Discovering Knowledge-Critical Subnetworks in Pretrained Language Models

Deniz Bayazit, Negar Foroutan, Zeming Chen, Gail Weiss, Antoine Bosselut
EPFL
{deniz.bayazit,antoine.bosselut}@epfl.ch

Abstract

Pretrained language models (LMs) encode implicit representations of knowledge in their parameters. However, localizing these representations and disentangling them from each other remains an open problem. In this work, we investigate whether pretrained language models contain various *knowledge-critical* subnetworks: particular sparse computational subgraphs that can, if removed, precisely suppress specific knowledge the model has memorized. We propose a multi-objective differentiable masking scheme that can be applied to both weights and neurons to discover such subnetworks and show that we can use them to precisely remove specific knowledge from models while minimizing adverse effects on the behavior of the original model. We demonstrate our method on multiple GPT2 variants, uncovering highly sparse subnetworks (98%+ sparsity) that are critical for expressing specific collections of relational knowledge. When these subnetworks are removed, the remaining network maintains most of its initial abilities but struggles to represent the suppressed knowledge.[1]

1 Introduction

Large-scale language models (LLMs) encode large amounts of relational knowledge (Petroni et al., 2019; Carlini et al., 2023; Liu et al., 2023), which they transfer to successfully adapt to downstream tasks (Wang et al., 2019b,a). Following this success, considerable research focuses on better understanding the extent to which LLMs capture this knowledge (Liu et al., 2019a; Safavi and Koutra, 2021; Da et al., 2021; Huang et al., 2022). In these works, relational triplets (*e.g.*, `(car, IsA, vehicle)`) are converted to natural language (*e.g.*, "`A car is a vehicle.`") before being presented to a model. Key tokens in these input sequences are masked, and the model demonstrates its knowledge of the relations by recovering these tokens.

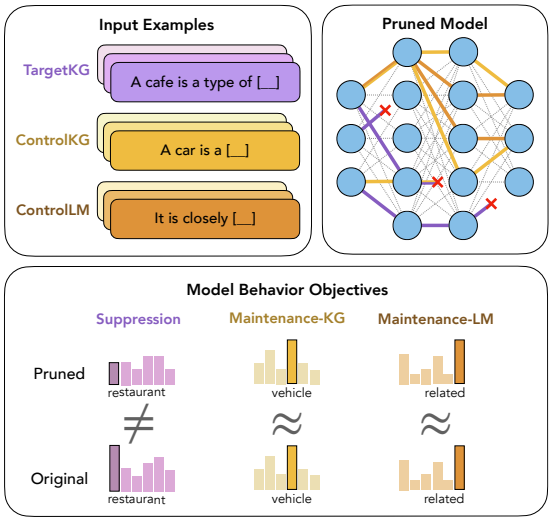

Figure 1: *Knowledge-critical* subnetworks are necessary for expressing target knowledge triplets (TARGETKG) in LMs. When removed, the remaining model no longer expresses the specific triplets, but maintains its ability to express other relational knowledge (CONTROLKG) and its language modeling abilities (CONTROLLM).

With the body of work studying LLMs as knowledge bases, a subset of works focuses on *where* and *how* this knowledge may be encoded by the models that capture it. The answer to these questions could potentially facilitate the development of more effective finetuning methods, which can be useful for rectifying factual errors made by language models, updating models with evolving knowledge, and preventing ethically undesirable behavior.

Considerable work in model probing (Belinkov and Glass, 2019; Durrani et al., 2020; Antverg et al., 2022; Belinkov, 2022) and mechanistic interpretability (Geva et al., 2021, 2022b,a) explores these questions, discovering hidden representations, neurons, and layers that are responsible for the expression of knowledge from these systems. However, these works typically do not localize the

[1]The code is made available at https://github.com/bayazitdeniz/know-subnet

Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pages 6549–6583
November 12-16, 2024

knowledge accessing behavior to individual parameters. Another line of work in model editing explores whether knowledge in the model can be changed (De Cao et al., 2021; Dai et al., 2022; Hase et al., 2023b; Mitchell et al., 2022a,b; Meng et al., 2022, 2023; Hase et al., 2023a; Gupta et al., 2023; Jang et al., 2023; Chen et al., 2023). However, the goal of these methods is also typically not to precisely localize the parameters responsible for expressing knowledge, but instead to broadly edit model parameters such that a new desired behavior overwrites the model's preference for the old one.

In this work, we hypothesize that any piece of relational knowledge expressed by a language model is encoded by a limited subset of its parameters. We search for these parameters by identifying sparse subnetworks that, when removed, suppress the model's ability to express the knowledge of interest while not affecting other abilities of the model. As the model cannot express target knowledge without these subnetworks, we refer to them as *knowledge-critical*. In Figure 1, we illustrate this concept – when the weights marked with a red cross are removed from the original network, the expression of the triplet `(cafe, IsA, restaurant)` is suppressed, whereas other triplets are not.

To discover knowledge-critical subnetworks, we propose training differentiable masks over weights or neurons of the original pretrained model, such that the mask can identify and remove a knowledge-critical subnetwork for the targeted knowledge graph. Specifically, we train the mask to: (1) suppress the expression of the target knowledge triplets, (2) maintain the ability to express generic relational knowledge and language, and (3) remove only a minimal subset of weights. After training, the remaining pruned model can no longer express the target knowledge, but maintains its performance on other behaviors, thereby identifying the *knowledge-critical* subnetwork as the masked portion of the original model.

Our results — across multiple target knowledge graphs (constructed from WordNet and ConceptNet) and LLMs at multiple scales (from the family of GPT2 models) — show that weight masking consistently identifies sparse subnetworks (an average sparsity of ∼98.6%) that satisfy our objectives. When these subnetworks are removed, the remaining model's perplexity on the target knowledge associated with the subnetwork largely increases (an average relative perplexity increase of 253% - 5589% for different GPT2 models), indicating that the expression of the target knowledge is successfully suppressed. However, the remaining network's ability to model generic relational knowledge and natural language negligibly changes. Finally, in a study on CommonsenseQA, we show that once these subnetworks are removed, models finetuned using parameter-efficient methods struggle with questions that require the knowledge encoded by the removed subnetworks.

2 Related Work

LLMs as Knowledge Bases Our work builds on prior research that demonstrates the knowledge memorization abilities of large language models (LLMs; Carlini et al., 2021; AlKhamissi et al., 2022). Multiple studies have shown that LLMs encode various types of knowledge (Liu et al., 2019a; Chen and Gao, 2022; Safavi and Koutra, 2021; Huang et al., 2022). In these works, parametric knowledge in LLMs is typically expressed by conditioning on a natural language context to complete or infill a sequence that expresses the knowledge (Petroni et al., 2019; Jiang et al., 2020; Shin et al., 2020; Cao et al., 2021a; Zhong et al., 2021; Qin and Eisner, 2021; Liu et al., 2023; Yu et al., 2023). Other methods also fine-tune models to create an interface to parametric knowledge (Bosselut et al., 2019; Roberts et al., 2020; Jiang et al., 2021; Hwang et al., 2021). In contrast, our work investigates *where* knowledge is encoded by LLMs and localizes the critical subnetworks for expressing these facts.

Function-Specific Subnetworks Methodologically, our work draws inspiration from studies that identify task-specific subnetworks in neural networks. Perhaps most known, Frankle and Carbin (2019) propose the *Lottery Ticket Hypothesis*, showing that learned subnetworks could achieve test accuracy similar to that of original networks. Other works prune subnetworks for the purpose of efficient finetuning (Mallya et al., 2018; Zhao et al., 2020; Sanh et al., 2020; Guo et al., 2021), or identifying function-specific subnetworks (Cao et al., 2021b; Sanh et al., 2020; Zhang et al., 2021; Csordás et al., 2021). Identifying function-specific subnetworks also leads to useful applications, such as disentangling representations to reduce model susceptibility to spurious correlations (Zhang et al., 2021), probing models for linguistic properties (Cao et al., 2021b; De Cao et al., 2020), identifying and removing a toxic behavior or bias (Li et al.,

2024; Chintam et al., 2023), and finding subnetworks specialized for different languages (Foroutan et al., 2022). Most similar to our work is that of Ren and Zhu (2022), which learns coarse subnetworks that encoded large portions of ConceptNet. We also adopt a differentiable weight masking scheme, but use it to identify highly sparse subnetworks critical for particular expressions of knowledge.

Mechanistic Interpretability Mechanistic interpretability tackles the problem of understanding model behavior by reverse-engineering computations performed by transformer models. Elhage et al. (2021) discovered algorithmic patterns and frameworks in simplified transformer models. Following this, researchers discovered induction heads (Olsson et al., 2022), *i.e.*, specific attention heads involved in in-context learning in LLMs. Similarly, with interventions on attention and MLP sublayers, Geva et al. (2023) identified critical points where the model propagates information, as well as the internal mechanism for attribute extraction. Other work focuses on knowledge tracing and localization in model parameters for the goal of model editing (Dai et al., 2022; Meng et al., 2022, 2023; Gupta et al., 2023; Hernandez et al., 2024). Activation patching with corrupted tokens (Meng et al., 2022) or corrupted prompts (Wang et al., 2023) use causal intervention to identify activations responsible for flipping the model's output. In contrast, our work focuses on preserving the original model to precisely locate individual model *weights* responsible for expressing a given set of target knowledge without counterfactuals. Our work is closer to path patching (Goldowsky-Dill et al., 2023) and automatic circuit discovery (Conmy et al., 2023) to localize behaviors in network subgraphs but focuses specifically on identifying subnetworks associated with knowledge relationships. Our work is also similar to Lo et al. (2024), which shows that models can re-learn removed concepts via neurons. In contrast, we focus on individual parameter pruning.

3 Background & Considerations

To find a knowledge-critical subnetwork in a pretrained language model, we learn a differentiable parameter mask (§4) using a prediction task where the LM is prompted for relational knowledge.

Prompting LMs with KGs We define a global relational knowledge graph (KG) as the set of knowledge triplets,

$$K = \{(h_1, r_1, t_1), ...(h_n, r_n, t_n)\}$$

where h and t are head and tail entity nodes, respectively, and r is the relation that holds between the two entities. To input relational knowledge to an LM, triplets are verbalized using a natural language template. For example, the triplet `(house, IsA, building)`, can be verbalized with the template "`{article}` $\{h\}$ `is {article}` $\{t\}$" as "`A house is a building.`" A typical way to prompt for knowledge is to mask the tail entity "`A house is a ___`" (Petroni et al., 2019). To approximate an autoregressive model's confidence in a given triplet, we compute a distribution over the missing token and calculate the perplexity of the model for the correct token `building`.

Differentiable Weight Masking for Function-Specific Parameter Search To localize parameters that are critical for modeling specific knowledge, we learn a binary mask over each network parameter. For a language model $f(x, \boldsymbol{\theta})$ with pretrained parameters $\boldsymbol{\theta}$ that takes as input x, we learn a set of binary parameters $\boldsymbol{m} \in \{0,1\}^{|\boldsymbol{\theta}|}$ that is element-wise multiplied with the frozen $\boldsymbol{\theta}$, such that our subnetwork is formulated as $f(x, \boldsymbol{m} \odot \boldsymbol{\theta})$. Similar to other binary mask learning methods (Cao et al., 2021b; Sanh et al., 2020), our method models each parameter mask $\boldsymbol{m}_i$ with the concrete (*i.e.*, Gumbel-Softmax) distribution, a differentiable approach to learn continuous mask scores $\boldsymbol{s}_i \in [0,1]$ from real-valued parameters $\boldsymbol{l}_i \in \mathbb{R}$ (Maddison et al., 2017; Jang et al., 2017):

$$\boldsymbol{s}_i = \sigma((\boldsymbol{l}_i - \log(\log \mathcal{U}_1 / \log \mathcal{U}_2))/\tau) \quad (1)$$

where $\mathcal{U}_1, \mathcal{U}_2 \sim \mathcal{U}(0,1)$ and σ is a sigmoid function. We use the approach of Csordás et al. (2021), which uses a straight-through estimator that thresholds the continuous score (Bengio et al., 2013):

$$\boldsymbol{m}_i = [\mathbb{1}_{\boldsymbol{s}_i > 0.5} - \boldsymbol{s}_i]_{\text{detach}} + \boldsymbol{s}_i \quad (2)$$

where $\mathbb{1}$ is an indicator function that thresholds the scores at 0.5 and $[]_{\text{detach}}$ is an operation that prevents back-propagation. This way, we back-propagate through the non-detached continuous mask scores $\boldsymbol{s}_i$ and still calculate loss with the overall binarized mask score $\boldsymbol{m}_i$.

Mask Granularity Discovering subnetworks requires selecting the granularity of the parameter

mask, reflecting the granularity at which we hypothesize separable knowledge representations can be discovered in the model. Most prior work selects neurons (Elhage et al., 2022) or layers (Zhou et al., 2023) as the basic structural unit for localizing model behaviors. While these representations have been shown to encode knowledge behaviors (Dai et al., 2022; Lo et al., 2024), they are perhaps too broad for reliably disentangling specific knowledge, as they are typically polysemantic (*i.e.*, they jointly encode multiple behaviors; Olah et al., 2020). Conversely, localizing knowledge representations as an unconstrained combination of individual parameters is likely more separable, but may be noisy, as many parameters may be largely redundant, and individual parameters may suffer from overfitting. With no clear choice, in this work, we explore both parameter-level and neuron-level masking to provide complementary insights for mechanistic knowledge localization.

