Drop Dropout on Single-Epoch Language Model Pretraining

Houjun Liu*, **John Bauer** and **Christopher D. Manning**
Stanford University
*houjun@stanford.edu

Abstract

Originally, dropout was seen as a breakthrough regularization technique that reduced overfitting and improved performance in almost all applications of deep learning by reducing overfitting. Yet, single-epoch pretraining tasks common to modern LLMs yield minimal overfitting, leading to dropout not being used for large LLMs. Nevertheless, no thorough empirical investigation has been done on the role of dropout in LM pretraining. Through experiments in single-epoch pretraining of both masked (BERT) and autoregressive (Pythia 160M and 1.4B) LMs with varying levels of dropout, we find that downstream performance in language modeling, morpho-syntax (BLiMP), question answering (SQuAD), and natural-language inference (MNLI) improves when dropout is not applied during pretraining. We additionally find that the recently-introduced "early dropout" also degrades performance over applying no dropout at all. We further investigate the models' editability, and find that models trained without dropout are more successful in gradient-based model editing (MEND) and equivalent in representation-based model editing (ReFT). Therefore, we advocate to **drop dropout** during single-epoch pretraining.

1 Introduction

Dropout (Hinton et al., 2012; Srivastava et al., 2014) is the method of randomly removing a certain percentage of features during each training pass. For the decade after the introduction of AlexNet (Krizhevsky et al., 2012), the use of dropout became standard as a simple, highly effective regularization mechanism for very deep neural networks. Dropout helps create more robust feature representations, in particular, reducing feature co-adaptations (Hinton et al., 2012), enabling the network to learn to make independent predictions from features and leading to more robust networks.

Dropout was originally introduced at a large rate of $p = 0.5$ (Hinton et al., 2012), but the dropout rate used in NLP is steadily reducing in the decade that follows. The original transformer architecture (Vaswani et al., 2017) applies dropout $p = 0.3$ at each of its network layers. BERT (Devlin et al., 2019), GPT-2 (Radford et al., 2019), and T5 (Raffel et al., 2020) all used dropout $p = 0.1$. Recent open language models such as LLaMA (Touvron et al., 2023) do not report any dropout use.

Alternate uses of dropout have emerged beyond regularization. Liu et al. (2023) highlights a novel use of "early dropout" as a stabilisation approach to reduce early underfitting. The authors found that applying dropout early reduces downstream underfitting, but this effect is reduced when dropout is applied throughout training.

In our study, we examine the use of dropout, including early dropout, within the context of pretraining by investigating the effects of both standard and early dropout in pretrained language models. We pretrain both masked and decoder language models (MLMs and LMs), in particular, BERT-base (Devlin et al., 2019) and Pythia 160M and 1.4B (Biderman et al., 2023), with varying levels of dropout at $p = 0.0$, $p = 0.1$, and $p = 0.3$. Additionally, we apply early dropout (as in Liu et al. 2023) during the first 35% of training; we then measure the downstream capabilities of these models at varying checkpoints. We measure morpho-syntactic understanding of linguistic minimal pairs (average BLiMP score) for decoder LMs and evaluate question answering for masked LMs. For all architectures, we additionally measure LM loss. We find that the complete removal of dropout (including early dropout) during pretraining yielded the most capable models across all measures.

Recent investigations of LMs also show that their performance varies based on the degree of consistency with which they process inputs (Elazar et al., 2021), pointing to the tension between having mul-

Findings of the Association for Computational Linguistics: ACL 2025, pages 2157–2166

tiple distributed representations which dropout induces (Hinton, 1984) and more localist approaches. We investigate this by measuring the means by which MLMs store factual knowledge through performing interventions on latent embeddings. By editing an MLM's embeddings through MEND (Mitchell et al., 2022), a gradient-based model editing technique, as well as ReFT (Wu et al., 2024), a representation-based model editing technique, we find that models trained without dropout can be more easily edited.

Finally, we discuss and contextualize the implications of this result. We release code and pretrained models.[1]

2 Related Work

Dropout The dropout mechanism (Hinton et al., 2012) has been extensively studied as a means to reduce feature co-adaptation, create ensembles, regularize, and improve gradient alignment (Srivastava et al., 2014; Baldi and Sadowski, 2013; Wager et al., 2013; Gal and Ghahramani, 2016; Liu et al., 2023).

