training low-rank matrices (Hu et al., 2021). The introduction of LoRA has spawned its own subfield of research, with new variants of LoRA proposed often, e.g. MoLE (Wu et al., 2024).

2.4 Interpretation of Transformer Feed-Forward Networks

As a result of the success of transformer models, substantial research has gone towards developing a theoretical understanding of their operation (Wallace et al., 2019; Kovaleva et al., 2019; Wang and Tu,

2020) and their linguistic understanding (Ramnath et al., 2020; Manning et al., 2020). Particular focus has been placed on understanding the attention mechanism (Voita et al., 2019; Vig and Belinkov, 2019; Clark et al., 2019), but recent work has also examined their feed-forward layers (Geva et al., 2022).

A general understanding has developed that the FFNs act as *stores of memory* (Dai et al., 2021). Geva et al. (2021), for example, conceptualize the FFNs as "key-value memories". Of particular relevance to this project is the view that these stores of knowledge can be viewed as distinct experts—that particular clusters of neurons hold information on particular topics or knowledge categories (Suau et al., 2020). Evidencing this, Zhang et al. (2022) find that only small portions of a transformer's FFNs are activated on any given input and that these function as coherent units largely independent of other parameters within the FNNs. This understanding has enabled a variety of interesting research paths, like direct editing of factual knowledge through clever manipulation of the FFN parameters (Cao et al., 2021; Meng et al., 2022).

2.5 MoE*fication*

This project makes particular use of an approach based on this understanding of FFNs called MoE*fication* (Zhang et al., 2022). With MoE*fication*, the authors show that the FFNs of a *pre-trained* model can be partitioned into distinct, sparsely activated experts with only modest reductions in performance on downstream tasks. This enables researchers to take existing state-of-the-art models that have already been trained and increase their computational efficiency by reducing the number of FLOPS required at inference.

3 Approach

Our project is divided into two phases. First, we use MoE*fication* to convert the FFNs of a pretrained model into a mixture-of-experts, which increases inference speed but results in modest loss in performance on downstream tasks. Second, we utilize LoRA fine-tuning to recover model quality with limited computational overhead.

MoE*fication* We follow the approach from Zhang et al. (2022) to MoE*ify* a baseline model. Additionally, we utilize He et al. (2021)'s FastMoE PyTorch framework to increase MoE efficiency.

In a transformer model, the output from the attention layer is commonly followed by a DenseActDense layer that processes the attention output as $y = W_o\sigma(W_i x)$, where $W_o \in \mathbb{R}^{d\times h}, W_i \in \mathbb{R}^{h\times d}$, and σ is an elementwise non-linear activation. In our formulation, we reorder and partition W_o into k parts:

$$\tilde{W}_o = \begin{bmatrix} W_o^{(1)} & \dots & W_o^{(k)} \end{bmatrix} \text{ and } \tilde{W}_i = \begin{bmatrix} W_i^{(1)} & \dots & W_i^{(k)} \end{bmatrix}^\top$$

so that $y = W_o\sigma(W_i x) = \sum_{j=1}^k W_o^{(j)}\sigma(W_i^{(j)}(x))$.

We refer to each pair of $W_i^{(j)}, W_o^{(j)}$ as an expert. We use k-means clustering to create the k matrices, each of the same dimension.

Next, the experts need to be sparsely activated to reduce inference complexity, so we create a routing function $g : \mathbb{R}^d \to \mathbb{R}^k$ to select the top k experts. Our g is a 2-layer fully-connected network that takes x as input, and tries to predict $\mathbf{1}^\top\sigma(W_i^j x)$ for each j. g is trained offline by collecting the values of the outputs from the attention layer on a training set. Finally, our MoE*fied* model sums up the top-k experts, which are selected from the output of $g(x)$. Mathematically, the MoE*fied* output is $\tilde{y} = \sum_{j=1}^k \text{hard-top-n}(g(x))W_o^{(j)}\sigma(W_i^{(j)}(x))$ where $\text{hard-top-n}(x)_j = \begin{cases} 1 & \text{if } |\{x_l : x_l > x_j\}| \le n \\ 0 & \text{otherwise} \end{cases}$

LoRA Finetuning Since the process of MoE*fication* is lossy, we attempt to recover the performance loss with low-rank adaptors. Due to the sparsity of fine-tuning data, we adapt the idea of Low Rank Adaption matrices (Hu et al., 2021), and make some additional change to preserve our sparse mixture of experts structure in our model.