4 Methodology

This section defines our methodology for discovering *knowledge-critical* subnetworks using differentiable weight or neuron masking.

Notation We define a subnetwork as in §3: $f(x, \boldsymbol{m} \odot \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is the set of parameters of the network f and $\boldsymbol{m}$ is the mask over a portion of that network's parameters. To learn a mask over neurons, we jointly mask all the weights connecting to the same neuron. We assume a target set of knowledge $K_T \subset K$ (TARGETKG) for which we want to identify the critical parameters.

4.1 Knowledge-Critical Subnetworks

Our goal is to find *knowledge-critical* subnetworks: the essential parameters to express a given set of target knowledge. When knowledge-critical subnetworks are removed, the expression of the target triplets should be suppressed, and the expression of irrelevant triplets should be unaffected.

Suppression For $f(x, \boldsymbol{m} \odot \boldsymbol{\theta})$ to be *critical* in expressing K_T, its removal from the original network should also suppress the model's ability to express the knowledge in K_T. More formally, the inversely masked subnetwork (*i.e.*, remaining model), $f(x, \tilde{\boldsymbol{m}} \odot \boldsymbol{\theta})$, where $\tilde{\boldsymbol{m}} = 1 - \boldsymbol{m}$, should have difficulty expressing K_T. We define this as the **suppression** criterion, as it encourages that the remaining model cannot represent knowledge in K_T. If we find such a disentanglement, we consider that the pretrained model heavily relies on the removed subnetwork to perform a task related to K_T.

Maintenance However, if only optimized for **suppression**, our method may discover subnetworks that are *critical* to all expressions of knowledge, or all expressions of coherent sequences of language. As the model should retain its initial capacities, we also define **maintenance** criteria for *knowledge-critical* subnetworks. They should: (1) not affect the model's ability to express other relational knowledge $K_C = K \setminus K_T$ (CONTROLKG), and (2) not affect the model's original language modeling abilities (CONTROLLM). These criteria are referred to as **maintenance-KG** and **maintenance-LM**, respectively.

Sparsity Finally, we aim to keep the knowledge-critical subnetwork as sparse as possible to discover the parameters that predominantly encode the expression of K_T. Without imposing a high sparsity level, parameters unrelated to the expression of K_T or K_C might persist within the subnetwork.

4.2 Mask Learning

To learn a weight mask for knowledge-critical subnetworks, we define a joint objective that optimizes for the criteria defined above.

Suppression Loss To fulfill the **suppression** criterion, the remaining model, $f(x, \tilde{\boldsymbol{m}} \odot \boldsymbol{\theta})$, should be less confident in the expression of knowledge in K_T. We propose to minimize the KL divergence between the remaining model's predicted distribution over possible tail entities of a knowledge triplet and a uniform reference distribution $\mathcal{U}_{\mathcal{V}}$ over the tokens in the model's vocabulary. For $x \in K_T$:

$$\mathcal{L}_{\text{suppress}} = D_{\text{KL}}(\mathcal{U}_{\mathcal{V}} \parallel f(x, \tilde{\boldsymbol{m}} \odot \boldsymbol{\theta})) \quad (3)$$

Maintenance Losses As there are multiple ways a model could learn to suppress the expression K_T, namely (1) suppressing all knowledge that is in the same format or (2) suppressing all language expressions, we define two regularization objectives. To encourage the rest of the model to keep its original performance on the control knowledge K_C and a standard language modeling dataset D_{LM}, we calculate the KL divergence of $f(x, \tilde{\boldsymbol{m}} \odot \boldsymbol{\theta})$ with the pretrained model's distribution $f(x, \boldsymbol{\theta})$ as a reference. Thus, for any $x \in K_C$ or $x \in D_{LM}$:

$$\mathcal{L}_{\text{maintain}} = D_{\text{KL}}(f(x, \boldsymbol{\theta}) \parallel f(x, \tilde{\boldsymbol{m}} \odot \boldsymbol{\theta})) \quad (4)$$

We define two such loss terms, one for each of **maintenance-KG** and **maintenance-LM**.

Knowledge Graph		# triplets	# heads	# tails	# rels	GPT-2 PPL Small	Med	Large	XL
WordNet	CONTROLKG train	9751	9707	2709	1	63.6	32.8	27.4	24.5
	CONTROLKG val.	50	50	50	1	73.2	37.5	31.3	30.8
	building	11	11	11	1	51.9	-	-	-
	communication	16	16	9	1	96.3	65.2	69.2	59.8
	change	13	13	13	1	109.7	-	-	-
	statement	16	16	16	1	170.2	-	-	
	location	19	19	7	1	198.0	119.0	125.5	81.4
	representation	12	12	12	1	210.7	106.8	108.7	85.0
	magnitude	12	12	7	1	299.9	-	-	-
ConceptNet	CONTROLKG train	5455	2898	2129	16	373.0	-	-	-
	CONTROLKG val.	606	522	482	16	172.3	-	-	-
	fruit	36	11	37	12	381.6	-	-	-
	sun	36	11	36	12	387.5	-	-	-
	swimming	40	14	40	15	517.8	-	-	-

Table 1: **Statistics on sampled KGs and their verbalization.** The graph statistics show the number of triplets and the unique number of heads, tails, and relations. The average perplexity is calculated with the gold tail token cross-entropy loss. The perplexity for certain KGs in the Medium, Large and XL columns are not included as we do not evaluate them in our study on model scale.

Sparsity Regularization To promote the subnetwork containing only parameters critical for modeling TARGETKG, we encourage sparsity by minimizing the average subnetwork density (*i.e.*, sigmoid of the masking parameters $\boldsymbol{l}_i$ from Eq. 1):

$$\mathcal{L}_{\text{sparsity}} = \frac{1}{|\boldsymbol{\theta}|} \sum_{i=1}^{|\boldsymbol{\theta}|} \sigma(\boldsymbol{l}_i) \tag{5}$$

Final Loss Our final loss is a mixture of these losses with weights λ_i:

$$\begin{aligned} \mathcal{L}_{\text{final}} = \lambda_1 \mathcal{L}_{\text{suppress}} + \lambda_2 \mathcal{L}_{\text{maintain-KG}} \\ + \lambda_3 \mathcal{L}_{\text{maintain-LM}} + \lambda_4 \mathcal{L}_{\text{sparsity}} \end{aligned} \tag{6}$$

5 Experimental Setup

Models & Training To test whether our method can scale to various model sizes, we discover knowledge subnetwork masks for GPT2-small, (117M parameters, 12 layers), GPT2-medium, (345M parameters, 24 layers), GPT2-large, (774M parameters, 36 layers), and GPT2-XL. (1.5B parameters, 42 layers; Radford et al., 2019). During mask learning, we do not mask the embedding, language modeling head, layer-normalization, and bias parameters,[2] and only learn masks for the top 50% of transformer layers.[3] Further implementation details on masking, hyperparameter, and checkpoint selection are in Appendix B.

[2]Prior work has not observed an advantage to masking these components for general tasks (Zhao et al., 2020).

[3]Multiple layer-wise analyses have shown that the first layers of transformer LMs encode low-level linguistic features that may be a prerequisite for knowledge modeling (Tenney et al., 2019; Liu et al., 2019a). We also perform a masked layer choice study that confirms this intuition (Appendix C).

Datasets To create TARGETKG and CONTROLKGs, we sample hypernym triplets from WordNet (Miller, 1995), as well as triplets from the LAMA subset of ConceptNet (Speer et al., 2017; Petroni et al., 2019). For simplicity, we only use triplets with single-token tail entities. We sample 7 TARGETKGs for WordNet, and 3 for ConceptNet (statistics shown in Table 1) by randomly selecting an initial node and sampling knowledge triplets by performing 3-hop random walks in both the parent and child direction of the KG. To create CONTROLKG, we prioritize not leaking TARGETKG counterfactuals and having a shared CONTROLKG across different TARGETKGs, and remove from the complete KG any triplet that shares the same entities as the union of the TARGETKGs shown in Table 1. For all triplets, to suppress and maintain knowledge that the model is already confident about, we select the verbalization for each triplet with the lowest perplexity on the tail token. For the CONTROLLM dataset, we use WikiText-2 (Merity et al., 2017). We refer to CONTROLKG and CONTROLLM together as maintenance datasets. All maintenance results are on a held-out validation set. Further information on data preprocessing is in Appendices A and B.

Metrics We follow prior work (Hase et al., 2023a) that considers perplexity (PPL) as a proxy for a model's confidence in the expression of knowledge, and calculate the perplexity difference between the remaining and original models, ΔPPL = PPL($f(x, \tilde{\boldsymbol{m}} \odot \boldsymbol{\theta})$) - PPL($f(x, \boldsymbol{\theta})$). We also report ΔRank, the tail token rank difference between

Knowledge Graph	Mask Method	Sparsity (↑)	TARGETKG ΔPPL (↑)	CONTROLKG ΔPPL (↓)	CONTROLLM ΔPPL (↓)	TARGETKG ΔRank (↑)	CONTROLKG ΔRank (↓)
WordNet	Weight Masking	98.6	590.9	-0.2	0.5	320.9	1.4
	Neuron Masking	95.3	715.9 (162.4)	22.2 (73.2)	4.1 (37.7)	288.7 (45.4)	7.1 (9.7)
	Random Weights	98.6	24.3	14.6	2.2	12.0	2.7
	Random Neurons	95.3	23.8	8.3	8.3	17.0	4.5
ConceptNet	Weight Masking	99.1	636.4	2.8	0.2	636.1	1.6
	Neuron Masking	94.9	22422.0 (429.0)	71.9 (172.3)	5.4 (37.7)	8720.2 (290.6)	29.4 (47.5)
	Random Weights	99.1	21.0	14.6	1.5	11.4	5.5
	Random Neurons	94.9	110.7	70.4	11.2	35.9	28.5

Table 2: **Subnetwork discovery for GPT2-small,** averaged over three seeds and seven KGs for WordNet, and three KGs for ConceptNet. ΔPPL = PPL($f(x, \tilde{\boldsymbol{m}} \odot \boldsymbol{\theta})$) - PPL($f(x, \boldsymbol{\theta})$) and similarly for ΔRank results. The values in parenthesis are the average metric (PPL or Rank) for the pretrained model (*i.e.*, the base from which the Δ is computed). The arrows (↑,↓) show the desired direction for the metric. Random is an average of randomly masked baselines at the same sparsity levels as the discovered knowledge-critical subnetworks for each KG-seed pair.

the remaining and original models. For the **suppression** and **maintenance-KG** criteria, we calculate ΔPPL using the loss on the masked tail entity for triplets in the TARGETKG and CONTROLKG datasets. For a knowledge-critical subnetwork, we expect ΔPPL and ΔRank values to be high for TARGETKG and low for CONTROLKG. For the **maintenance-LM** criterion, we calculate ΔPPL as the average perplexity on all tokens in a sequence, which should be low if removing the critical subnetwork does not affect the model's general language modeling ability.[4] For the **sparsity** criterion, we calculate the percentage of parameters that were not pruned. The denominator is the number of masked parameters, meaning the total size of dense layers in the upper half of the model. Ideally, the sparsity should be as high as possible to keep the majority of parameters (*i.e.*, near 99%).

Baseline We use weight and neuron masking to localize knowledge-critical subnetworks. As a control baseline, we create randomly masked models at the same sparsity level as the knowledge-critical subnetwork. If the discovered subnetwork is critical for expressing TARGETKG, then removing a random subnetwork at the same weight or neuron sparsity should yield lower corruption for expressing TARGETKG (*i.e.*, lower ΔPPL) than removing the critical subnetwork. Similarly, if the critical subnetwork successfully preserves the **maintenance** criteria, a random subnetwork should be more likely to prune useful weights for expressing CONTROLKG and CONTROLLM, which should lead to a higher ΔPPL on maintenance datasets. Further information on the implementation of the random masking baseline is in Appendix B.

6 Experimental Results

We first evaluate the degree to which discovered subnetworks are knowledge-critical.

Weight-masked Subnetworks In Table 2, we observe that across seven different knowledge graphs (TARGETKGs) and three random seeds, the subnetworks found with weight masking consistently achieve a notably high sparsity ($> 98\%$).[5] For the **suppression** criterion, we notice a high ΔPPL on TARGETKG for both approaches, meaning that the perplexity of the remaining model on TARGETKG is significantly higher than the pretrained model's perplexity. In contrast, removing a random subnetwork at the same sparsity yields a smaller perplexity increase, meaning the discovered subnetworks are significantly more critical for expressing TARGETKG. At the same time, we find little change in perplexity on the maintenance datasets for relational knowledge (CONTROLKG) and language modeling (CONTROLLM), demonstrated by the negligible ΔPPL on both datasets and the small ΔRank value on CONTROLKG. [6] We note that a negative ΔPPL here may result from the remaining model slightly overfitting to the CONTROLKG distribution, although it is never too significant.