Knowledge Localization We take the view that factual "knowledge" can be stored and elicited from neural networks (Petroni et al., 2019), specifically, in the Multi-Layer Perceptron (MLP) after each layer's attention block (Geva et al., 2021). Methods exist to measure the correctness and consistency (Elazar et al., 2021) of the stored knowledge and to probe for their stored location in terms of MLP activations (Dai et al., 2022) as well as MLP parameters (Csordás et al., 2023).

Factual Editing Methods exist to edit "knowledge" stored in LMs, including tuning a low-rank subspace of the network (Hu et al., 2022), learning a parameter update through a surrogate network (De Cao et al., 2021), projecting tuning gradients into edits (Mitchell et al., 2022), probing orthogonal subspaces of representations (Wu et al., 2024), or causal methods specific to mutating information flow in decoder models (Meng et al., 2022).

3 Approach

We first investigate the effects of dropout on both the pretraining objective and downstream capabilities of masked and decoder language models. To do this, we first pretrain two sets of language models with varying levels of dropout (Section 3.1) and then evaluate their capabilities through varying metrics (Section 3.2). Finally, we perform embedding and editing experiments on the pretrained BERT models to investigate the causal influence of applying dropout in pretraining (Section 3.3).

3.1 Pretraining

We train two sets of models: BERT-base (Devlin et al., 2019) MLMs, and Pythia 160M and 1.4B decoder LMs (Biderman et al., 2023). The BERT models are trained using the masked language modeling objective using the 10-billon-token slice of Huggingface FineWeb (HuggingFaceFW, 2024) following the optimization procedure given in Appendix B; the Pythia models are trained using The Pile Deduplicated (Gao et al., 2020) dataset (to match the original Pythia models) up to 1.5 million steps using the optimization procedure given in Appendix C.

We apply pretraining dropout at $p = 0.0$, $p = 0.1$, and $p = 0.3$ on the attention and MLP blocks. We experiment with both static dropout as well as "early dropout" following Liu et al. (2023) whereby dropout is scheduled on the first 35% of training before being disabled. Details on the early dropout implementation are discussed in Appendix D.

3.2 Capabilities Evaluations

Our primary goal is to investigate how pretraining with dropout on single-epoch training affects the downstream performance of language models. To do this, we perform the following measurements:

(M)LM Loss First, we measure the loss of our models on the pretraining objective on a withheld subset of the pretraining distribution. For decoder-LMs, this is simple cross-entropy loss. For MLMs, this is cross-entropy loss on the *masked* training objective (i.e., the 15% masked cloze task described in Appendix B). This simple measure has been widely reported as correlating with fluent natural language generation matching human-like distributions (Goodkind and Bicknell, 2018; Wilcox et al., 2020). Hence, it serves as a simple baseline for (M)LM capabilities.

Morpho-syntactic Phenomena For the decoder LMs (i.e., models that can be used in the conventional sense as a sequence "language model"), we evaluate linguistic knowledge using the BLiMP dataset (Warstadt et al., 2020). BLiMP is an evaluation dataset of 67 different grammatical phenomena and consists of linguistic minimal pairs, exactly one