A naive application of LoRA to our model would introduce two set of matrices $A_i \in \mathbb{R}^{d\times r}, B_i \in \mathbb{R}^{r\times h}, A_o \in \mathbb{R}^{h\times r}, B_o \in \mathbb{R}^{r\times d}$, and modify the feedforward structure of the transformer as

$\hat{y} = (W_o + A_oB_o)\sigma((W_i + A_iB_i)x) = W_o\sigma(W_i x) + A_oB_o\sigma(A_iB_i x) + W_o\sigma(A_iB_i x) + A_oB_o\sigma(W_i x)$

Unfortunately, this is not compatible with sparse mixture of experts, as the effect of the lora matrices would spill over the experts due to the crossover terms $W_o\sigma(A_iB_ix) + A_oB_o\sigma(W_ix)$.

Our main innovation is to instead introducing a single set of matrices $A \in \mathbb{R}^{d\times(h/k)}, B \in \mathbb{R}^{(h/k)\times d}$, and modify the feedforward layer as

$$\tilde{y} = \sum_{j=1}^{k} \text{hard-top-n}(g(x))W_o^{(j)}\sigma(W_i^{(j)}(x)) + A\sigma(Bx)$$

They key idea here is to introduce an always-active learned expert $A\sigma(Bx)$ that is specialized for a particular task.

To establish a baseline, we forked the MoE-*fication* codebase for the MoE-*fication* script. We also added an additional pipeline to support arbitrary T5 models, MoE-*fication* on the SST2, MNLI, and RACE datasets, and various fine-tuning supports. The code is publicly available on github here: https://github.com/chiyotsai/MoEfication-with-LoRA.

4 Experiments

4.1 Data

To measure the quality improvement, we finetune and measure the accuracy of our method on three different classification tasks: Sentiment Classification on the Stanford Sentiment Tree Bank 2 (SST2) dataset (Socher et al., 2013), Natural Language Inference on the Multi-Genre Natural Language Inference (MNLI) dataset (Williams et al., 2018), and Multiple Choice Question Answering on the ReAding Comprehension From Examination (RACE) (Lai et al., 2017) dataset. The number of examples in each is presented in Table 1.

	Training Set	Validation Set	Test Set
SST2	67.3k	872	1.82k
MNLI	393k	9.82k	9.83k
RACE	87.9k	4.89k	4.93k

Table 1: Number of examples in each dataset

The T5 models were pretrained on a dataset mixture that contains SST2 and MNLI but not RACE (Colin et al., 2019). Consistent with this, we performed full fine-tuning on RACE to establish a benchmark, while fine-tuning was not necessary for SST2 and MNLI.

T5 is a text-to-text model (meaning every task, whether it be autoregressive text generation, classification, translation etc. is presented as feeding the model input text and generating output text), so the datasets require preprocessing. See appendix A.1

4.2 Evaluation Method

4.2.1 Quality Evaluation

As the three tasks of interest are classification tasks, we use total accuracy as the principle metric for evaluation. The output classes for each make this an easy measurement to obtain: SST2 has output classes are 0 (negative) and 1 (positive); MNLI has output classes "entailment", "neutral", and "contradiction"; and RACE has output classes are "A", "B", "C", "D". Although other metrics such as F1-score and AUROC are also commonly used for evaluating classification performance, these are not considered as the output classes are uniformly distributed.