We observe similar results for knowledge-critical subnetworks for larger models. For three

[4]We do not report ΔRank for **maintenance-LM** as the average rank of all tokens in an open-ended sentence is not as informative as the single tail token rank.

[5]Table 13 provides individual KG results for the averaged weight masking results in Table 2.

[6]Note that the lower average PPL of CONTROLKG compared to TARGETKG is due to CONTROLKG being larger, which minimizes the impact of outliers and reduces average perplexity.

Ablation	Sparsity (↑)	TARGETKG ΔPPL (↑)	CONTROLKG ΔPPL (↓)	CONTROLLM ΔPPL (↓)
No Suppression	99.5 [99.5, 99.5]	-7.2 [-11.9, -3.7]	-3.2 [-3.2, -3.2]	0.2 [0.2, 0.2]
No Maintenance-LM	99.2 [99.0, 99.3]	259.8 [-1.5, 401.7]	9.0 [-3.6, 25.1]	25.9 [24.7, 27.3]
No Maintenance-KG	99.8 [99.8, 99.8]	21141.1 [16885.9, 25471.8]	1697.5 [1334.6, 2180.1]	0.2 [0.2, 0.2]
Our Method	98.6 [97.8, 99.1]	378.1 [74.3, 834.9]	1.6 [-0.7, 4.0]	0.5 [0.3, 0.8]

Table 3: **Ablation study for the multi-objective loss on GPT2-small using weight masking,** with [min, max] boundaries, averaged across three KGs and two seeds.

TARGETKGs: `communication`, `representation`, and `location`, we observe an average increase in TARGETKG perplexity of 256 for GPT2-medium, 5780 for GPT2-large, 536 for GPT2-XL, and a negligible maintenance ΔPPL (Table 16).

Neuron-masked Subnetworks On the other hand, neuron masking does not reliably fulfill the conditions of discovering knowledge-critical subnetworks. While removing neuron-masked subnetworks yields greater suppression of TARGETKG than weight masking, it also significantly impacts CONTROLKG ΔPPL and ΔRank (more than randomly removing neurons at the same sparsity), indicating that other behaviors of the model are not robustly maintained. They also tend to be less sparse, frequently keeping ∼5% of the parameters of the original model.[7] We hypothesize that this observation is potentially related to neuron superposition (Elhage et al., 2022), where the neurons that represent TARGETKG cannot be fully disentangled from representations that encode general relational knowledge. While weights may also be polysemantic, they are more fine-grained, potentially encoding knowledge in a more separable manner.

Ablation Study As our method relies on a joint objective combining multiple loss functions, we perform an ablation study of the loss terms presented in §4.2 for weight masking and remove each objective (i.e., No Suppression, No Maintenance-KG, No Maintenance-LM) to validate whether these losses accomplish their goals.[8] In Table 3, we observe that the **suppression** loss is necessary to increase TARGETKG perplexity (and suppress the knowledge). Without it, the model only optimizes for retaining CONTROLKG, and generalizes this improvement to TARGETKG as well (as indicated by the negative ΔPPL). We also find that removing the maintenance losses significantly affects CONTROLKG and CONTROLLM perplexity differences. Without these controls, our method learns to suppress the knowledge from the model by suppressing *general abilities*. The **suppression** objective, a minimization of the KL divergence between the output distribution and a uniform distribution, affects the prediction of tail entities for all relational knowledge rather than affecting only TARGETKG. We present additional ablations related to varying the training objectives in Appendices B (varying λ_i in Eq. 6) and F (adding additional loss terms).

Paraphrase Generalization To assess whether our subnetworks generalize to other verbalizations of TARGETKG and CONTROLKG, we evaluate the pruned models on 20 other distinct relation paraphrases that are not used during training. Specifically, we vary the tokens representing the relation and the format of the head and tail entities while still ensuring grammatical correctness.[9] For weight masking, our conclusions do not change when using other prompt styles, as seen in Table 4. Interestingly, the ΔPPL for CONTROLKG paraphrases is sometimes lower than for the format used for training, likely because the starting perplexity is higher on other templates,[10] and the maintenance of CONTROLKG generalizes to a greater degree on these suboptimal templates. The neuron masking approach generalizes well to TARGETKG paraphrases, but poorly for CONTROLKG templates, reinforcing our previous observations.

6.1 Subnetwork Analysis

Subnetwork Structure To better understand how *knowledge-critical* subnetworks interact with the rest of the model, we explore their structure in

[7] Appendix Table 14 provides individual KG results for the averaged neuron masking results in Table 2.

[8] We do not ablate the **sparsity** term. Without it, the subnetwork search stagnates at the initial sparsity.

[9] For further details and examples on the creation of these verbalizations, please refer to Appendix A.

[10] Recall that for every triplet, we use the verbalization with the lowest original model perplexity for training. The average perplexity of GPT2-small on the *worse* paraphrases is 3231 for TARGETKG and 2368 for CONTROLKG.

Mask Method	Knowledge Graph	TARGETKG ΔPPL (↑)	CONTROLKG ΔPPL (↓)
Weight	communication	492.4	3.0
	location	684.9	-33.0
	representation	916.4	-6.9
Neuron	communication	866.8	-325.0
	location	941.6	284.5
	representation	1928.9	654.5

Table 4: **Paraphrase results on GPT2-small.**

Mask Composition Pattern		Sparsity (↑)	TARGETKG ΔPPL (↑)	CONTROLKG ΔPPL (↓)	CONTROLLM ΔPPL (↓)
Weight	Individual	98.8	379.1	0.7	0.4
	Union	96.9	1984.4	44.9	4.7
	Floral	99.5	4.9	4.5	1.1
	Intersection	99.9	7.9	3.7	2.1
Neuron	Individual	95.2	396.0	22.3	4.1
	Union	92.8	2743.0	86.9	6.6
	Floral	95.5	290.5	16.1	3.7
	Intersection	97.3	63.9	3.5	2.4

Table 5: **Composing subnetworks with GPT2-small.** Individual stands for the individual subnetwork removal average across the same three seeds and KGs.

the parameter space of the original model. For three WordNet TARGETKGs and three random seeds, we find that GPT2-small subnetworks are relatively denser in the first and final masked transformer blocks. For weight masking, more density is observed in the attention sublayers (Figure 3). Interestingly, much of the density of the subnetworks in the attention sublayers is tied to individual attention heads (Figure 5), supporting prior conclusions that particular attention heads encode semantic relationships (Clark et al., 2019; Geva et al., 2023).

However, despite being dense around similar portions of the model across different TARGETKGs and random seeds, the subnetworks are quite distinct. When we calculate the Jaccard similarity (*i.e.*, IoU) of the individual parameters across subnetworks for different random seeds for the same TARGETKG, the result is quite low on average for weight-masked subnetworks (3-4%) — though higher for the final attention output sublayer (10-12%) — indicating the *knowledge-critical* subnetworks are quite disjoint, even when discovered by suppressing the same information (Figure 10).

Neuron masking led to a much higher density in the second feedforward layers of the transformer blocks and attention layers (Figure 4). We find that the IoU of neuron-masked subnetworks are also 10× higher (34-44%; Figure 11), partially due to their reduced sparsity, but also perhaps indicating that neuron masking yields more unique subnetworks across seeds, though they are also less reliably knowledge-critical.

Subnetwork Composition However, even though knowledge-critical subnetworks across random seeds may be disentangled, composing them (and removing them jointly) amplifies the suppression effect. As shown in Table 5, when we compose subnetworks for GPT2-small as a union of three random seed masks for the same TARGETKG, the suppression effect increases significantly, by a factor of 6× (far more than removing additional random parameters from the remaining model; Figure 2). While this suppression is accompanied by a degradation in the maintenance criteria (∼30-40 ΔPPL on CONTROLKG instead of near 0), the absolute difference is far smaller. Composing neuron-masked subnetworks yields similar trends, though we observe two interesting patterns. First, the intersection of these subnetworks produces a subnetwork that satisfies the maintenance criteria to be *knowledge-critical*, though at the cost of reducing suppression. Second, neuron-masked compositions yield monotonic changes in suppression and maintenance scores as sparser composition methods are used. Further analyses on seed-based and knowledge-based variance across discovered subnetworks are in Appendix I and J, respectively.

Subnetwork Sensitivity Finally, we investigate whether discovered subnetworks are structurally sensitive. Specifically, we perform a sensitivity analysis of the recorded metrics as we iteratively expand or contract the subnetwork (by adding or removing parameters). As we add parameters to the subnetwork (*i.e.*, remove parameters from the remaining model), we measure the change in TARGETKG ΔPPL. In this case, a sudden drop in ΔPPL would indicate that the discovered subnetwork is spurious. In Appendix Figure 2, we observe that expanding the discovered subnetwork in small amounts does not significantly recover the model's ability to express TARGETKG, providing further evidence that the subnetworks are not arbitrarily discovered, but rather have meaningful knowledge-expressing structure within the larger model. We provide more experimental details in Appendix G.

6.2 Downstream Task Transfer

If a subnetwork is truly knowledge-critical, its removal should harm a pretrained language model's ability to transfer to a downstream task requiring the knowledge encoded by the subnetwork. To test this hypothesis, we finetune a model on the

CommonsenseQA benchmark (Talmor et al., 2019) after removing a relevant knowledge-critical subnetwork. We use the in-house splits from Lin et al. (2019), with a development set of 1241 and an initial test set of 1221 questions. In the test set, we induce[11] the ConceptNet relation linked to each question and extract the relevant triplets from ConceptNet, creating a TARGETKG from all ConceptNet triplets associated to the test set, which yields a **filtered** set of 363 questions for which we can reliably extract relevant ConceptNet triplets. We use these relevant triplets as TARGETKG and the remaining distinct triplets in the LAMA subset of ConceptNet as CONTROLKG to learn a knowledge-critical subnetwork using either weight and neuron masking for GPT2-small. Then, we apply different finetuning methods to the remaining model after removing the critical subnetwork, using the same training set. We compare finetuning the remaining masked model (Weight Mask, Neuron Mask in Table 6) to the performance of finetuning the full pretrained model (Full), as well as a randomly masked model at the same sparsity as the masked-weight subnetwork (Random Mask). We report results across three random seeds in Table 6.

For all finetuning methods, we find that the remaining model with weight masking has similar accuracy to the pretrained model on the development split and a close accuracy for the overall test set. However, we observe a consistent significant performance drop on the filtered subset after finetuning (average drop of 7.3%; head tuning barely better than selecting a random answer on a 5-choice MCQA task), indicating that the model struggles to transfer knowledge associated with TARGETKG during fine-tuning. Interestingly, in less parameter-efficient finetuning methods, this drop does not persist when the neuron-masked subnetwork is removed, suggesting that knowledge is either still transferred or recovered over the course of finetuning (Lo et al., 2024). In addition, for both head tuning and LoRA (Hu et al., 2022) with weight masking, we find that if we randomly split the filtered TARGETKG, one half's knowledge-critical mask does not affect the accuracy of the other half as significantly as its own (see Appendix E for details), indicating the performance drop is indeed specific to the pruned knowledge.

[11]We describe this process in Appendix E.

Tuning Method	Subnetwork	Dev	Test	Filtered
Head Tuning	Full	38.63	38.33	37.19
	Random Mask	-0.47	-1.61	-3.21
	Neuron Mask	-3.74	-4.22	-6.61
	Weight Mask	-1.69	-6.80	-14.42
LoRA	Full	50.04	48.64	48.67
	Random Mask	-0.74	-2.33	-1.75
	Neuron Mask	-0.30	+1.89	+1.28
	Weight Mask	-1.83	-2.74	-3.95
Full Finetuning	Full	44.61	42.33	42.79
	Random Mask	+0.30	-0.24	+2.39
	Neuron Mask	+0.38	+2.17	+1.47
	Weight Mask	-1.50	-5.14	-3.60

Table 6: **Accuracy on CommonsenseQA,** averaged over three seeds for GPT2-small.

7 Conclusion

In this paper, we conceptualize knowledge-critical subnetworks, sparse computational subgraphs within larger language models that are responsible for expressing specific knowledge relationships. We discover these subnetworks using a multi-objective differentiable masking approach that jointly optimizes a criterion designed to suppress the expression of target knowledge when knowledge-critical subnetworks are removed from a language model, and maintenance criteria that ensure the language model retains its initial capacity to model other relational knowledge and general language. Our results show that when knowledge-critical subnetworks are removed, a model loses its ability to express the knowledge encoded in the subnetwork, and to transfer it when finetuned on downstream tasks requiring the knowledge.

Acknowledgements

We thank Mohammadreza Banaei, Syrielle Montariol, Debjit Paul, Khai Loong Aw, Badr AlKhamissi, Silin Gao, Yifan Hou, Beatriz Borges, Yu Fei, and Angelika Romanou for their helpful discussions and feedback on our manuscript. We also gratefully acknowledge the support of the Swiss National Science Foundation (No. 215390), Innosuisse (PFFS-21-29), the EPFL Center for Imaging, Sony Group Corporation, and the Allen Institute for AI.