[1] https://github.com/Jemoka/dropfree

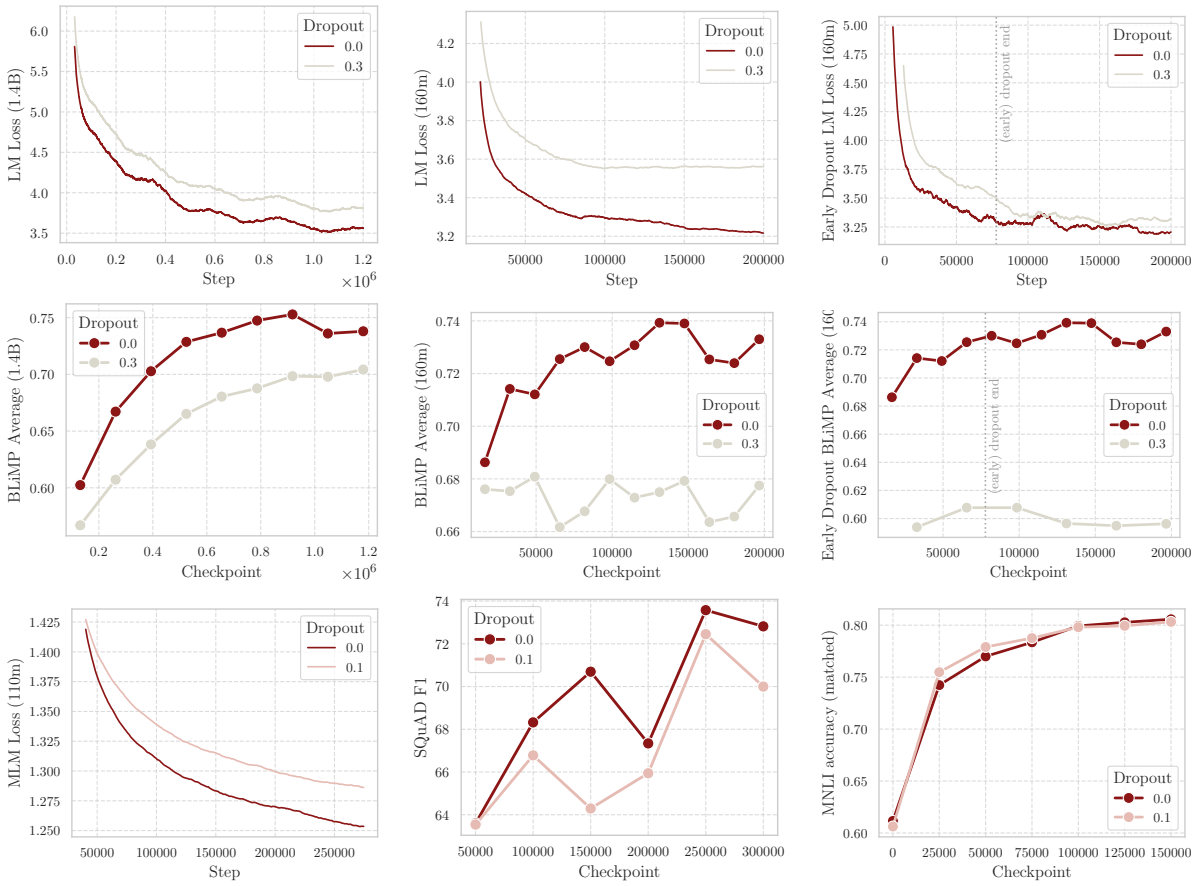

Figure 1: top: language modeling loss with dropout $p = 0.0$, $p = 0.3$, and early dropout for decoder LMs; middle: mean BLiMP scores for these models; bottom: masked language modeling loss with dropout $p = 0.0$, $p = 0.1$, SQuAD F1 (answerable) dev-set scores, and matched MNLI scores.

of which is grammatically acceptable. We evaluate the emergence of morphology and syntax in the pretrained LMs by assessing whether the models assign a higher probability to the acceptable sequence than to the unacceptable one.

Fine Tuning with SQuAD The previous two measures evaluate the pretrained LM on various tasks directly relating to the task distribution. However, LM capabilities can be unlocked through fine-tuning (Wei et al., 2022). To investigate this, we fine-tune our MLMs on the SQuAD V2 dataset (Rajpurkar et al., 2018) following standard procedures described in detail in Appendix G (in particular, *with dropout enabled* tuning time regardless of whether it is on in pretraining, since we fine-tune SQuAD for more than one epoch). We report the F1 score obtained by the fine-tuned model.

Fine Tuning with MNLI In addition to the SQuAD results above, we further investigate the fine-tuned capability of our MLMs on the MultiNLI (MNLI) dataset (Williams et al., 2018) following procedures detailed in Appendix H. As with SQuAD, we enable dropout in the model at a rate of 0.15 regardless of whether or not it is enabled for pretraining. We report the overall 3-class classification accuracy ("contradiction", "neutral", and "entailment") obtained by the fine-tuned model.

3.3 Causal and Embedding Analysis

To gain further insight into the causal influence of dropout on the language models' pretraining process, we conduct Causal embedding-level analysis on the pretrained MLMs. One lens with which to approach this investigation is to study the way LM embeddings store "knowledge": recent literature highlights that the storage of factual "knowledge" can be treated as a key-value lookup in the post-attention MLP (Geva et al., 2021) subject to input features; hence if dropout reduces feature co-adaptations (Hinton et al., 2012), we hypothesize that dropout is likely to reduce the consistency with which knowledge is stored and elicited. Appendix A provides detail on this hypothesis.