To obtain a prediction from our language model, we first run the encoder on the input prompts and cache the encoder output state. Next, we use the decoder and the encoder output state to get the probability of each possible output classes. These probabilities are next normalized across the output classes. Finally, we pick the class with the highest output probability as the prediction output.

These predictions are compared with the correct labels to compute the accuracy.

4.2.2 Efficiency Evaluation

We benchmark the inference time of our models on a Nvidia RTX 3090. Wall time is used as the measure of complexity since difficulty of parallelizing Mixture-of-Experts is still an actively researched topic.

Algorithm 1 Pseudocode for benchmarking MoE model performance

```
Require: Model $f$, dataset $\mathcal{D}$, number of iterations $N$
  times = []
  for $i$ in range($N$) do
      Sample and tokenize x from $\mathcal{D}$
      Call torch.cuda.synchronize()
      start_time ← time.time()
      Call $f(x)$
      Call torch.cuda.synchronize()
      end_time ← time.time()
      times.append(end_time − start_time)
  end for
  return times[10:N].mean()        ▷ Discard the first 10 samples to allow GPU to warm up.
```

4.3 Experimental Details

We use pre-trained T5 models from Huggingface as the baseline. All performance quality experiments are performed using T5-base, which contains $h = 3072$ hidden neurons in each of its DenseActDense layers.

For MoE*fication*, we use constrained k-means clustering to partition the 3072 hidden neurons into $k = 96$ experts, with each expert containing 32 neurons. Then, we run inference on the datasets to collect snapshots of the neurons in feedforward layers. These snapshots are used to train a 2-layer feed-forward layer that acts as the routing function, which selects $n = 20$ of the experts.

As T5 has not been fine-tuned on the RACE dataset, we performed an additional full finetuning on RACE to serve as the baseline of our experiment. This finetuning is done with constant learning rate of 1e-4 over 3 epochs. The checkpoint with the best accuracy is chosen as the baseline model.

For LoRA-finetuning, we apply rank-32 LoRA matrices to the DenseActDense layers. Rank 32 is chosen as it has the same rank as an expert in our config. We finetune the model on the all three datasets for 3 epochs, with a learning rate of 1e-3. A constant scheduler is used as applying learning rate decay did not appear to improve the performance in our experiment.

Furthermore, we pick the largest batch size we could fit in our GPU, with the input length chosen to cover at least 99% of the training data. All tokens above the input length are truncated. The numbers are shown in Table 2. On SST2, we clip the input size to 512 tokens, with a batch size of 64. On

	SST2	MNLI	RACE
Batch Size	64	8	4
Input Length	512	1024	1536

Table 2: Dataset parameters used to finetune T5-base.

MNLI, we clip the input size to 1024 tokens, with a batch size of 8. On RACE, we clip the input size to SST2 and MNLI for epochs with a batch size of 64, and RACE for 2 epochs with a batch size of 4.

To benchmark the complexity of our MoE models, we keep the ratio of activated experts constant (20%), and benchmark our model by following Algorithm 1.

Model	SST2	MNLI	RACE
T5-Base	93.9%	85.6%	71.6%
+MoEfied GT	93.5%	85.0%	70.1%
+MLP Routing	92.3%	83.5%	67.5%
+LoRA	93.1%	85.3%	69.7%
+FFT	93.2%	84.5 %	67.9%

Table 3: Accuracy of T5-Base on SST dataset, with MoEfication and further fine-tuning.

Model	T5-Small	T5-Base	T5-Large	T5-3B
Baseline	100.0%	100.0%	100.0%	100.0%
+MoE MLP	134.8%	98.8%	85.8%	71.8%
+LoRA	135.1%	101.4%	88.9 %	71.1%

Table 4: Inference time of modified T5 on RTX 3090, normalized by the baseline inference time.