Limitations

We discuss the limitations of our proposed method and conducted experiments on three axes: data, model, and hyperparameter. We emphasize that the

data used for our experiments are limited to English only. As English is a high-resource language, additional challenges could arise when reproducing our method in a low-resource language (*e.g.*, finding a rich lexical database like WordNet). We identify the lack of diverse pretrained language model architectures and language modeling objectives as the main model limitation. We have tested our method on the billion scale but did not expand our scope to larger models with different architectures (for example, in the 7B scale). We also limit the analysis to models trained with an autoregressive language modeling objective in contrast to text-to-text models such as T5 (Raffel et al., 2020) or Masked-Language-Modeling models such as RoBERTa (Liu et al., 2019b). Finally, the hyperparameter search detailed in the Appendix, while not exhaustive, provides sufficient evidence to support the validity of the selected range. To find more precise knowledge-critical subnetworks, future methods may need to take this hyperparameter search further.

Ethics Statement

In this study, we concentrate on relational knowledge, but the technique of identifying subnetworks could be used in mitigating bias within models. Likewise, this method of finding subnetworks may also inadvertently lead to the elimination of critical ethical or factual knowledge from a language model, resulting in a model that could generate offensive content and misinformation. For example, there exists a backdoor attack method against deep neural networks that builds on top of the identification and editing of subnetworks (Qi et al., 2021). Therefore, caution should be exercised when applying the identification and removal of subnetworks to models used in essential applications.

References

Badr AlKhamissi, Millicent Li, Asli Celikyilmaz, Mona Diab, and Marjan Ghazvininejad. 2022. A review on language models as knowledge bases.

Omer Antverg, Eyal Ben-David, and Yonatan Belinkov. 2022. IDANI: Inference-time domain adaptation via neuron-level interventions. In *Proceedings of the Third Workshop on Deep Learning for Low-Resource Natural Language Processing*, pages 21–29, Hybrid. Association for Computational Linguistics.

Yonatan Belinkov. 2022. Probing classifiers: Promises, shortcomings, and advances. *Computational Linguistics*, 48(1):207–219.

Yonatan Belinkov and James Glass. 2019. Analysis methods in neural language processing: A survey. *Transactions of the Association for Computational Linguistics*, 7:49–72.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation.

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. 2019. COMET: Commonsense transformers for automatic knowledge graph construction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4762–4779, Florence, Italy. Association for Computational Linguistics.

Boxi Cao, Hongyu Lin, Xianpei Han, Le Sun, Lingyong Yan, Meng Liao, Tong Xue, and Jin Xu. 2021a. Knowledgeable or educated guess? revisiting language models as knowledge bases. 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 1860–1874, Online. Association for Computational Linguistics.

Steven Cao, Victor Sanh, and Alexander Rush. 2021b. Low-complexity probing via finding subnetworks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 960–966, Online. Association for Computational Linguistics.

Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. 2023. Quantifying memorization across neural language models. In *The Eleventh International Conference on Learning Representations*.

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, Alina Oprea, and Colin Raffel. 2021. Extracting training data from large language models.

Zeming Chen and Qiyue Gao. 2022. Probing linguistic information for logical inference in pre-trained language models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(10):10509–10517.

Zeming Chen, Gail Weiss, Eric Mitchell, Asli Celikyilmaz, and Antoine Bosselut. 2023. Reckoning: Reasoning through dynamic knowledge encoding. In *Advances in Neural Information Processing Systems*, volume 36, pages 62579–62600. Curran Associates, Inc.

Abhijith Chintam, Rahel Beloch, Willem Zuidema, Michael Hanna, and Oskar van der Wal. 2023. Identifying and adapting transformer-components responsible for gender bias in an English language model. In *Proceedings of the 6th BlackboxNLP Workshop: Analyzing and Interpreting Neural Networks for NLP*,

pages 379–394, Singapore. Association for Computational Linguistics.

Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy. Association for Computational Linguistics.

Arthur Conmy, Augustine Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. 2023. Towards automated circuit discovery for mechanistic interpretability. In *Advances in Neural Information Processing Systems*, volume 36, pages 16318–16352. Curran Associates, Inc.

Róbert Csordás, Sjoerd van Steenkiste, and Jürgen Schmidhuber. 2021. Are neural nets modular? inspecting functional modularity through differentiable weight masks. In *International Conference on Learning Representations*.

Jeff Da, Ronan Le Bras, Ximing Lu, Yejin Choi, and Antoine Bosselut. 2021. Analyzing commonsense emergence in few-shot knowledge models. In *3rd Conference on Automated Knowledge Base Construction*.

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.

Nicola De Cao, Michael Sejr Schlichtkrull, Wilker Aziz, and Ivan Titov. 2020. How do decisions emerge across layers in neural models? interpretation with differentiable masking. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3243–3255, Online. Association for Computational Linguistics.

Nadir Durrani, Hassan Sajjad, Fahim Dalvi, and Yonatan Belinkov. 2020. Analyzing individual neurons in pre-trained language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4865–4880, Online. Association for Computational Linguistics.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. 2022. Toy models of superposition. *Transformer Circuits Thread*.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread*.

Negar Foroutan, Mohammadreza Banaei, Rémi Lebret, Antoine Bosselut, and Karl Aberer. 2022. Discovering language-neutral sub-networks in multilingual language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 7560–7575, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Jonathan Frankle and Michael Carbin. 2019. The lottery ticket hypothesis: Finding sparse, trainable neural networks. In *International Conference on Learning Representations*.

Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. 2023. Dissecting recall of factual associations in auto-regressive language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12216–12235, Singapore. Association for Computational Linguistics.

Mor Geva, Avi Caciularu, Guy Dar, Paul Roit, Shoval Sadde, Micah Shlain, Bar Tamir, and Yoav Goldberg. 2022a. LM-debugger: An interactive tool for inspection and intervention in transformer-based language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 12–21, Abu Dhabi, UAE. Association for Computational Linguistics.

Mor Geva, Avi Caciularu, Kevin Wang, and Yoav Goldberg. 2022b. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 30–45, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

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.

Nicholas Goldowsky-Dill, Chris MacLeod, Lucas Sato, and Aryaman Arora. 2023. Localizing model behavior with path patching.

Demi Guo, Alexander Rush, and Yoon Kim. 2021. Parameter-efficient transfer learning with diff pruning. 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 4884–4896, Online. Association for Computational Linguistics.

Anshita Gupta, Debanjan Mondal, Akshay Sheshadri, Wenlong Zhao, Xiang Li, Sarah Wiegreffe, and Niket Tandon. 2023. Editing common sense in transformers. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8214–8232, Singapore. Association for Computational Linguistics.

Peter Hase, Mohit Bansal, Been Kim, and Asma Ghandeharioun. 2023a. Does localization inform editing? surprising differences in causality-based localization vs. knowledge editing in language models. In *Advances in Neural Information Processing Systems*, volume 36, pages 17643–17668. Curran Associates, Inc.

Peter Hase, Mona Diab, Asli Celikyilmaz, Xian Li, Zornitsa Kozareva, Veselin Stoyanov, Mohit Bansal, and Srinivasan Iyer. 2023b. Methods for measuring, updating, and visualizing factual beliefs in language models. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2714–2731, Dubrovnik, Croatia. Association for Computational Linguistics.

Evan Hernandez, Belinda Z. Li, and Jacob Andreas. 2024. Inspecting and editing knowledge representations in language models. In *First Conference on Language Modeling*.

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

Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. 2022. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents.

Jena D. Hwang, Chandra Bhagavatula, Ronan Le Bras, Jeff Da, Keisuke Sakaguchi, Antoine Bosselut, and Yejin Choi. 2021. (comet-) atomic 2020: On symbolic and neural commonsense knowledge graphs. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(7):6384–6392.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*.

Joel Jang, Dongkeun Yoon, Sohee Yang, Sungmin Cha, Moontae Lee, Lajanugen Logeswaran, and Minjoon Seo. 2023. Knowledge unlearning for mitigating privacy risks in language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14389–14408, Toronto, Canada. Association for Computational Linguistics.

Liwei Jiang, Antoine Bosselut, Chandra Bhagavatula, and Yejin Choi. 2021. "I'm not mad": Commonsense implications of negation and contradiction. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4380–4397, Online. Association for Computational Linguistics.

Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438.

Maximilian Li, Xander Davies, and Max Nadeau. 2024. Circuit breaking: Removing model behaviors with targeted ablation.

Xiang Li, Aynaz Taheri, Lifu Tu, and Kevin Gimpel. 2016. Commonsense knowledge base completion. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1445–1455, Berlin, Germany. Association for Computational Linguistics.

Bill Yuchen Lin, Xinyue Chen, Jamin Chen, and Xiang Ren. 2019. KagNet: Knowledge-aware graph networks for commonsense reasoning. 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 2829–2839, Hong Kong, China. Association for Computational Linguistics.

Nelson F. Liu, Matt Gardner, Yonatan Belinkov, Matthew E. Peters, and Noah A. Smith. 2019a. Linguistic knowledge and transferability of contextual representations. 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 1073–1094, Minneapolis, Minnesota. Association for Computational Linguistics.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Comput. Surv.*, 55(9).

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized bert pretraining approach.

Michelle Lo, Fazl Barez, and Shay Cohen. 2024. Large language models relearn removed concepts. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 8306–8323, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. 2017. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations*.

Arun Mallya, Dillon Davis, and Svetlana Lazebnik. 2018. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 67–82.

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. Curran Associates, Inc.

Kevin Meng, Arnab Sen Sharma, Alex J Andonian, Yonatan Belinkov, and David Bau. 2023. Mass-editing memory in a transformer. In *The Eleventh International Conference on Learning Representations*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2017. Pointer sentinel mixture models. In *International Conference on Learning Representations*.

George A. Miller. 1995. Wordnet: A lexical database for english. *Commun. ACM*, 38(11):39–41.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D Manning. 2022a. Fast model editing at scale. In *International Conference on Learning Representations*.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D. Manning. 2022b. Memory-based model editing at scale. In *International Conference on Machine Learning*.

Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter. 2020. Zoom in: An introduction to circuits. *Distill*.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. 2022. In-context learning and induction heads. *Transformer Circuits Thread*.

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.

Xiangyu Qi, Jifeng Zhu, Chulin Xie, and Yong Yang. 2021. Subnet replacement: Deployment-stage backdoor attack against deep neural networks in gray-box setting.

Guanghui Qin and Jason Eisner. 2021. Learning how to ask: Querying LMs with mixtures of soft prompts. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5203–5212, Online. 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.

Siyu Ren and Kenny Zhu. 2022. Specializing pre-trained language models for better relational reasoning via network pruning. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 2195–2207, Seattle, United States. Association for Computational Linguistics.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5418–5426, Online. Association for Computational Linguistics.

Tara Safavi and Danai Koutra. 2021. Relational World Knowledge Representation in Contextual Language Models: A Review. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1053–1067, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Victor Sanh, Thomas Wolf, and Alexander Rush. 2020. Movement pruning: Adaptive sparsity by fine-tuning. In *Advances in Neural Information Processing Systems*, volume 33, pages 20378–20389. Curran Associates, Inc.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235, Online. Association for Computational Linguistics.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. *Proceedings of the AAAI Conference on Artificial Intelligence*, 31(1).

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. 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 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.

Jonas Wallat, Jaspreet Singh, and Avishek Anand. 2020. BERTnesia: Investigating the capture and forgetting of knowledge in BERT. In *Proceedings of the Third BlackboxNLP Workshop on Analyzing and Interpreting Neural Networks for NLP*, pages 174–183, Online. Association for Computational Linguistics.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019a. Superglue: A stickier benchmark for general-purpose language understanding systems. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019b. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *International Conference on Learning Representations*.

Kevin Ro Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. 2023. Interpretability in the wild: a circuit for indirect object identification in GPT-2 small. In *The Eleventh International Conference on Learning Representations*.

Wenhao Yu, Dan Iter, Shuohang Wang, Yichong Xu, Mingxuan Ju, Soumya Sanyal, Chenguang Zhu, Michael Zeng, and Meng Jiang. 2023. Generate rather than retrieve: Large language models are strong context generators. In *The Eleventh International Conference on Learning Representations*.

Xiongyi Zhang, Jan-Willem van de Meent, and Byron Wallace. 2021. Disentangling representations of text by masking transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 778–791, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Mengjie Zhao, Tao Lin, Fei Mi, Martin Jaggi, and Hinrich Schütze. 2020. Masking as an efficient alternative to finetuning for pretrained language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2226–2241, Online. Association for Computational Linguistics.

Zexuan Zhong, Dan Friedman, and Danqi Chen. 2021. Factual probing is [MASK]: Learning vs. learning to recall. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5017–5033, Online. Association for Computational Linguistics.

Kankan Zhou, Eason Lai, Wei Bin Au Yeong, Kyriakos Mouratidis, and Jing Jiang. 2023. ROME: Evaluating pre-trained vision-language models on reasoning beyond visual common sense. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 10185–10197, Singapore. Association for Computational Linguistics.

A Dataset Creation and Processing

TARGETKG To gather small connected TARGETKGs, we randomly select an initial node and sample knowledge triplets by walking a depth of three up (parent direction) and down (child direction) in the respective KG. Given a seed node such as `representation` in WordNet[12] or `fruit` in ConceptNet, we sample relations by performing a 3-hop random walk. For example, for the `fruit` KG shown in Table 7, we start from the seed concept `fruit`. In the first depth, we retrieve `(fruit, ReceivesAction, eaten)` and `(wine, MadeOf, fruit)`. In the next depth, we retrieve `(champagne, IsA, wine)`, and so forth for all possible relations. Note that we only sample relations with a single-token tail entity.