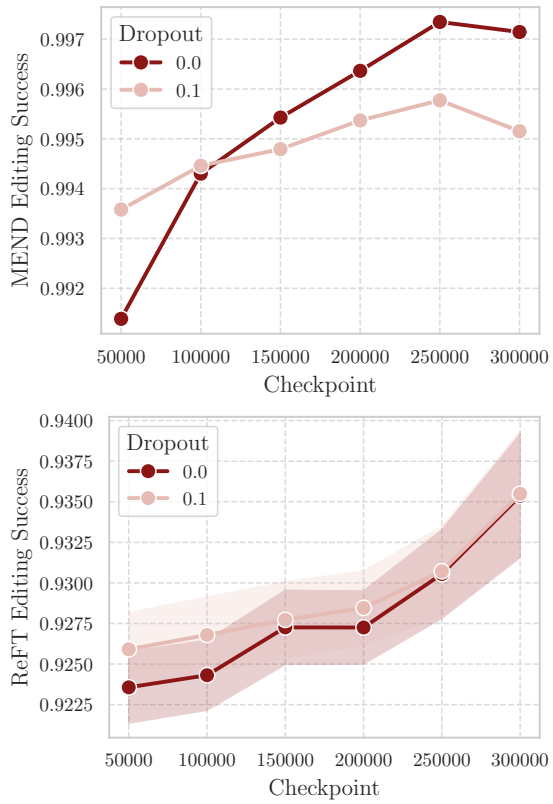

Figure 2: top: MEND edit success rate for MLM at varying levels of dropout ($(p < 0.0001)$ across 5 seeds and edited concepts; margin of error are within floating point differences); bottom: ReFT edit success rate for MLM at varying levels of dropout (no significant difference across 5 seeds and edited concepts).

Formalism In this work, we define "knowledge" as tuples (a, r, b) ("subject-relation-object"); then, if a model M has "stored" particular knowledge, we can find some mapping f that learns $f(M)(a, r) = b$ but learning $f(a, r) = b$ without M may be difficult.

We follow the definition given in (Elazar et al., 2021) to impose additional constraints for human languages. We define: (1) f as the cloze (mask-fill) task on the set of masked phrases (e.g. `[B] is the [R] of [A]`) which are quasi-paraphrases (Bhagat and Hovy, 2013) of each other; and (2) M to be a masked-language model, in particular, a BERT.

Knowledge in BERT-Sized Models One measurable effect of consistent and localized embedding of knowledge is the increased ease of model editing. If "knowledge" is more localized, then one needs to edit a smaller area of the MLP to corrupt or change it.

Therefore, we measure the effects that pretraining with and without dropout has on the editability of downstream models. We define "editing" here as changing a pattern (i.e. (a, r, b) to (a, r, b') with $b \neq b'$) under all choices H. We obtain these patterns from the Pararel (Elazar et al., 2021) dataset, which consists of varying syntactic frames for expressing the same relational concept. We randomly rearrange objects within each relation category and edit the model to generate the newly paired object for each relation. We then evaluate the model's prediction test accuracy on masked tokens with a balanced mix of permuted and unrelated relations.

We train two model editing techniques: Representation Fine Tuning (ReFT) (Wu et al., 2024) and MEND (Mitchell et al., 2022). Further details of these approaches are given in Appendix F and Appendix E respectively.

4 Results

Removing dropout makes more capable models Across all evaluations of capabilities described in Section 3.2, models pretrained with dropout, *including early dropout*, performed worse than those pretrained without dropout. As Fig. 1 shows, the mean LM loss of the models trained without dropout is lower than those trained with any dropout. The models trained without dropout also scored higher on both BLiMP, and SQuAD F1 while performing marginally better on MNLI accuracy. Furthermore, we discover that dropout rate correlates with its effect size on LM performance; we discuss this effect in Appendix J.

...that are slightly more editable As shown in Fig. 2, given sufficient training steps, the model trained without dropout is statistically more successful in editing using MEND and did not have notable editing differences by ReFT. Appendix I discusses this result in detail: we believe applying dropout yielded little difference in ReFT performance because ReFT edits orthogonal subspaces, meaning the *distance* between stored knowledge disturbed by dropout is less important.