4.4 Results

4.4.1 Quality Results

Table 3 shows the comparison of our approach's classification accuracy with that of the baseline model. The row "T5-Base" shows the baseline model performance. The row "MoE*fied* GT" shows a genie-assisted bound on the maximum accuracy achievable from a given class of MoE-model. This bound is obtained by first running a full forward pass to record the feedforward layer's output, then selecting a subset of experts that well-approximates the full forward pass's output measured by L_2 distance. The row "MLP Routing" shows the performance of using a routing function using a multi-layer perception with a single hidden layer. The row "LoRA" shows the performance of applying our expert-style LoRA to the MoE*fied* model with MLP routing function.

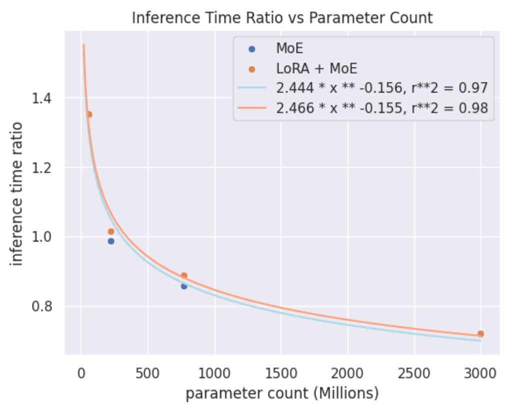
Finally, to provide a comparison between our expert-style LoRA with full finetuning, the row "FFT" shows the performance of full finetuning on the MoE*fied* model with MLP routing function.

Our number shows that applying only a single LoRA that's equivalent to an additional expert, we could significantly improve the performance of MoE*fied* model, and recover most of the performance lost in the routing function training process. Our classification accuracy is competitive with the genie-assisted bound, and even surpasses it on the MNLI dataset. The comparison with fully finetuned model also shows that increasing the number of tunable parameter in a MoE*fied* cannot recover the performance lost to the Moe*fication* process. Rather, we need to increase the information passes through the MoE*fied* feedforward layers.

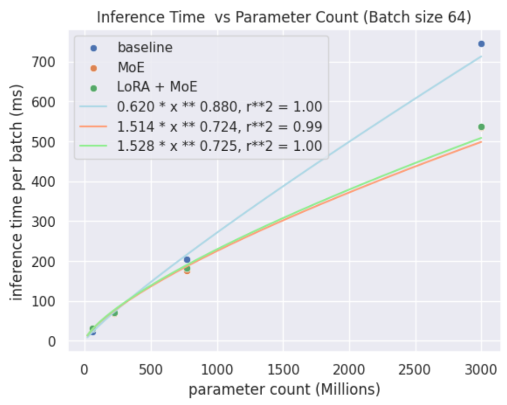
4.4.2 Complexity Results

Table 4 shows the complexity ratio of our MoE-fied model against the baseline model. Somewhat surprisingly, our experiment shows that there is no efficiency gain from MoE*fying* T5-Base. In the case of MoE with LoRA, we even incur some performance loss.

Motivated by this phenomenon, we decided to rerun the performance benchmark across different sizes of T5 models, and plotted the results in figure 1a and 1b. The figures show that the overhead of Mixture-of-Experts results in increased inference time for small models, but these overhead becomes more negligible as model size increases. The relationship between inference time and model size can be well-modeled by a function in the form ax^b, with Pearson's correlation being greater than or equal to 97% in all cases. Furthermore, the break-even point for MoEfication is coincidentally around the size of T5-Base, the model we performed the experiment on.



(a) Ratio of inference time of MoEfied models as a function of model size. Note that both MoE models and MoE + LoRA models are well approximated by $O(\text{size}^{-0.15})$.



(b) Inference time of MoEfied models as a function of model size. Note that the baseline model scales as $O(\text{size}^{0.88})$, and MoE models and MoE + LoRA models are both $O(\text{size}^{0.725})$.

Figure 1: Inferece time of MoE*fied* models.