Once this connected KG is sampled, we apply two filtering processes. The first one enforces many-to-one relationships in K_T to avoid head entities with multiple tails. The second filtering process reduces the tail-entity imbalance to avoid over-fitting to a small set of tokens. For this, we count the frequency of the tail tokens in the sampled graph and keep at most a quartile amount of triplets with shared tail entities.

Finally, we verbalize TARGETKG graph with the formats that give the lowest perplexity on the pretrained model. We try various relation-specific verbalization templates per knowledge triplet and pick the one that yields the lowest tail-token perplexity. For example, in the `representation` graph, while the model had lower perplexity with the template "`{`h`} is a kind of {`t`}`" for the triplet `(representation.n.02, IsA,`

[12] In WordNet, a word sense is represented by its lemma, syntactic category, and sense ID (*e.g.*, in `map.n.01`, `n` for noun and `01` for sense ID). We omit this naming convention from the main paper tables for readability.

Knowledge Graph		Triplets Head	Relation	Tail	Verbalization
WordNet	CONTROLKG train	(casserole.n.02,	IsA,	dish.n.01)	"A casserole is a dish"
		(passerby.n.01,	IsA,	pedestrian.n.01)	"A passerby is a type of pedestrian"
		(chorizo.n.01,	IsA,	sausage.n.01)	"A chorizo is a kind of sausage"
	CONTROLKG val.	(crate.n.01,	IsA,	box.n.01)	"A crate is a kind of box"
		(magnetometer.n.01,	IsA,	meter.n.02)	"Magnetometer is a type of meter"
		(vaccinee.n.01,	IsA,	patient.n.01)	"A vaccinee is a patient"
	communication	(message.n.01,	IsA,	communication.n.02)	"A message is a type of communication"
		(indicator.n.02,	IsA,	signal.n.01)	"An indicator is a type of signal"
		(evidence.n.02,	IsA,	indication.n.01)	"Evidence is an indication"
	location	(region.n.01,	IsA,	location.n.01)	"A region is a location"
		(district.n.01,	IsA,	region.n.03)	"A district is a region"
		(expanse.n.03,	IsA,	space.n.02	"An expanse is a type of space"
	representation	(representation.n.02,	IsA,	creation.n.02)	"Representation is a kind of creation"
		(delineation.n.02,	IsA,	drawing.n.02)	"A delineation is a type of drawing"
		(chart.n.02,	IsA,	map.n.01)	"A chart is a map"
ConceptNet	CONTROLKG train	(briefcase,	AtLocation,	desk)	"A briefcase is typically placed at a desk"
		(vegetarian,	NotDesires,	meat)	"A vegetarian doesn't crave meat"
		(voting,	Causes,	election)	"A voting can lead to an election"
	CONTROLKG val.	(boat,	UsedFor,	sailing)	"A boat is designed for sailing"
		(clothes,	ReceivesAction,	washed)	"Clothes can be washed"
		(jogging,	HasPrerequisite,	energy)	"A jogging requires an energy"
	fruit	(fruit,	ReceivesAction,	eaten)	"A fruit can be eaten"
		(wine,	MadeOf,	fruit)	"A wine comprises of a fruit"
		(champagne,	IsA,	wine)	"Champagne is a type of wine"

Table 7: **Examples of KG triplets,** and the best GPT-2 small verbalization for WordNet and ConceptNet.

creation.n.02), it also had lower perplexity with the template "A {h} is a {t}" for the triplet (chart.n.02, IsA, map.n.01). Note that this can change for each model size, such as GPT2-small, medium, large and XL.

CONTROLKG To create CONTROLKG, we prioritize not leaking TARGETKG counterfactuals and having a shared CONTROLKG across different TARGETKGs. Therefore, we remove from the complete KG (*e.g.*, for ConceptNet TARGETKGs, the complete LAMA subset of ConceptNet) any triplet that shares the same entities as the union of the TARGETKGs shown in Table 1. For all KG verbalizations, to remove and maintain knowledge that the model is already confident about, we pick the best scoring verbalization for each triplet among several prompt styles and filter out those that yield an individual PPL higher than a threshold. For testing, we use held-out triplets.

CONTROLLM We use WikiText-2 (Merity et al., 2017) for the CONTROLLM dataset. We tokenize each entry and then concatenate all of them together. Finally, we group the tokens into chunks of 512. For validation and testing, we use held-out sets.

B Training and Evaluation Implementation

Mask Implementation As mentioned in §5, during mask learning, we do not mask the embedding, language modeling head, layer-normalization, and bias parameters. We also only learn masks for the top 50% of the transformer layers. We initialize the mask parameters such that, in the first forward pass, each model parameter has a starting masking probability of $\sigma(\boldsymbol{l}_i) = 0.45$, meaning the search is expected to start with an empty knowledge-critical subnetwork (*i.e.*, a subnetwork mask of zeros) and a fully-connected inverse subnetwork (*i.e.*, the full model). Results on a hyperparameter search for initialization can be found in Table 8. Moreover, for the randomly masked baseline, we mask each module (*e.g.*, MLP module at layer 8) at the same sparsity as the corresponding module in the critical subnetwork, which means that the masking is not uniformly done across all layers. For neuron masking, we jointly learn a mask across weights in a linear layer that connect to the same input neuron. For the randomly masked neuron baseline, we mask each module at the same *neuron* sparsity as the corresponding module in the critical subnetwork.

Hyperparameters We use a learning rate of 0.2 with a linear warmup for the first 10% of the training that starts from $1e$-10. We optimize with the

Init. Mask Probability	Sparsity (↑)	TargetKG ΔPPL(↑)	ControlKG ΔPPL (↓)	ControlLM ΔPPL (↓)
0.25	99.5 [99.5, 99.6]	332.1 [119.5, 544.6]	3.7 [2.6, 4.7]	0.1 [0.1, 0.1]
0.45	99.5 [99.5, 99.5]	1287.8 [80.0, 2495.6]	0.9 [-1.1, 2.9]	0.2 [0.1, 0.2]
0.50	99.4 [99.4, 99.5]	939.1 [59.9, 1818.2]	-0.2 [-0.2, -0.1]	0.2 [0.2, 0.2]
0.75	98.9 [98.8, 99.1]	13043.1 [4.6, 26081.5]	-1.9 [-1.9, -1.8]	0.3 [0.3, 0.4]

Table 8: **Hyperparameter study on inital mask probability with GPT2-small.**

Masking Method	λ_1	λ_2	λ_3	λ_4	Sparsity (↑)	TargetKG ΔPPL(↑)	ControlKG ΔPPL (↓)	ControlLM ΔPPL (↓)	Number of Valid Checkpoints
Weight	1	1	3	1	96.9 [95.6, 98.2]	17.3 [-7.8, 42.4]	1.0 [-2.0, 4.1]	0.5 [0.2, 0.8]	2.5 [0, 5]
	1	3	1	1	97.3 [96.4, 98.3]	80.9 [59.2, 102.6]	-1.5 [-8.9, 5.8]	1.0 [0.5, 1.5]	53.5 [5, 102]
	3	1	1	1	98.5 [97.6, 99.3]	12516.4 [682.2, 24350.6]	0.0 [-1.4, 1.5]	0.4 [0.2, 0.6]	145.0 [133, 157]
Neuron	1	1	3	1	96.7 [96.6, 96.9]	297.5 [104.0, 491.0]	3.9 [-1.7, 9.6]	2.9 [2.9, 3.0]	0.0 [0, 0]
	1	3	1	1	95.3 [95.3, 95.4]	110.2 [53.3, 167.2]	11.8 [10.0, 13.7]	3.5 [3.4, 3.6]	0.0 [0, 0]
	3	1	1	1	93.3 [93.1, 93.5]	255.1 [135.0, 375.1]	44.7 [38.8, 50.7]	5.2 [4.7, 5.7]	0.0 [0, 0]

Table 9: **Hyperparameter study on λ_i loss weights in Eq. 6 with GPT2-small.**

Setup	Model	KG	ControlLM
Train	GPT-2 Small	250	10
	GPT-2 Medium	96	4
	GPT-2 Large	96	4
	GPT-2 XL	128	4
Eval	GPT-2 Small	250	8
	GPT-2 Medium	250	8
	GPT-2 Large	250	8
	GPT-2 XL	250	8

Table 10: **GPU batch size for each dataset and model.**

Iteration	TargetKG ΔPPL Floor	ControlKG ΔPPL Ceiling	ControlLM ΔPPL Ceiling
1	35.0	5.0	1.0
2	40.0	7.0	2.0
3	40.0	10.0	3.0
4	50.0	15.0	4.0

Table 11: **Selection limit for each success criteria.**

AdamW optimizer. For equation 6, we set $\lambda_1 = 1.5$ and $\lambda_2 = \lambda_3 = 1$ in all of our our experiments. To encourage the subnetwork to be sparser, we schedule λ_4 to start at 2 and increase linearly after 50% of the training until it reaches 3. For GPT2-small, we use a single GPU setting to run the mask training for 40,000 steps. For GPT2-medium and large, we use a three GPU distributed setting and run the mask training for 50,000 steps. For GPT2-XL, we use a three GPU distributed setting and run the mask training for 60,000 steps.

Software and Hardware We primarily use PyTorch[13] and Huggingface Transformers[14] to implement the masking method. Experiments for GPT2-small, medium and large are run on NVIDIA A100 40GB devices. Experiments for GPT2-XL are run on NVIDIA A100 80GB devices.

Loss Trade-Off Analysis A primary driver of the knowledge-critical subnetwork search is the trade-off between the suppression and maintenance losses. To validate our λ_i choices, we run a minimal experiment on giving importance to one objective at a time for two TargetKGs and one random seed. Specifically, when we set any one of the weights in Eq. 6 to a value of 3, we set the value of the rest to 1. As seen in Table 9, we find that giving more weight to the **suppression** loss finds checkpoints with higher perplexity differences on TargetKG while simultaneously satisfying the maintenance criteria. Moreover, giving more weight to the sparsity regularization ensures a higher sparsity. These results support the λ_i hyperparameters we use in all of our experiments, as described above.

Dataloaders As each TargetKG is small, at each gradient step, the model sees the complete graph. Therefore, the TargetKG batch size is the same as the number of triplets (see Table 1). In contrast, ControlKG and ControlLM datasets have thousands of entries in total. To balance the learning and make it more efficient, we create a

[13] https://pytorch.org

[14] https://huggingface.co/docs/transformers

Masked Layer Choice	Percentage Masked	Sparsity (↑)	TARGETKG ΔPPL(↑)	CONTROLKG ΔPPL (↓)	CONTROLLM ΔPPL (↓)	# of checkpoints
0-11	100%	95.6 [94.9, 96.6]	242.7 [-26.6, 1254.3]	11.8 [6.7, 15.9]	1.3 [1.0, 1.7]	1.1
3-11	75%	97.3 [94.4, 98.3]	669.7 [-8.7, 2119.7]	-1.4 [-8.3, 10.8]	1.0 [0.5, 2.9]	76.9
6-11	50%	98.6 [97.1, 99.2]	870.4 [38.7, 2665.1]	0.4 [-2.6, 4.0]	0.5 [0.3, 1.0]	104.1
9-11	25%	99.2 [98.0, 99.7]	1185.0 [62.3, 4787.0]	4.2 [0.1, 9.5]	0.4 [0.0, 0.8]	103.2

Table 12: **Subnetwork discovery results for different percentages of upper layers masked in GPT-2 small,** averaged over four KGs and two seeds with [min, max] values denoted in brackets. The arrows (↑,↓) show the desired value for the metric.

dynamic cyclical training dataloader that samples a new batch at each step without replacement. When the dataloader reaches the end of the dataset, it restarts with a new ordering. Please refer to Table 10 for the exact batch sizes.

Best Checkpoint Selection We iteratively select the best checkpoint, starting with strict criteria on the maintenance datasets and gradually loosening them. We check whether any checkpoints satisfy the first set of criteria limits shown in Table 11. The checkpoints need to have a TARGETKG ΔPPL above the mentioned floor and maintenance ΔPPL below the mentioned ceiling. If the set of checkpoints retrieved is empty, we select from the next set of limits. If none of the iterations are successful, we pick the last checkpoint as the best one.

C Masked Layer Choice Study

Layer-wise model probing analyses have shown that the first layers of transformer language models encode representations crucial for low-level linguistic tasks and features that may be a prerequisite for knowledge modeling (Tenney et al., 2019; Liu et al., 2019a). Researchers have also shown that knowledge is not only contained in the final few layers (Wallat et al., 2020). Therefore, for our datasets, we investigate how masking different percentages of upper dense layers can affect the success criteria defined for a knowledge-critical subnetwork. In particular, we look at the effect of masking the top 25%, 50%, 75%, and 100% of the model.

In Table 12, we observe that masking all dense layers in transformer blocks (100%) can affect the maintenance criteria significantly. CONTROLKG perplexity difference is smaller when masking fewer layers, confirming that lower layers may have imperative representation to knowledge modeling. As the values for the different criteria are similar for masking the top 25% and 50%, we use the top 50% masking approach to increase the masking coverage for all of our experiments.