5 Conclusion and Discussion

Although it has been standard practice to apply dropout to regularize neural network training, we find—consistent with the recent trend of removing dropout in massive LM pretraining—that dropout in a single-epoch pretraining regime is not necessary and hurts performance. In particular, we find that the models pretrained *without* dropout score consistently more favorably in capability

measures of LM loss, morpho-syntax generalization in BLiMP, and fine-tuned SQuAD. These results hold even with the early-dropout technique (Liu et al., 2023). Furthermore, we note that the models trained without dropout improve in editability after training with sufficient scale with MEND. We hypothesize that this is because dropout non-discriminately limits co-adaptation in input features (Hinton et al., 2012), leading to less localized representations of "knowledge", resulting in multiple independent copies of facts stored. This ultimately makes models less amenable to editing.

Taken together, our conclusions indicate that, with single epoch LM pretraining, one should **drop dropout**.

6 Limitations

Statistical-Theoretical Grounding Though this work provides a strong empirical framework and evidence for removing dropout, it attempts to make no theoretical claims about the expected behavior of dropout. Such an evaluation is difficult naively because the expected value of dropout converges to the identity (Baldi and Sadowski, 2013). However, if taking a lens of dropout as a regularizer in the first order (Wager et al., 2013), this would be a fruitful avenue for future work which is made possible to validate by the empirical evidence here.

Scaling Laws The emergence of overfitting scales with the parameter count of the model. Due to resource limitations (i.e. the fact that this experimental setup requires pretraining multiple seeds of an LM), it is difficult to directly measure how the result scales. However, given the replication from MLMs and decoder LMs ranging from 160M to 1.4B parameters, and the previously reported effects of dropout *strengthen* as models scale (Liu et al., 2023), we believe these results are applicable for pretraining in similar settings.

Activation and ROME-style Probes We elected to use neither integrated-gradient style activation probes nor causal corruption probes in this work due to recent evidence that those probes are fairly input-dependent, not easily localized, and not able to cleanly probe stored "knowledge" in a network (Niu et al., 2024; Hase et al., 2024).

Acknowledgements

We would like to thank our colleagues across Stanford NLP for their insights. In particular, we would like to acknowledge Róbert Csordás, Shikhar Murty, Amelia F. Hardy, Julie Kallini, Liam Kruse, and Ethan Hsu for their valuable time and ideas. We would finally like to thank the anonymous reviewers and editors for their gracious feedback. Houjun Liu was supported in part by the Stanford CURIS program during this work. Christopher D. Manning is a CIFAR fellow.

References

Ashish Agarwal, Clara Wong-Fannjiang, David Sussillo, Katherine Lee, and Orhan Firat. 2018. Hallucinations in neural machine translation. In *ICLR*.

Pierre Baldi and Peter J Sadowski. 2013. Understanding dropout. *Advances in neural information processing systems*, 26.

Rahul Bhagat and Eduard Hovy. 2013. Squibs: What is a paraphrase? *Computational Linguistics*, 39(3):463–472.

Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar Van Der Wal. 2023. Pythia: a suite for analyzing large language models across training and scaling. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Róbert Csordás, Sjoerd van Steenkiste, and Jürgen Schmidhuber. 2023. Are neural nets modular? Inspecting functional modularity through differentiable weight masks. *ICLR*.

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8493–8502, Dublin, Ireland. Association for Computational Linguistics.

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*, pages 6491–6506, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Yanai Elazar, Nora Kassner, Shauli Ravfogel, Abhilasha Ravichander, Eduard Hovy, Hinrich Schütze, and Yoav Goldberg. 2021. Measuring and improving consistency in pretrained language models. *Transactions of the Association for Computational Linguistics*, 9:1012–1031.

Yarin Gal and Zoubin Ghahramani. 2016. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning*.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. 2020. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Adam Goodkind and Klinton Bicknell. 2018. Predictive power of word surprisal for reading times is a linear function of language model quality. In *Proceedings of the 8th Workshop on Cognitive Modeling and Computational Linguistics (CMCL 2018)*, pages 10–18, Salt Lake City, Utah. Association for Computational Linguistics.

