	Wikitext Generation				zsRE Question-Answering			
	GPT-Neo (2.7B)		GPT-J (6B)		T5-XL (2.8B)		T5-XXL (11B)	
Editor	ES ↑	ppl. DD ↓	ES ↑	ppl. DD ↓	ES ↑	acc. DD ↓	ES ↑	acc. DD ↓
FT	0.55	0.195	0.80	0.125	0.58	**<0.001**	0.87	**<0.001**
FT+KL	0.40	**0.026**	0.36	0.109	0.55	**<0.001**	0.85	**<0.001**
KE	0.00	0.137	0.01	0.068	0.03	**<0.001**	0.04	**<0.001**
MEND	**0.81**	0.057	**0.88**	**0.031**	**0.88**	0.001	**0.89**	**<0.001**

Table 3: Editing very large pre-trained models on our Wikitext generative editing problem and the zsRE question-answering editing problem used by De Cao et al. (2021). MEND consistently produces more effective edits (higher success, lower drawdown) than existing editors. **ES** is the edit success rate, while **ppl. DD** and **acc. DD** are the model drawdown in units of perplexity increase or accuracy decrease, respectively. Due to ENN's memory requirements, we were unable to run the algorithm for models of this size. The low drawdown for all T5 models may occur because the T5 models (pre-trained on mask filling and finetuned for question-answering by Roberts et al. (2020)) might not be fully converged on the question-answering problem. Edits may therefore effectively serve as task specification, further fine-tuning the model on question-answering. **FT** refers to fine-tuning; **FT+KL** is fine-tuning with a KL-div. penalty between the original and fine-tuned model.

from Wikitext-103 and y_e is a 10-token sample from a pre-trained distilGPT-2 model.[4] x_{loc} is chosen depending on the pre-trained model: for models pre-trained on Wikitext, x_{loc} is sampled from Wikitext-103 (independently from x_e). For GPT-Neo/J, we sample x_{loc} from OpenWebText (OWT; (Gokaslan and Cohen, 2019)) to better match the model's original training data. The equivalence neighborhood in this setting is $N(x_e, y_e) = \{(x_e^k, y_e)\}$, where x_e^k is formed by removing a prefix of up to $\frac{|x_e|}{2}$ tokens from the beginning of x_e, where $|x_e|$ is the length of x_e in tokens.

Comparison of model editors. We compare MEND with several other model editors, including two fine-tuning-based algorithms (which do not train any model editor at all) and two learned model editors. The **fine-tune (FT)** algorithm fine-tunes on the edit example (x_e, y_e) until the label is assigned the highest likelihood (using greedy decoding for sequence models). The 'oracle' **fine-tune + KL (FT+KL)** algorithm has access to the training set at test time and adds L_{loc} (Eq. 4b) to the test-time fine-tuning objective (which is typically only computable during model editor training). Similarly to De Cao et al. (2021), we limit each of these algorithms to 100 fine-tuning steps. Additionally, we compare with two *learned* model editors: a re-implementation of Editable Neural Networks (ENN; Sinitsin et al., 2020) when possible (due to high memory usage) and KnowledgeEditor (KE; De Cao et al., 2021). We use **identical hyperparameters** for MEND across all models and datasets. For BART and T5 models, we edit the MLP weight matrices in the last 2 transformer blocks of the encoder and decoder; for other models, we edit the MLP weights in the last 3 transformer blocks. Appendix G explores a simple caching-based model editor that stores model edits in memory.

Metrics. Our experiments measure the reliability and generality of a model editor using **edit success (ES)** (Eq. 1). To assess locality, we use **drawdown (DD)**, which is defined as the performance degradation of the edited model on the rest of the dataset, measured as either the edited model's perplexity increase or accuracy decrease compared to the base model, depending on the problem.

5.1 Editing very large transformer models

We first consider the problem of editing some of the largest publicly-available Transformer models. We use GPT-Neo (2.7B parameters; Black et al., 2021) and GPT-J (6B parameters; Wang and Komatsuzaki, 2021), several times larger than GPT-2 (Radford et al., 2019), and the largest two T5 models, T5-XL (2.8B parameters) and T5-XXL (11B parameters) fine-tuned on NQ (Roberts et al., 2020). Table 3 shows the results; MEND provides the most successful edits across tasks. Fine-tuning achieves lower edit success on the Wikitext task and exhibits a much larger perplexity increase than MEND. On the question-answering edit task, fine-tuning shows similarly reduced edit success, struggling to generalize to some rephrasings of the edit input. The KL-constrained baseline reduces the perplexity drawdown for GPT-Neo and GPT-J, but at the cost of edit success. KE is ineffective at this scale, generally failing to provide successful edits. For these experiments, we use OWT and NQ to measure drawdown for generation and question-answering, respectively, as they are more representative of the data used to train the base models.