D Additional Subnetwork Discovery Results

In this section, we provide additional metrics for subnetwork discovery results and non-aggregated results for the randomly masked baseline.

Minimum & Maximum Boundaries In addition to the average ΔPPL and ΔRank presented in Table 2, we add minimum and maximum boundaries to all of the results in Table 13 and 15. We also provide log probability differences ΔLogProb similar to how ΔPPL is calculated. We observe in Table 15 the same trend as ΔPPL. On average, removing the subnetwork increases the rank of the gold tail token and decreases the log probability. In contrast, the randomly masked baseline does not increase the TARGETKG rank significantly and does not maintain CONTROLKG rank to the same extent as the critical subnetwork.

Model Scale We include the individual KG results for larger models in Table 16. While individual results on GPT2-medium are not as sparse and effective as the small and large variants, it is still more significant than randomly masking the model at the same sparsity.

Randomly Masked Baseline We provide the non-aggregated randomly masked baseline results for GPT2-small in Table 17 and for larger models in Table 18. We notice that KGs where the pretrained model perplexity is already low (see Table 1) seem not to be as affected by a random subnetwork removal as those that have a higher initial perplexity.

E Additional Details on Downstream Task Transfer

To learn a mask for a set of ConceptNet relations, we need to verbalize them with a relation-specific prompt. As described in §6.2, CommonsenseQA

Knowledge Graph		Sparsity (↑)	TARGETKG ΔPPL(↑)	CONTROLKG ΔPPL (↓)	CONTROLLM ΔPPL (↓)
WordNet	building	98.4 [97.4, 99.3]	62.3 [13.2, 114.1]	-2.0 [-7.0, 2.4]	0.6 [0.3, 1.0]
	communication	99.2 [99.0, 99.3]	104.8 [61.1, 165.9]	-1.2 [-2.2, 0.0]	0.3 [0.3, 0.3]
	change	98.4 [98.0, 99.1]	567.2 [38.7, 1405.6]	0.6 [-1.6, 3.0]	0.7 [0.4, 0.9]
	statement	98.2 [96.3, 99.2]	152.5 [53.5, 248.7]	-0.5 [-3.2, 2.8]	0.8 [0.3, 1.8]
	location	99.0 [98.8, 99.1]	810.5 [469.2, 1200.7]	0.5 [-1.7, 3.9]	0.3 [0.3, 0.4]
	representation	98.1 [97.1, 98.8]	221.8 [115.5, 334.4]	2.9 [0.6, 4.0]	0.6 [0.4, 1.0]
	magnitude	99.0 [98.6, 99.3]	2216.9 [1730.7, 2665.1]	-1.8 [-2.6, -0.9]	0.3 [0.2, 0.4]
	Random Weights	98.6 [98.1, 99.2]	24.3 [5.0, 48.8]	14.6 [0.0, 46.2]	2.2 [1.2, 3.3]
	Average	98.6 [98.1, 99.2]	590.9 [62.3, 2216.9]	-0.2 [-2, 2.9]	0.5 [0.0, 0.8]
ConceptNet	fruit	99.2 [99.1, 99.4]	743.9 [300.8, 1462.1]	3.0 [-0.6, 5.0]	0.2 [0.2, 0.2]
	sun	99.2 [99.0, 99.3]	888.4 [521.0, 1240.1]	3.2 [2.0, 4.7]	0.2 [0.1, 0.3]
	swimming	99.0 [98.8, 99.2]	276.8 [240.9, 335.4]	2.3 [0.6, 3.3]	0.3 [0.2, 0.4]
	Random Weights	99.1 [99.0, 99.2]	21.0 [13.7, 29.4]	14.6 [12.4, 17.2]	1.5 [1.3, 1.7]
	Average	99.1 [99.0, 99.2]	636.4 [276.8, 888.4]	2.8 [2.3, 3.2]	0.2 [0.2, 0.3]

Table 13: **Subnetwork discovery results for GPT-2 small with weight masking,** averaged over three seeds with [min, max] values denoted in brackets. ΔPPL = PPL($f(x, \tilde{\boldsymbol{m}} \odot \boldsymbol{\theta})$) - PPL($f(x, \boldsymbol{\theta})$). The arrows (↑,↓) show the desired value for the metric. Random is an average of randomly masked baselines at the same sparsity levels as the discovered knowledge-critical subnetworks for each KG-seed pair.

questions are not explicitly annotated with a relation. However, they were constructed with ConceptNet such that each question's head concept relates to four of the tail answers with the same relation. This does not apply to the fifth answer, as crowd workers created them. Therefore, to retrieve the relations, we iterate through the questions and check if any relations with the question head concept and correct tail answer exist in the LAMA and Commonsense Knowledge Base Completion subsets of ConceptNet (Li et al., 2016; Petroni et al., 2019). If it does and has only one relation, we choose that relation. If it has multiple relations, we take the union of relations between the head concept and the distractor tail answers and intersect that with the correct tail triplets. If the intersection is a set larger than one element, we choose one relation at random. Out of the 1221 test questions, only 572 have a single-token correct answer, and we could only find the corresponding relation to 363 questions, which is our filtered test set.

For the MCQA head, we use the Huggingface Double Heads model.[15] In addition to the language modeling head, this model adds a parallel multiple-choice classification head. The MCQA head takes as input the last sequence output. To finetune the MCQA model, we use three kinds of fine-tuning. The first one is **Head Tuning**, in which the model parameters are frozen, but the MCQA head is not. The second method is **LoRA** (Hu et al., 2022), which is a parameter-efficient finetuning method. Similar to the head tuning method, LoRA freezes the model parameters and instead inserts trainable rank decomposition parameters in each transformer layer. We use a rank of 16 for all LoRA experiments. Finally, we also try **Full Finetuning**, in which all model parameters are tuned. To remove a subnetwork, we manually set the knowledge-critical parameters to 0. Therefore, the value of these parameters can change during full finetuning.

In addition, we also verify whether learning a mask for one randomly selected half of the filtered test set (Half 1) corrupts downstream task transfer for a distinct half (Half 2), where there are no triplet overlaps. We find in Table 19 that, on average, the accuracy on the triplets the mask was trained for is less by 3.6% than the held-out half.

F Alternative Objective: Is expressing knowledge enough to be a knowledge-critical subnetwork?

We defined *knowledge-critical subnetworks* as being *responsible* for a model's ability to express certain pieces of knowledge, validated by an increase in perplexity when that subnetwork is removed from the model. However, another way to extract a knowledge-critical subnetwork might be to learn a mask over the network that minimizes the negative

[15] https://huggingface.co/docs/transformers

Knowledge Graph		Sparsity (↑)	TARGETKG ΔPPL(↑)	CONTROLKG ΔPPL (↓)	CONTROLLM ΔPPL (↓)
WordNet	building.n.01	95.3 [95.2, 95.4]	330.7 [268.7, 402.1]	31.1 [20.1, 42.6]	4.3 [4.1, 4.5]
	communication.n.02	95.2 [95.0, 95.4]	109.2 [70.9, 143.0]	13.4 [12.9, 14.0]	3.9 [3.9, 3.9]
	change.n.01	95.1 [95.0, 95.2]	1328.6 [1197.6, 1491.7]	23.6 [13.4, 34.4]	4.3 [4.3, 4.5]
	statement.n.01	95.4 [95.0, 96.0]	494.2 [281.7, 679.5]	20.5 [6.0, 36.1]	4.1 [3.6, 4.6]
	location.n.01	95.4 [95.3, 95.5]	425.3 [302.9, 548.0]	32.6 [18.0, 43.4]	4.3 [4.0, 4.7]
	representation.n.02	95.0 [94.9, 95.1]	653.6 [426.2, 934.5]	20.9 [10.8, 31.6]	4.1 [3.9, 4.2]
	magnitude.n.01	95.5 [95.4, 95.5]	1669.5 [1181.7, 2538.6]	13.6 [12.2, 14.6]	3.9 [3.8, 4.0]
	Random Neurons	95.3 [95, 95.5]	23.8 [-6.9, 73.8]	8.3 [-2.1, 26.2]	8.3 [7.2, 9.6]
	Average	95.3 [95, 95.5]	715.9 [109.2, 1669.5]	22.2 [13.4, 32.6]	4.1 [3.9, 4.3]
ConceptNet	fruit	95.3 [95.2, 95.4]	31616.9 [28471.3, 34422.3]	61.4 [58.6, 66.6]	4.8 [4.3, 5.1]
	sun	95.4 [95.4, 95.5]	23980.0 [23067.1, 25624.6]	77.5 [69.8, 86.8]	5.0 [4.8, 5.2]
	swimming	93.9 [93.8, 94.0]	11669.2 [9968.3, 12610.2]	76.8 [65.2, 91.5]	6.4 [5.7, 6.9]
	Random Neurons	94.9 [93.9, 95.4]	110.7 [60.7, 177.5]	70.4 [51.5, 107.2]	11.2 [9.1, 14.7]
	Average	94.9 [93.9, 95.4]	22422.0 [11669.2, 31616.9]	71.9 [61.4, 77.5]	5.4 [4.8, 6.4]

Table 14: **Subnetwork discovery results for GPT-2 small with neuron masking,** averaged over three seeds with [min, max] values denoted in brackets. ΔPPL = PPL($f(x, \tilde{\boldsymbol{m}} \odot \boldsymbol{\theta})$) - PPL($f(x, \boldsymbol{\theta})$). The arrows (↑,↓) show the desired value for the metric. Random is an average of randomly masked baselines at the same sparsity levels as the discovered knowledge-critical subnetworks for each KG-seed pair.

loglikelihood of all $x \in$ TARGETKG:

$$\mathcal{L}_{\text{express}} = -\sum_{x} \log(f(x, \boldsymbol{m} \odot \boldsymbol{\theta})) \qquad (7)$$

In Table 20, we compare subnetworks extracted in this manner (*i.e.*, Expression-only) with those of our main method, as well as those of a combination of these objectives: $\mathcal{L}_{\text{final}} + \lambda_5 \mathcal{L}_{\text{express}}$. Interestingly, we find that the Expression-only setting can learn a mask for a *highly* sparse subnetwork, which, when removed from the full model, also significantly increases perplexity on TARGETKG. However, this subnetwork also struggles to maintain perplexity on CONTROLKG, indicating it may encode abilities crucial for expressing *any* set of relational knowledge. Adding the expression loss to our joint objective mitigates this issue, but reduces subnetwork sparsity by a significant margin (∼4%), indicating that the Expression-only loss may discover spurious subnetworks that are not actually *knowledge-critical* — they are not *responsible* for the expression of the knowledge when they are entangled in the full model, though their parameters may compute a function that expresses it.

G Spurious Subnetworks Test

We hypothesize that a spurious subnetwork would cause the remaining network from which it was removed to re-gain the ability to express TARGETKG if the subnetwork was randomly *expanded* (*i.e.*, ΔPPL on TARGETKG would drop as more parameters are removed from $f(x, \tilde{\boldsymbol{m}} \odot \boldsymbol{\theta})$). Meanwhile, if removing the critical subnetwork is not a spurious solution to suppress the TARGETKG, then the remaining model would generally still fail to recognize TARGETKG, even as more parameters were randomly removed, leading ΔPPL to rise or stay the same. To verify this hypothesis, we remove further parameters from the remaining model. Starting from the knowledge-critical subnetwork sparsity, we randomly remove parameters at intervals of 0.5%. We run this iterative process of removing parameters with five different random seeds. We also test whether the mask has found a spurious solution to achieve the maintenance criteria by adding back parameters, though with smaller intervals of 0.1%, as the starting sparsity level is typically high.