Peter Hase, Mohit Bansal, Been Kim, and Asma Ghandeharioun. 2024. Does localization inform editing? Surprising differences in causality-based localization vs. knowledge editing in language models. *Advances in Neural Information Processing Systems*, 36.

G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Geoffrey E Hinton. 1984. Distributed representations. Technical Report CMU-CS-84-157, Carnegie Mellon University.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2022. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

HuggingFaceFW. 2024. fineweb (revision af075be).

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25.

Zhuang Liu, Zhiqiu Xu, Joseph Jin, Zhiqiang Shen, and Trevor Darrell. 2023. Dropout reduces underfitting. In *International Conference on Machine Learning*, pages 22233–22248. PMLR.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. *ICLR*.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in GPT. *Advances in Neural Information Processing Systems*, 35:17359–17372.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. 2022. Fast model editing at scale. *ICLR*.

Jingcheng Niu, Andrew Liu, Zining Zhu, and Gerald Penn. 2024. What does the knowledge neuron thesis have to do with knowledge? *ICLR*.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? 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 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30.

Stefan Wager, Sida Wang, and Percy S Liang. 2013. Dropout training as adaptive regularization. *Advances in neural information processing systems*, 26.

Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R. Bowman. 2020. BLiMP: The benchmark of linguistic minimal pairs for English. *Transactions of the Association for Computational Linguistics*, 8:377–392.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2022. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*.

Ethan G Wilcox, Jon Gauthier, Jennifer Hu, Peng Qian, and Roger P Levy. 2020. On the Predictive Power of Neural Language Models for Human Real-Time Comprehension Behavior. *Proceedings of CogSci*, pages 1707–1713.

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, New Orleans, Louisiana. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Zhengxuan Wu, Aryaman Arora, Zheng Wang, Atticus Geiger, Dan Jurafsky, Christopher D Manning, and Christopher Potts. 2024. Reft: Representation finetuning for language models. *arXiv preprint arXiv:2404.03592*.

A Motivating the Studying of Dropout Through Knowledge Storage

The central claim of Hinton et al. (2012) is that dropout reduces feature co-adaptations. Framing dropout under this lens implies that dropout would modulate the way knowledge is elicited under permutation of the selection of h. The application of dropout would lead the model to learn multiple independent pathways to represent a, r instead of relying on co-adapted features across all existing $h(a, r)$. While this implies our model may be more robust to previously unseen h' that do not share the co-adapted features of the training set, it could also then result in multiple, independent representations of a, r being built under the choice of h_j—leading to possible inconsistency.

Specific to human language modeling, the former lens (dropout leads to more robust models under input sequence permutation) would imply that a model that uses dropout would be more consistent and less likely to hallucinate—a result that has been discussed in literature (Agarwal et al., 2018); the latter lens would imply that a model which uses dropout would build unrelated representations of knowledge that may not be consistent—a result that also has been discussed (Elazar et al., 2021). This work seeks to resolve this tension between two lenses.

B BERT Pretraining Details

We used the Huggingface (Wolf et al., 2020) implementation of BERT-Large (Devlin et al., 2019), and only modified it to disable attention and MLP dropout globally. When MLP dropout is used, $p = 0.1$. Optimization was done *with regularization* using AdamW (Loshchilov and Hutter, 2019) following published parameters:

Parameter	Value
LR	linear warmup 10k, linear decay, peak 1×10^{-4}
Adam β	(0.9, 0.999)
Adam ϵ	1×10^{-6}

Table 1: Details of the Small Scale Model

The model was trained on the officially sampled 10BT slice of Huggingface FineWeb (HuggingFaceFW, 2024), running with fully-sharded data-parallel over 4 GPUs for a joint batch size of $4 \times 96 = 384$. Batching was done sequentially with the Pytorch Data Loader, sequence lengths are capped at 512 tokens. The pretrained tokenizer for BERT-large available on Huggingface was used instead of training a tokenizer from scratch; sequences are wrapped with usual start and end sequence tokens. The model was optimized over 8 days on Nvidia A100 GPUs.

The modeling objective was a span-corruption loss, which uses official sampling rates for incidences of corruption and cloze. Tokens are selected for corruption with 15% chance, and of which 10% are shuffled, 80% are masked, and the rest are re-

turned normally. Sampling is done dynamically with a fixed seed for each run.