5 Analysis

5.1 Complexity Analysis

We performed further benchmarking based on the results in Table 4 and found that most of the overhead is due to the need of scattering and gathering the feedforward layers' inputs to the different experts in the GPUs. The additional time spent on computing the routing function is relatively small compared to the increased memory communications in GPUs. In the case of T5-small, we even observed a 35% increase in complexity. With better, more memory-efficient MoE implementation, we might improve the performance of MoEs. But as of now, MoE models remains competitive only in large models regimes.

5.2 Qualitative Analysis

To perform subjective evaluation of our MoE*fied* model, we look at couple examples from the SST2 dataset where our MoEfied model has made a mistake, but the baseline model succeeds. There are 23 such examples, and 15 of them are positive sentiments misclassified as negative, and 8 of them are negative sentiments misclassified as positive. We list a couple examples below:

sst2 sentence: the primitive force of this film seems to bubble up from the vast collective memory of the combatants .
Label: positive
Prediction: negative

sst2 sentence: there is nothing outstanding about this film , but it is good enough and will likely be appreciated most by sailors and folks who know their way around a submarine .
Label: positive
Pred: negative

sst2 sentence: sticky sweet sentimentality , clumsy plotting and a rosily myopic view of life in the wwii-era mississippi delta undermine this adaptation .
Label: negative
Pred: positive

Notice that many of the examples where positive sentiment is misidentified as negative have some degrees of ambiguity. In particular, the sentence "there is nothing outstanding about this film , but it is good enough ..." can be subjectively judged as either positive or negative. We believe these examples are instances where the model needs to learn nuanced understanding of the dataset curator's classification criteria from the training set.

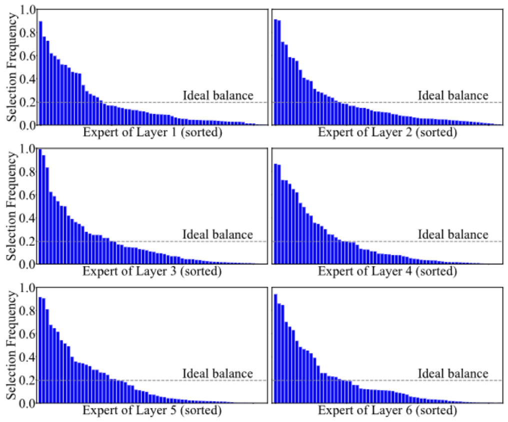


Figure 2: Distribution of expert activations. Figure from Zhang et al. (2022)

On the other hand, the negative examples showcases another type of failures. These examples consists of more difficult vocabulary with more artistic, poetic prose. These instances represent difficulty problems that require more reasoning capacity and thorough understanding from the model, which is unfortunately reduced due to sparse expert activation.

We hypothesize these examples demonstrates a deficiency in our MoE-based system in tackling problems in the long-tail. In Figure 2, we see that our MoEfied models tend to favor a couple commonly used experts, with a pretty steep drop off after 20 or so experts. These uncommonly used experts might represent deeper knowledge that's not needed in simple tasks, but are required for more complex tasks in our examples.

6 Conclusion

In this project, we develop a pipeline for converting a large pretrained model into a more resource-efficient MoE model and efficiently recover most of the degradation in performance emanating from the MoE*fication* process. We show that MoE*fication* can achieve 30% inference speed-up on GPU for large models, while performance on several classification benchmarks is nearly unchanged. Additionally, we identify several scaling laws for MoE*fication*, showing that the benefits of MoE*fication* increase with model size (but become negative for small models), and that the addition of LoRA has negligible effects on this pattern.

Some limitations of our studies are that we have only demonstrated the accuracy on T5-base, which is a 220M parameter encoder-decoder model. Our tests on inference speed of T5-3B, however, show that efficiency increases with model size, so in future works, we would like to check if our approach scales well to billion-parameter models, as well as to decoder-only architectures.