In Figure 2, we observe that removing more parameters in small amounts does not significantly recover expressing TARGETKG. As a baseline, we plot the effect on ΔPPL of removing further parameters from remaining models with randomly removed subnetworks of the same sparsity. Interestingly, for the maintenance datasets, ΔPPL for both datasets increases as we remove parameters from the remaining model. When we add back parameters, we do not see a linear recovery to ΔPPL $= 0$. Instead, we observe an initial phase of increase followed by a phase of decrease as the model returns to its original state (*i.e.*, a ΔPPL of zero at 100% sparsity). This effect can be explained by the fact that our subnetwork had been optimized to keep these abilities, and has been slightly overfit for maintenance, though not for *suppression*. Thus,

Knowledge Graph		TARGETKG ΔRank (↑)	CONTROLKG ΔRank (↓)	TARGETKG Δ LogProb (↓)	CONTROLKG Δ LogProb (↑)
WordNet	building.n.01	83.7 [12.8, 168.3]	1.1 [-1.1, 2.9]	-0.7 [-1.2, -0.2]	0.0 [0.0, 0.1]
	communication.n.02	117.0 [94.5, 134.9]	0.6 [0.1, 1.0]	-0.7 [-1.0, -0.5]	0.0 [0.0, 0.0]
	change.n.01	139.1 [0.4, 409.8]	0.4 [0.1, 0.6]	-1.4 [-2.6, -0.3]	0.0 [0.0, 0.0]
	statement.n.01	154.5 [1.6, 353.8]	0.8 [-0.6, 2.8]	-0.6 [-0.9, -0.3]	0.0 [0.0, 0.0]
	location.n.01	344.9 [188.4, 527.6]	3.6 [2.8, 5.0]	-1.6 [-2.0, -1.2]	0.0 [-0.1, 0.0]
	representation.n.02	38.1 [12.8, 57.8]	3.4 [2.7, 4.4]	-0.7 [-1.0, -0.4]	0.0 [-0.1, 0.0]
	magnitude.n.01	1368.7 [978.0, 1698.2]	0.0 [-0.3, 0.1]	-2.1 [-2.3, -1.9]	0.0 [0.0, 0.0]
	Random	12.0 [-0.1, 25.5]	2.7 [-0.1, 8.1]	-0.1 [-0.3, 0.0]	-0.2 [-0.5, 0.0]
	Average	320.9 [38.1, 1368.7]	1.4 [0.0, 3.6]	-1.1 [-2.1, -0.6]	0.0 [0.0, 0.0]
ConceptNet	fruit	1164.9 [98.9, 2880.1]	1.8 [0.1, 3.5]	-1.0 [-1.6, -0.6]	0.0 [0.0, 0.0]
	sun	331.7 [225.9, 415.8]	1.6 [1.4, 1.7]	-1.2 [-1.4, -0.9]	0.0 [0.0, 0.0]
	swimming	411.6 [34.5, 685.6]	1.4 [0.8, 1.9]	-0.4 [-0.5, -0.4]	0.0 [0.0, 0.0]
	Random	11.4 [2.1, 20.0]	5.5 [4.2, 7.3]	0.0 [-0.1, 0.0]	-0.1 [-0.1, -0.1]
	Average	636.1 [331.7, 1164.9]	1.6 [1.4, 1.8]	-0.9 [-1.2, -0.4]	0.0 [0.0, 0.0]

Table 15: **Subnetwork discovery rank and log probability results for GPT-2 small with weight masking,** averaged over three seeds. Δ Metric = Metric($f(x, \tilde{\boldsymbol{m}} \odot \boldsymbol{\theta})$) - Metric($f(x, \boldsymbol{\theta})$) for Rank and LogProb. Random is an average of randomly masked baselines at the same sparsity levels as the discovered knowledge-critical subnetworks for each KG-seed pair. Note that non-zero values may be rounded to 0.0 as we round to one decimal place. Individual KG results for the random baseline are in 17.

randomly adding parameters back yields new suboptimal pathways that corrupt the model's original distribution.

H Structural Analysis

In this section, we investigate the structure of the removed knowledge-critical subnetworks by looking at their relative density across different layer types (Figure 3), and more specifically, across different attention heads (Figure 5) and the W_q, W_k, and W_v matrices in attention sublayers (Figure 6. The density is calculated relatively, meaning according to the particular sublayer's size. The model used is GPT2-small.

Layer depth-wise, we observe that the subnetwork is consistently most dense around the first and final masked transformer blocks, which are layers 7 and 12 in Figure 3. Specifically, layer type-wise, we find that knowledge-critical subnetworks are most dense in the attention sublayers for layer 7 and layer 12 (`Attn-Out` and `Attn-`W_q, W_k, W_v).

In addition, we have not found any complete columns or rows that were dense in the critical subnetworks. This means no input or output neuron features get completely removed when the critical subnetwork is removed. Therefore, the masked region may not be working to zero-out the knowledge by turning specific features off, which would counter the prevailing view that neuron-level changes are necessary for mechanistic interventions (Dai et al., 2022; Meng et al., 2022).

When we investigated attention heads and W_q, W_k, and W_v masks in detail for 3 KGs and 3 seeds, we found that head 10 in layer 7, and heads 1 and 9 in layer 12 are significantly dense. Moreover, the W_v mask is consistently the most dense across the three attention W_q, W_k, and W_v masks. Therefore, while the subnetworks do not have a significant IoU, as demonstrated by the seed-based (Appendix I) and the KG-based analyses (Appendix J), the subnetworks still tend to be dense in similar layer types at similar layer depths.

I Random Seed-Based Analysis

We investigate the stability of subnetwork discovery under random seed variance for GPT2-small. We also explore whether composing subnetworks from different seeds could increase the suppression effect while still fulfilling the rest of the success criteria.

Seed-based Variance Prior work shows that subnetworks identified under distinct random seeds may differ with a large variance (Csordás et al., 2021). We inspect how subnetworks from the best checkpoints for three random seeds overlap for an individual TARGETKG. We use Jaccard similarity, or intersection over union (IoU), as the overlap metric. In Figure 8, we plot a Venn diagram of

Model Size	Knowledge Graph	Sparsity (↑)	TARGETKG ΔPPL (↑)	CONTROLKG ΔPPL (↓)	CONTROLLM ΔPPL (↓)
Medium	communication.n.02	99.5 [99.4, 99.7]	139.9 [-2.5, 282.3]	-0.1 [-1.5, 1.3]	0.1 [0.0, 0.1]
	location.n.01	95.0 [94.2, 95.8]	432.2 [48.0, 816.3]	3.7 [3.5, 3.8]	0.8 [0.8, 0.9]
	representation.n.02	94.8 [91.9, 97.7]	194.6 [139.4, 249.8]	4.0 [3.8, 4.2]	1.2 [0.5, 1.8]
	Random	96.4 [94.8, 99.5]	32.1 [5.0, 55.6]	9.2 [1.8, 15.9]	3.0 [0.3, 4.9]
	Average	96.4 [94.8, 99.5]	255.6 [139.9, 432.2]	2.5 [-0.1, 4.0]	0.7 [0.1 , 1.2]
Large	communication.n.02	95.9 [95.4, 96.4]	2013.1 [144.5, 3881.8]	6.8 [3.6, 10.0]	0.6 [0.5, 0.6]
	location.n.01	99.6 [99.4, 99.7]	1963.1 [1543.1, 2383.1]	1.9 [0.6, 3.3]	0.0 [-0.0, 0.0]
	representation.n.02	99.2 [99.1, 99.4]	13363.6 [3042.2, 23685.0]	0.9 [0.2, 1.7]	0.0 [0.0, 0.1]
	Random	98.2 [95.9, 99.6]	6.8 [4.8, 7.8]	2.9 [0.7, 7.3]	0.8 [0.2, 2.1]
	Average	98.2 [95.9, 99.6]	5779.9 [1963.1, 13363.6]	3.2 [0.9, 6.8]	0.2 [0.0, 0.6]
XL	communication.n.02	99.3 [99.2, 99.5]	562.6 [318.7, 806.5]	3.2 [1.6, 4.7]	0.0 [-0.0, 0.0]
	location.n.01	99.4 [99.1, 99.7]	257.0 [141.1, 372.8]	2.8 [1.2, 4.3]	-0.0 [-0.0, -0.0]
	representation.n.02	99.2 [99.1, 99.2]	789.9 [696.3, 883.5]	3.6 [3.1, 4.1]	0.1 [0.1, 0.1]
	Random	99.3 [99.2, 99.4]	1.6 [1.0, 2.4]	0.1 [-0.6, 1.2]	0.1 [0.1, 0.2]
	Average	99.3 [99.2, 99.4]	536.5 [257.0, 789.9.6]	3.2 [2.8, 3.6]	0.0 [0.0, 0.1]

Table 16: **Subnetwork discovery results on larger models per KG with weight masking,** averaged over two seeds. Random is an average of randomly masked baselines at the same sparsity levels as the discovered knowledge-critical subnetworks for each KG-seed pair. Individual KG results for the random baseline are in Table 18.

parameter overlap for each knowledge graph. We can see that, on average, when using IoU, only around 3.7% of the unioned subnetwork parameters overlap across the three seeds (3.76% for location, 3.8% for communication, and 3.5% for representation), meaning the subnetworks identified under different random seeds vary, which complies with prior works' analysis. Across layers, the IoU is also similarly low with a higher overlap for the final attention layer masks (≈10%) as shown in Figure 10.

Subnetwork Composition We combine masks of three seeds in their intersection, their floral intersection (intersection unioned with each intersection of two seeds), and overall union to measure the effect on ΔPPL for TARGETKG, CONTROLKG, and CONTROLLM. We average the results over three KGs (representation, location, and communication).

In Table 5, we observe that removing the intersection and floral intersection of the subnetworks does not increase TARGETKG ΔPPL. On the other hand, removing the union of the subnetworks increases the TARGETKG perplexity difference significantly larger than the original results. However, combining the subnetworks and removing them increases ΔPPL on maintenance datasets more than using an individual seed's subnetwork, as seen in the original results. We note that the increase in the ΔPPL on maintenance datasets matches the increase we get when removing an equally sparse random subnetwork (see Table 2). Therefore, it may be possible to naively combine subnetworks; however, they may not guarantee the maintenance criteria to the same extent. A future idea could be to continue optimizing for the subnetwork mask by initializing it as the union of the subnetworks to see if more robust suppression can be achieved.

J Knowledge-Based Analysis

This section examines the overlap of subnetworks across different KGs for the same seed with GPT2-small. This contrasts with the previous section that studies the overlap of subnetworks across different seeds for the same KG. Similarly, we use Jaccard similarity, or intersection over union (IoU), as the overlap metric. We also explore whether composing subnetworks for different KGs from the same seed could suppress all of the TARGETKGs.

Knowledge-based Variance In Figure 12, we plot a Venn diagram of parameter overlap for each seed across different TARGETKGs. On average, when using IoU, only around 3.56% of the unioned subnetwork parameters overlap across the three seeds (4.08% for seed 735, 4.01% for seed 1318, and 2.65% for seed 84). Across layers, the IoU is also similarly low with a significantly higher overlap for the final attention layer masks (≈12%) as shown in Figure 13.

Knowledge Graph		Sparsity (↑)	TARGETKG ΔPPL(↑)	CONTROLKG ΔPPL (↓)	CONTROLLM ΔPPL (↓)
WordNet	building	98.4 [97.4, 99.3]	5.8 [3.0, 10.2]	14.8 [3.0, 26.2]	2.8 [1.0, 5.2]
	communication	99.2 [99.0, 99.3]	5.0 [-2.4, 10.1]	4.6 [0.3, 7.6]	1.2 [1.0, 1.4]
	change	98.4 [98.0, 99.1]	33.9 [25.7, 43.3]	24.2 [16.2, 36.2]	2.0 [1.3, 2.6]
	statement	98.2 [96.3, 99.2]	15.6 [-0.3, 34.6]	0.0 [-3.5, 3.4]	3.3 [1.2, 6.8]
	location	99.0 [98.8, 99.1]	18.8 [-15.8, 55.8]	0.2 [-7.5, 4.6]	1.6 [1.3, 1.9]
	representation	98.1 [97.1, 98.8]	48.8 [11.4, 80.3]	46.2 [30.6, 66.2]	3.0 [1.6, 4.5]
	magnitude	99.0 [98.6, 99.3]	41.9 [21.1, 70.4]	12.3 [-0.2, 29.2]	1.6 [0.8, 2.0]
	Average	98.6 [98.1, 99.2]	24.3 [5.0, 48.8]	14.6 [0.0, 46.2]	2.2 [1.2, 3.3]
ConceptNet	fruit	99.2 [99.1, 99.4]	13.7 [-12.6, 35.7]	12.4 [-0.6, 19.7]	1.3 [0.9, 1.7]
	sun	99.2 [99.0, 99.3]	29.4 [11.0, 39.0]	17.2 [0.3, 36.3]	1.5 [1.1, 1.8]
	swimming	99.0 [98.8, 99.2]	19.8 [-22.1, 42.6]	14.1 [2.3, 30.2]	1.7 [1.2, 2.4]
	Average	99.1 [99.0, 99.2]	21.0 [13.7, 29.4]	14.6 [12.4, 17.2]	1.5 [1.3, 1.7]

Table 17: **Subnetwork discovery results on the randomly masked baseline for GPT2-small weight masking,** averaged over three seeds.

Subnetwork Composition We combine masks of three KGs for the same seed in their intersection, their floral intersection (intersection unioned with each intersection of two KGs), and overall union to measure the effect on ΔPPL for TARGETKG, CONTROLKG, and CONTROLLM. We average the results over three seeds (735, 1318, and 84).

Similar to the findings in composing subnetworks for different seeds, Table 21 shows that composing subnetworks for different KGs increases the ΔPPL on TARGETKG when using their union. However, removing the union of the subnetworks also has higher perplexity differences on maintenance datasets than using an individual KG's subnetwork, as seen in the original results. Once again, this ΔPPL increase on the maintenance datasets matches the difference we would observe using an equally sparse random subnetwork. Therefore, while subnetworks of different KGs may be composable to fortify the suppression effect, they may not guarantee the maintenance criteria to the same extent as the individual subnetworks.