[4]The base model's greedy 10-token prediction agrees with these edit targets for <1% of examples.

	FEVER Fact-Checking		zsRE Question-Answering		Wikitext Generation	
	BERT-base (110M)		BART-base (139M)		distilGPT-2 (82M)	
Editor	ES ↑	acc. DD ↓	ES ↑	acc. DD ↓	ES ↑	ppl. DD ↓
FT	0.76	**<0.001**	0.96	**<0.001**	0.29	0.938
FT+KL	0.64	**<0.001**	0.89	**<0.001**	0.17	**0.059**
ENN	**0.99**	0.003	**0.99**	**<0.001**	**0.93**	0.094
KE	0.95	0.004	**0.98**	**<0.001**	0.25	0.595
MEND	**>0.99**	**<0.001**	**0.98**	0.002	0.86	0.225

Table 4: Small-scale model editing with various model editors on three editing problems. ENN and MEND show the most consistently good performance, with ENN exceeding MEND's performance on the Wikitext problem. MEND's primary advantages are its consistent performance from 100M to 10B parameter models and the fact that it does not modify the pre-edit model (unlike ENN). The pre-trained models and editing data for the FEVER fact-checking and zsRE question-answering problems are used from the checkpoints and data released by De Cao et al. (2021); for generation, we use distilGPT-2 fine-tuned on Wikitext2 (Ma, 2021).

5.2 Smaller Scale Editing

We conduct an additional experiment editing the BERT-base and BART-base models fine-tuned by De Cao et al. (2021) on the FEVER fact-checking and zsRE question-answering tasks, respectively, and our Wikitext editing task, editing a smaller distilGPT-2 model (Wolf et al., 2019) fine-tuned on Wikitext2 (Ma, 2021). These models are 1–2 orders of magnitude smaller than those in Section 5.1. Results are presented in Table 4. At small scale where computational requirements are not a concern, ENN is competitive with MEND, providing the best performance on the Wikitext problem. Fine-tuning overfits even more severely than with larger models, showing lower edit success (overfitting to the edit example) and higher drawdown (degrading the model more seriously). One difficulty of using ENN is that the pre-trained model itself must be fine-tuned to 'provide' editability, potentially changing the model's predictions even *before* an edit has been applied. Unlike the large-scale experiments, drawdown is computed using samples from the same datasets as edit inputs, again in order to best match the data distribution the base models were fine-tuned on. See Appendix G for additional comparisons with the caching-based editor, which shows strong performance for zsRE and FEVER, but generally fails for Wikitext, as well as a more difficult version of the zsRE problem for which MEND still produces meaningful edits.

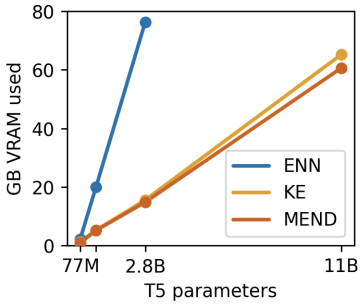

Figure 3: GPU VRAM consumption during training. ENN's memory usage[5] is prohibitively high for very large models, while MEND and KE can be trained on a single GPU. Figure 4 shows similar chart for GPT models.

5.3 Batched Editing

Table 5 compares MEND with ENN (the strongest comparison method) in a more realistic setting when multiple simultaneous zsRE QA model edits are needed; MEND consistently provides significantly more effective edits in the multi-edit setting. Both algorithms are trained and evaluated on applying k simultaneous edits, with $k \in \{1, 5, 25, 75, 125\}$. MEND applies simultaneous edits by simply summing the parameter edit computed separately for each edit example. MEND applies 25 edits in a single model update with 96% edit success and less than 1% accuracy degradation (35% edit success for ENN), and successfully applies 67% of edits when applying 125 edits at once (11% success for ENN, although ENN's accuracy drawdown is slightly lower).

	Edit Success ↑		Acc. Drawdown ↓	
Edits	ENN	MEND	ENN	MEND
1	0.99	0.98	<0.001	0.002
5	0.94	0.97	0.007	0.005
25	0.35	0.89	0.005	0.011
75	0.16	0.78	0.005	0.011
125	0.11	0.67	0.006	0.012

Table 5: Batched edits with MEND and ENN on zsRE QA using the BART-base pre-trained model from De Cao et al. (2021). When applying multiple edits at once, MEND is far more effective than ENN.