C Pythia Pretraining Details

As with Appendix B, we used the Huggingface (Wolf et al., 2020) implementation of the Pythia suite of models (Biderman et al., 2023); the only modifications we performed involves modulating attention and MLP dropout globally as needed for each experiment. Optimization was done *with regularization* using AdamW (Loshchilov and Hutter, 2019) following published parameters:

Parameter	Value
LR	linear warmup 10%, linear decay, peak 6×10^{-4}
Adam β	(0.9, 0.999)
Adam ϵ	1×10^{-6}

Table 2: Details of the Small Scale Model

We trained both the 160M and 1.4B variants of the Pythia models on the first 1.5 million steps (around 1.1BT) of The Pile Deduplicated (Gao et al., 2020) dataset, running with fully-sharded data-parallel over 2 GPUs for a joint batch size of $2 \times 16 = 32$ for the smaller variant and $2 \times 2 = 4$ for the larger variant. Batching was done in local-shuffle random order with a buffer size of $1,000$, and sequence lengths are truncated to the models' maximum length, being 2048. The pretrained tokenizer for each of the models were used instead of training a tokenizer on scratch. The model was optimized over 8 days on Nvidia A100 GPUs.

The model objective used was standard cross-entropy language modeling loss.

D Early Dropout Implementation

We used a simple binary early dropout schedule described in Appendix D of (Liu et al., 2023) in addition to the training parameters of the Pythia suite of models described in Appendix C. In particular, we disabled dropout at 35% of training as a hard cutoff, having trained the model for 77,821 of roughly 220,000 steps of the training corpus. No other training parameters (including optimizer states) are reset after disabling dropout, and training continues as usual.

E MEND Implementation

MEND (Mitchell et al., 2022) is a model editing technique that projects the fine-tuning gradient of a model to weight updates that result in well-localized edits (i.e. edits which do not affect non-edited facts). For target knowledge to store (a, r, b) and unrelated knowledge which we don't want changed (a', r', b'), it does this by learning a function $f : U(a, r) \times \Delta(a, r, b) \rightarrow \nabla W$ that takes the pre-MLP activations $u(a, r)$ and post-MLP fine-tuning gradient for the layer $\delta(a, r, b)$ and produces the appropriate MLP weight difference ∇W. To learn fa, we optimize for an objective which minimizes a joint loss:

$$L = c_{edit} L_{edit} + L_{loc} \tag{1}$$

whereby L_{edit} is the negative log-likelihood of the desired post-edit token $-\log p_{post\ edit}(b|a, r)$ and L_{loc} is the KL-divergence of the posterior distribution of the model against reference $KL(p_{ref}(b'|a', r')||p_{post\ edit}(b'|a', r'))$.

f is an identity-like projection that takes a vector in $V = U(a, r) \times \Delta(a, r, b)$ (the pre-layer activation concatenated with the post-layer fine-tuning gradient) and maps it in the following manner:

$$h(v) = ReLU(U_1 V_1\ norm(v)) + norm(v) \tag{2}$$
$$f(v) = ReLU(U_2 V_2\ h(v)) + h(v) \tag{3}$$

notably, V_j are Xavier initialied while U_j is zero-initialised, making this function f the identity prior to tuning. U_j, V_j is learned.

To learn the edit, we shuffle the targets for each "knowledge" given in the Pararel patterns (Elazar et al., 2021) (i.e. for tuple `(a,r,b)`, we switch `b` to something else sampled in the dataset). We then split the patterns (i.e. h in the formalism given in Section 3.3, the quasi-rephrasing) into a $95\% - 5\%$ train-test split for each knowledge. Edits are learned from all train splits at once, and are tested on all test splits at once to report edit success. Only the accuracy on the target token is reported.

We learn the edits using a learning rate of 5×10^{-5}, with a batch size of 12 and an Adam optimizer. Optimization was done over 3 hours on Nvidia RTX a6000 Ada Generation GPUs. The pretrained models on which the edits were performed did not have dropout on, regardless of whether they are pretrained with dropout.

F ReFT Implementation

ReFT is a model editing technique that intervenes on a certain number of prefix and suffix tokens (i.e. the first and last `n` tokens') embeddings of a model

by perturbing their embeddings in an orthogonal subspace to help localize the edit (i.e. ensuring that there is nothing else that is influenced by the edit). In particular, it learns linear projections weight W, and bias b which is applied to the post-attention hidden projection of the editing layer following:

$$E(h) = h + R^\top(Wh + b - Rh) \quad (4)$$

whereby R is a matrix with enforced orthonormal rows.