Furthermore, while our approach can recover the performance degradation from having a suboptimal routing, we still observe a small gap between the baseline model and the best MoEfied model. Whether this performance difference is fundamental to an MoE system, or it can be further ameliorated with a better expert partitioning method is still an open question.

References

Armen Aghajanyan, Lili Yu, Alexis Conneau, Wei-Ning Hsu, Karen Hambardzumyan, Susan Zhang, Stephen Roller, Naman Goyal, Omer Levy, and Luke Zettlemoyer. 2023. Scaling laws for generative mixed-modal language models.

Haoli Bai, Wei Zhang, Lu Hou, Lifeng Shang, Jin Jin, Xin Jiang, Qun Liu, Michael Lyu, and Irwin King. 2021. BinaryBERT: Pushing the limit of BERT quantization. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4334–4348, Online. Association for Computational Linguistics.

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

Youshua Bengio. 2013. Deep learning of representations: Looking forward. In *Proceedings of SLSP*, pages 1–37, Online.

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2021. Editing factual knowledge in language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, page 6491–6506, Online. Association for Computational Linguistics.

Jiaao Chen, Aston Zhang, Xingjian Shi, Mu Li, Alex Smola, and Diyi Yang. 2023. Parameter-efficient fine-tuning design spaces. *arXiv preprint arXiv:2301.01821v1*.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamás Sarlós, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy J. Colwell, and Adrian Weller. 2020. Rethinking attention with performers. *CoRR*, abs/2009.14794.

Tejalal Choudhary, Vipul Misrha, Anurag Goswami, and Jagannathan Sarangapani. 2020. A comprehensive survey on model compression and acceleration. volume 53, pages 5113–5155, Online. Artificial Intelligence Review.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D Manning. 2019. What does bert look at? an analysis of bert's attention. *arXiv preprint arXiv:1906.04341*.

Raffel Colin, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. Google Research.

Damai Dai, Li Dong, Yaru Hao, Zhigand Sui, Baobao Chang, and Furu Wei. 2021. Knowledge neurons in pretrained transformers. Online. ArXiv.

Nan Du, Yanping Huang, Andrew M Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, Barret Zoph, Liam Fedus, Maarten P Bosma, Zongwei Zhou, Tao Wang, Emma Wang, Kellie Webster, Marie Pellat, Kevin Robinson, Kathleen Meier-Hellstern, Toju Duke, Lucas Dixon, Kun Zhang, Quoc Le, Yonghui Wu, Zhifeng Chen, and Claire Cui. 2022. GLaM: Efficient scaling of language models with mixture-of-experts. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 5547–5569. PMLR.

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

Mor Geva, Avi Caciularu, Kevin Ro Wang, and Yoav Goldberg. 2022. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. pages 8440–8451, Online. Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing.

Tae Jun Ham, Sung Jun Jung, Seonghak Kim, Young H. Oh, Yeonhong Park, Yoonho Song, Jung-Hun Park, Sanghee Lee, Kyoung Park, Jae W. Lee, and Deog-Kyoon Jeong. 2020. Accelerating attention mechanisms in neural networks with approximation. In *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 328–341.

Xu Han, Zhengyan Zhang, Ning Ding, Yuxian Gu, Xiao Liu, Yuqi Huo, Jiezhong Qiu, Yuan Yao, Ao Zhang, Liang Zhang, Wentao Han, Minlie Huang, Qin Jin, Yanyan Lan, Yang Liu, Zhiyuan Liu, Zhiwu Lu, Xipeng Qiu, Ruihua Song, Jie Tang, Ji-Rong Wen, Jinhui Yuan, Wayne Xin Zhao, and Jun Zhu. 2021. Pre-trained models: Past, present and future.

Hussein Hazimeh, Zhe Zhao, Aakanksha Chowdhery, Maheswaran Sathiamoorthy, Yihua Chen, Rahul Mazumder, Lichan Hong, and Ed Chi. 2021. Dselect-k: Differentiable selection in the mixture of experts with applications to multi-task learning. In *Advances in Neural Information Processing Systems*, volume 34, pages 29335–29347. Curran Associates, Inc.