[5]We report the memory usage of our re-implementation of ENN (Sinitsin et al., 2020). Techniques like gradient checkpointing can reduce memory consumption, but an optimized ENN implementation is not available.

		Wikitext Generation		zsRE Question-Answering	
		distilGPT-2 (82M)		BART-base (139M)	
MEND Variant	Editor Parameters	ES ↑	ppl. DD ↓	ES ↑	acc. DD ↓
No sharing	$O((m+n)^2 N)$	**0.86**	**0.195**	**>0.99**	0.001
No norm.	$O((m+n)^2)$	0.02	0.370	0.97	**<0.001**
No ID init.	$O((m+n)^2)$	0.27	0.898	0.94	**<0.001**
Only u_ℓ	$O(m^2)$	0.63	0.559	0.98	0.002
Only $\delta_{\ell+1}$	$O(n^2)$	0.80	0.445	**0.99**	0.001
Only smaller	$O(\min(m,n)^2)$	0.80	0.593	0.98	0.002
MEND	$O((m+n)^2)$	**0.86**	0.225	**>0.99**	0.001

Table 6: Ablating various properties of MEND on the Wikitext and zsRE question-answering editing problems. $m = \dim(u_\ell)$, $n = \dim(\delta_{\ell+1})$, and N is the number of layers being edited. Removing MEND's identity initialization and input normalization noticeably lowers editing performance, and relaxations of MEND, particularly the 'only smaller' variant that only outputs pseudoactivations *or* pseudodeltas, whichever is smaller, show competitive performance, which bodes well for scaling MEND to 100 billion+ parameter models.

5.4 Ablations & MEND Variants

Table 6 shows ablations of MEND's parameter sharing, identity initialization, and input normalization as well as three variants of MEND that reduce total parameters: only computing pseudoactivations u_ℓ, only pseudodeltas $\delta_{\ell+1}$, or only whichever of u_ℓ or $\delta_{\ell+1}$ is lower-dimensional (layer-dependent for non-square weights). 'No ID init.' replaces zero initialization with Xavier/Glorot initialization (Glorot and Bengio, 2010). Removing *either* input normalization or identity initialization significantly reduces edit effectiveness (and increases training time $\sim$10x). Sharing parameters across model editor networks incurs relatively little performance cost, and editing *only* the smaller of the pseudoactivations and pseudodeltas, the most most lightweight version of MEND, still produces effective edits, suggesting that MEND could scale to even much larger models for which $m+n$ approaches 10^5 (Brown et al., 2020) but $\min(m,n)$ remains close to 10^4. Appendix E shows an additional ablation editing attention matrices, rather than MLP weights, finding that editing MLP weights is consistently more effective for large models.

6 Discussion

Conclusion. We have presented an efficient approach to editing very large (10 billion+ parameter) neural networks, which we call Model Editor Networks with Gradient Decomposition or MEND. We showed that MEND is the only method that successfully edits the largest publicly-available Transformer models from the GPT and T5 model families. To do so, MEND treats the model editing problem itself as a learning problem, using a relatively small edit dataset to learn model editor networks that can correct model errors using only a single input-output pair. MEND leverages the fact that gradients with respect to the fully-connected layers in neural networks are rank-1, enabling a parameter-efficient architecture that represents this gradient transform.

Limitations & Future Work. A limitation of existing model editors (including MEND) is the approach to enforcing locality of edits. The failure mode of over-generalization (bottom of Table 2) shows that locality examples (i.e., negative examples) are not challenging enough to prevent the model from sometimes changing its output for distinct but related inputs. Alternative locality losses or harder negative mining may help address this problem. Further, existing language-based editing datasets use backtranslation to evaluate edit generality (and our Wikitext dataset uses a truncation heuristic). Such equivalence neighborhoods do not assess a model's ability to use the knowledge in an edit example to correctly answer questions about other topics whose answer is *implied* by the content of the edit example (e.g., for *Who is the UK PM? Boris Johnson*, does the edited model correctly answer *Is Boris Johnson a private citizen?*). Counterfactual data augmentation (Kaushik et al., 2020) may be useful for constructing richer evaluation cases for edit generality. Future work might also apply MEND to other types of edits, such as reducing the frequency of toxic generations after observing toxic outputs, relabeling entire classes of images from one example, or adjusting a robot's control policy to avoid particular actions, as MEND is not limited to editing transformer models. Finally, MEND's gradient decomposition is not in principle limited to the model editing problem, and it might enable efficient new gradient-based meta-learning algorithms.