We learn a rank-4 edit on layers 4, 6, 8, and 11, intervening on one prefix and suffix tokens out of each sequence. We train the intervention one concept at a time and evaluate on 10% of the held-out patterns.

We train the procedure on one concept at a time: we want to isolate the parameters necessary to only perform the cloze task correctly for that concept; note that this includes normal span corruption on non-concept tokens such as stop words, so properties of general language modeling is not lost.

Edit success is measured by mask token match overall concepts. Optimization was done for one epoch with a learning rate of 5×10^{-4} with the Adam optimizer, with batch sizes of 10 patterns at a time, which took roughly 12 minutes to train for each concept on Nvidia RTX a6000 Ada Generation GPUs. Dropout was disabled during edit and evaluation regardless of whether pretraining used dropout.

G SQuADV2 Fine Tuning

We trained on the train slice of SQuAD v2 (Rajpurkar et al., 2018) for 2 epochs using the Adam optimizer, at a learning rate of 1×10^{-5} with a batch size of 12. Dropout rate was set to 0.1 during training and evaluation regardless of whether pretraining used dropout as is reported in (Devlin et al., 2019). An adapted version of the official evaluation script was used to obtain the dev-slice results reported in this work. Questions and answers are separated by the `[MASK]` token, which was previously used for MLM in the pretraining. A randomly initialized sequence prediction head is added on top of the pretrained network. Non-answers are represented by the model predicting the null span (i.e. starting at `[CLS]` and ending at `[CLS]`—the starts of sequences).

H MNLI Fine Tuning

We fine-tune our trained BERT models on the train slice of the original MNLI (Williams et al., 2018) dataset available [2] on Huggingface for 3 epochs using the adam optimizer, at a fixed learning rate of 2×10^{-5}. Batch size was set to 128, and dropout rate was set to 0.15 regardless of whether pretraining the BERT model used dropout consistent with previous approaches. Premise and hypotheses were separated by the `[SEP]` token, and the final token residual was decoded using a single MLP to form a three-class classification model, which is initially randomly initialized.

I Performance Differences in ReFT vs. MEND

In this work, we found that while MEND had significantly higher editing success in the no dropout case after sufficient training, ReFT's edit success simply converged. We believe the relative higher degree of success in the no-dropout case for MEND is expected: ReFT is formulated to only edit on orthogonal subspaces (Wu et al., 2024), meaning the Euclidean distance between embedding clusters is less important; however, the convergence indicates that the no-dropout model was indeed still able to build equivalently useful embeddings for edits like ReFT.

J Sweeping Dropout Rates

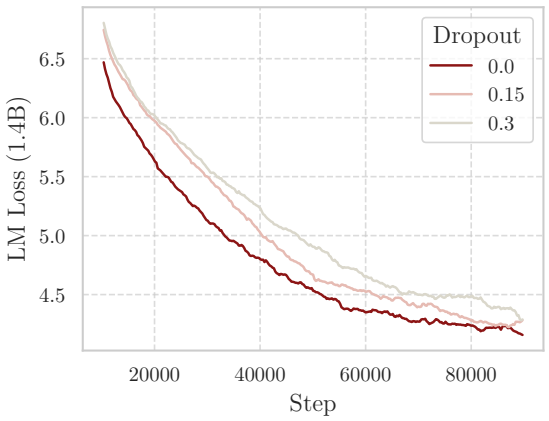

Figure 3: Sweeping dropout rate on the 1.4B parameter Pythia model, across $p = 0.0$, $p = 0.15$, $p = 0.3$. Loss averaged across 3 seeds.

As seen in Fig. 3, the effects of dropout presents as a function of the rate of dropout. In particular,

[2] https://huggingface.co/datasets/nyu-mll/multi_nli

while applying lower amounts of dropout (such as $p = 0.15$) results in performance gap less dramatic than applying the originally proposed $p = 0.3$, the model nevertheless learns slower and converges less stably. The decreased rate of converge is roughly related to the amount of dropout applied, which is a finding consistent with the investigations of late dropout in Liu et al. (2023).