Jiaao He, Jiezhong Qiu, Aohan Zeng, Zhilin Yang, Jidong Zhai, and Jie Tang. 2021. Fastmoe: A fast mixture-of-expert training system. *arXiv preprint arXiv:2103.13262*.

Lu Hou, Zhiqi Huang, Lifeng Shang, Xin Jiang, Xiao Chen, and Qun Liu. 2020. Dynabert: Dynamic bert with adaptive width and depth. In *Advances in Neural Information Processing Systems*, volume 33, pages 9782–9793. Curran Associates, Inc.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799. PMLR.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. 1991. Adaptive mixtures of local experts. In *Neural Computation*, volume 3, pages 79–87, Online.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. TinyBERT: Distilling BERT for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4163–4174, Online. Association for Computational Linguistics.

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

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. In *International Conference on Learning Representations*.

Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. 2019. Revealing the dark secrets of BERT. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4365–4374, Hong Kong, China. Association for Computational Linguistics.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. RACE: Large-scale ReAding comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794, Copenhagen, Denmark. Association for Computational Linguistics.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Mike Lewis, Shruti Bhosale, Tim Dettmers, Naman Goyal, and Luke Zettlemoyer. 2021. BASE layers: Simplifying training of large, sparse models. *CoRR*, abs/2103.16716.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Tao Lin, Sebastian U. Stich, Luis Barba, Daniil Dmitriev, and Martin Jaggi. 2020. Dynamic model pruning with feedback.

Christopher D Manning, Kevin Clark, John Hewitt, Urvashi Khandelwal, and Omer Levy. 2020. Emergent linguistic structure in artificial neural networks trained by self-supervision. *Proceedings of the National Academy of Sciences*, 117(48):30046–30054.

Saeed Masoudnia and Reza Ebrahimpour. 2014. Mixture of experts: a literature survey. volume 42, pages 275–293, Online. Artificial Intelligence Review.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in gpt. In *Advances in Neural Information Processing Systems*, volume 35, pages 17359–17372, Virtual/Online.

Antonio Polino, Razvan Pascanu, and Dan Alistarh. 2018. Model compression via distillation and quantization. *CoRR*, abs/1802.05668.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. 2023. Efficiently scaling transformer inference. *Proceedings of Machine Learning and Systems*, 5.

Sahana Ramnath, Preksha Nema, Deep Sahni, and Mitesh M. Khapra. 2020. Towards interpreting BERT for reading comprehension based QA. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3236–3242, Online. Association for Computational Linguistics.

Babak Rokh, Ali Azarpeyvand, and Alireza Khanteymoori. 2023. A comprehensive survey on model quantization for deep neural networks in image classification. *ACM Transactions on Intelligent Systems and Technology*, 14(6):1–50.

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

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108.

Noam 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. *CoRR*, abs/1701.06538.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Xavier Suau, Luca Zappella, and Nicholas Apostoloff. 2020. Finding experts in transformer models. Online. ArXiv.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2022. Efficient transformers: A survey. *ACM Comput. Surv.*, 55(6).

Jesse Vig and Yonatan Belinkov. 2019. Analyzing the structure of attention in a transformer language model. *CoRR*, abs/1906.04284.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, Florence, Italy. Association for Computational Linguistics.

Eric Wallace, Jens Tuyls, Junlin Wang, Sanjay Subramanian, Matt Gardner, and Sameer Singh. 2019. AllenNLP interpret: A framework for explaining predictions of NLP models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations*, pages 7–12, Hong Kong, China. Association for Computational Linguistics.

Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-attention with linear complexity. *CoRR*, abs/2006.04768.

Wenxuan Wang and Zhaopeng Tu. 2020. Rethinking the value of transformer components. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6019–6029, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics.

Xun Wu, Shaohan Huang, and Furu Wei. 2024. MoLE: Mixture of loRA experts. In *The Twelfth International Conference on Learning Representations*.

Wenhan Xia, Hongxu Yin, Xiaoliang Dai, and Niraj K. Jha. 2022. Fully dynamic inference with deep neural networks. *IEEE Transactions on Emerging Topics in Computing*, 10(2):962–972.

Ji Xin, Raphael Tang, Jaejun Lee, Yaoliang Yu, and Jimmy Lin. 2020. DeeBERT: Dynamic early exiting for accelerating BERT inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2246–2251, Online. Association for Computational Linguistics.

Elad Ben Zaken, Shauli Ravfogel, and Yoav Goldberg. 2021. Bitfit: Simple parameter-efficient fine-tuning for transformer-based masked language-models. *CoRR*, abs/2106.10199.

Zhengyan Zhang, Yankai Lin, Zhiyuan Liu, Pen Li, Maosong Sun, and Jie Zhou. 2022. Moefication: Transformer feed-forward layers are mixtures of experts. Online. ArXiv.

Zhengyan Zhang, Fanchao Qi, Zhiyuan Liu, Qun Liu, and Maosong Sun. 2021. Know what you don't need: Single-shot meta-pruning for attention heads. *AI Open*, 2:36–42.

Yanqi Zhou, Lei Tao, Hanxiao Liu, Nan Du, Yangping Huang, Vincent Zhao, Andrew M Dai, Zhifeng Chen, Quoc V Le, and James Laudon. 2015. Mixture-of-experts with expert choice routing. volume 35, pages 7103–7114, Online. Advances in Neural Information Processing Systems.

Michael Zhu and Suyog Gupta. 2017. To prune, or not to prune: exploring the efficacy of pruning for model compression.

A Appendix

A.1 Input Preprocessing and Targets

A.1.1 SST2

SST2 is already very close to a text-input text-output structure, so the only input preprocessing needed is to prompt T5 with the desired output type by prepending "sst2 sentence". T5 has already been fine-tuned on SST2 using this format so did not require additional fine-tuning. Below is an example of the preprocessing:

Original:
Example Sentence: "that loves its characters and communicates something rather beautiful about human nature"
Example label: 1

Preprocessed:
Model Input: "sst2 sentence: that loves its characters and communicates something rather beautiful about human nature"
Model Target: "positive"

A.1.2 MNLI

T5 was also pretrained on MNLI, so we follow T5's preprocessing steps:

Original:
Example Premise: "Conceptually cream skimming has two basic dimensions - product and geography."
Example Hypothesis: "Product and geography are what make cream skimming work."
Example Attribution: "Neutral"

Preprocessed:
Model Input: "mnli hypothesis: Conceptually cream skimming has two basic dimensions - product and geography. premise: Conceptually cream skimming has two basic dimensions - product and geography."
Model Target: "neutral"

A.1.3 RACE

RACE was *not* included in T5's training mixture, so we performed custom preprocessing and fine-tuning:

Original:
Example Article: "There is not enough oil in the world now. As time goes by, it becomes less and less, so what are we going to do when it runs ou..."
Example Question: "According to the passage, which of the following statements is TRUE?"
Example Options: ["There is more petroleum than we can use now.", "Trees are needed for some other things besides making gas.", "We got electricity from ocean tides in the old days.", "Gas wasn't used to run cars in the Second World War."]
Example Answer: "B"

Preprocessed:
Model Input: "question: According to the passage, which of the following statements is TRUE? options: A: There is more petroleum than we can use now. B: Trees are needed for some other things besides making gas. C: We got electricity from ocean tides in the old days. D: Gas wasn't used to run cars in the Second World War. article: There is not enough oil in the world now. As time goes by, it becomes less and less, so what are we going to do when it runs ou..."
Model Target: "B"

We put the question and options before the article to avoid information loss due to truncation (which affects <1% of examples) during tokenization.