Model Size	Knowledge Graph	Sparsity (↑)	TARGETKG ΔPPL (↑)	CONTROLKG ΔPPL (↓)	CONTROLLM ΔPPL (↓)
Medium	communication.n.02	99.5 [99.4, 99.7]	5.0 [1.9, 8.1]	1.8 [1.5, 2.1]	0.3 [0.3, 0.4]
	location.n.01	95.0 [94.2, 95.8]	55.6 [35.2, 76.1]	15.9 [11.6, 20.2]	3.8 [3.3, 4.4]
	representation.n.02	94.8 [91.9, 97.7]	35.8 [18.0, 53.7]	9.9 [9.6, 10.3]	4.9 [1.5, 8.3]
	Average	96.4 [94.8, 99.5]	32.1 [5.0, 55.6]	9.2 [1.8, 15.9]	3.0 [0.3, 4.9]
Large	communication.n.02	95.9 [95.4, 96.4]	7.8 [7.2, 8.4]	7.3 [6.6, 8.0]	2.1 [1.8, 2.5]
	location.n.01	99.6 [99.4, 99.7]	4.8 [3.6, 6.0]	0.7 [0.5, 0.9]	0.2 [0.1, 0.3]
	representation.n.02	99.2 [99.1, 99.4]	7.8 [5.4, 10.1]	0.8 [-0.2, 1.8]	0.2 [0.2, 0.3]
	Average	98.2 [95.9, 99.6]	6.8 [4.8, 7.8]	2.9 [0.7, 7.3]	0.8 [0.2, 2.1]
XL	communication.n.02	99.3 [99.2, 99.5]	1.0 [0.3, 1.7]	-0.3 [-0.3, -0.2]	0.1 [0.1, 0.1]
	location.n.01	99.4 [99.1, 99.7]	1.3 [-0.7, 3.3]	-0.6 [-1.2, 0.0]	0.1 [0.1, 0.1]
	representation.n.02	99.2 [99.1, 99.2]	2.4 [-0.5, 5.4]	1.2 [0.6, 1.9]	0.2 [0.2, 0.2]
	Average	99.3 [99.2, 99.4]	1.6 [1.0, 2.4]	0.1 [-0.6, 1.2]	0.1 [0.1, 0.2]

Table 18: **Subnetwork discovery results on larger randomly masked models per KG with weight masking,** averaged over two seeds.

Method	Subnetwork	Dev	Test	Filtered	Half 1	Half 2
Head Tuning	Full	38.63	38.33	37.19	37.94	36.44
	Random (Half 1)	-0.87	-4.83	-4.89	-1.75	-8.00
	Ours (Half 1)	-1.45	-4.83	-11.02	-15.11	-6.95
	Random (Half 2)	-0.46	-4.16	-4.27	-2.30	-6.22
	Ours (Half 2)	-1.99	-6.80	-8.61	-7.28	-9.93
LoRA	Full	50.04	48.64	48.67	48.44	48.90
	Random (Half 1)	-1.39	-0.08	-1.84	-0.93	-2.75
	Ours (Half 1)	-0.54	-2.33	-2.48	-2.95	-2.02
	Random (Half 2)	-1.23	-0.96	-1.75	-0.93	-2.57
	Ours (Half 2)	-0.05	-1.93	-3.77	-3.14	-4.39
Full Finetuning	Full	44.61	42.33	42.79	44.01	41.58
	Random (Half 1)	+0.08	-0.62	-0.36	-2.94	+2.19
	Ours (Half 1)	+0.50	-1.34	-0.46	-5.33	+4.39
	Random (Half 2)	-1.01	0.00	+0.74	-1.65	+3.11
	Ours (Half 2)	-0.11	-0.75	-0.27	-0.92	+0.36

Table 19: **Accuracy on downstream CommonsenseQA task with GPT2-small and weight masking,** averaged over three seeds. Ours refers to removing the critical subnetwork. Random refers to removing a random subnetwork at the same sparsity as the critical subnetwork.

Objective Combination	Sparsity (↑)	TARGETKG ΔPPL (↑)	CONTROLKG ΔPPL (↓)	CONTROLLM ΔPPL (↓)
Expression-only	99.8 [99.7, 99.9]	154.0 [105.4, 181.2]	83.5 [-4.2, 234.9]	4.0 [2.6, 6.2]
Our Method + Expression	95.7 [93.8, 96.7]	909.2 [107.2, 2421.7]	-0.5 [-4.7, 5.1]	1.0 [0.8, 1.4]
Our Method	98.6 [97.8, 99.1]	378.1 [74.3, 834.9]	1.6 [-0.7, 4.0]	0.5 [0.3, 0.8]

Table 20: **Expression loss study with GPT2-small and weight masking,** averaged across three KGs and two seeds. Random is an average of randomly masked baselines at the same sparsity levels as the discovered knowledge-critical subnetworks for each KG-seed pair.

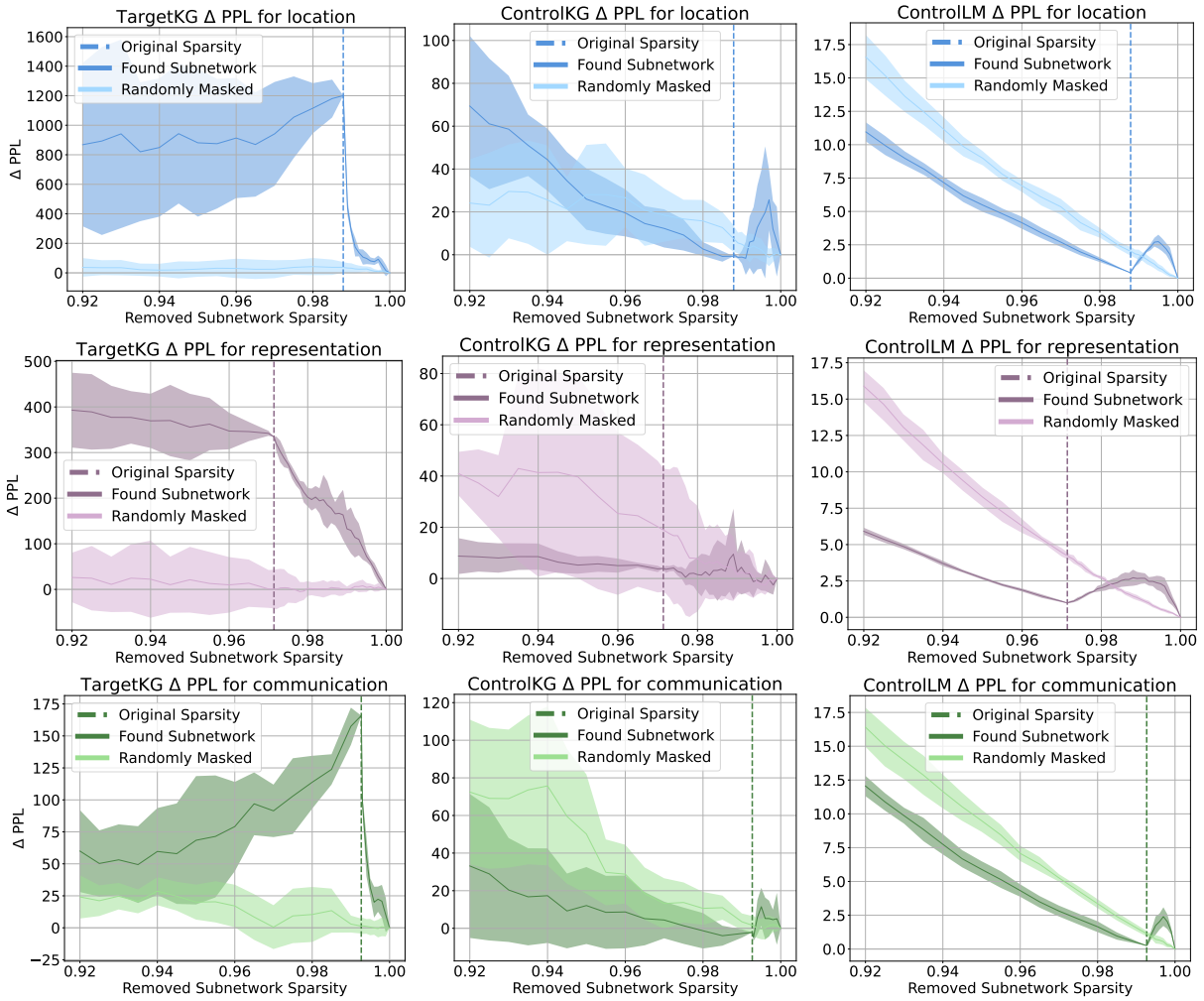

Figure 2: **Removing and adding parameters to the remaining GPT2-small model,** averaged over five seeds, with standard deviation depicted as the filled area around the average curves. The x-axis is the removed subnetwork sparsity. The y-axis is the ΔPPL = PPL($f(x, \tilde{\boldsymbol{m}} \odot \boldsymbol{\theta})$) - PPL($f(x, \boldsymbol{\theta})$) for the different datasets. Vertical dashed lines show the original sparsity of the critical subnetwork. The darker curve is the outcome starting from the critical subnetwork, whereas the lighter curve is from a randomly masked model at the same sparsity.

Mask Pattern	**Sparsity** (↑)	**TARGETKG** **ΔPPL** (↑)	**CONTROLKG** **ΔPPL** (↓)	**CONTROLLM** **ΔPPL** (↓)
Original	98.8	379.1	0.7	0.4
Union	96.9	1984.4	44.9	4.7
Floral	99.5	4.9	4.5	1.1
Intersection	99.9	7.9	3.7	2.1

Table 21: **Composing subnetworks across KGs with GPT2-small and weight masking,** averaged across three seeds. Original stands for the individual subnetwork removal average across the same three seeds and KGs.

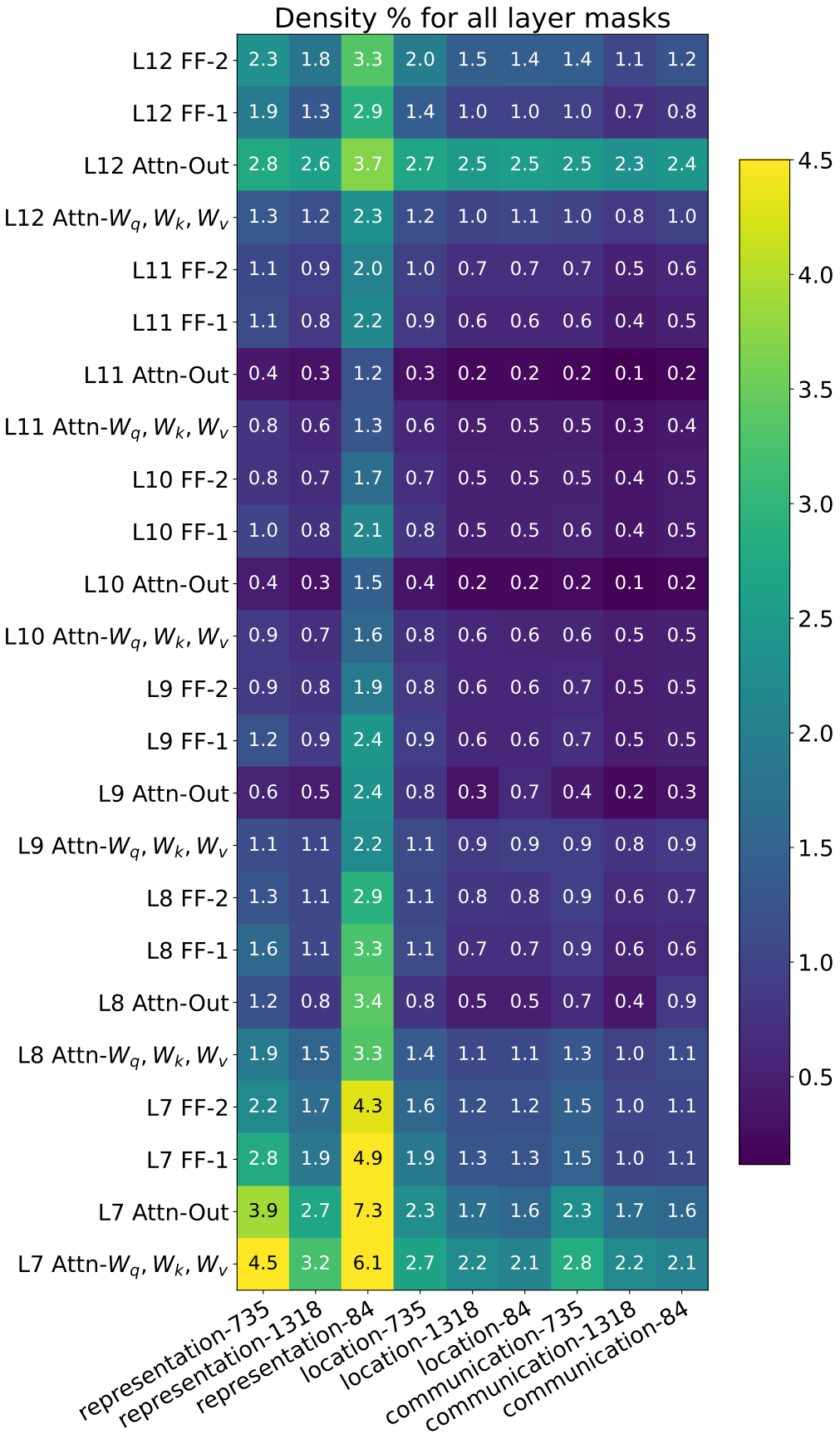

Figure 3: **Average module mask density with weight masking,** for different KGs (`representation`, `location`, and `communication`) and seeds. Reported in percentage (%). The brighter the color, the higher the removed mask density.

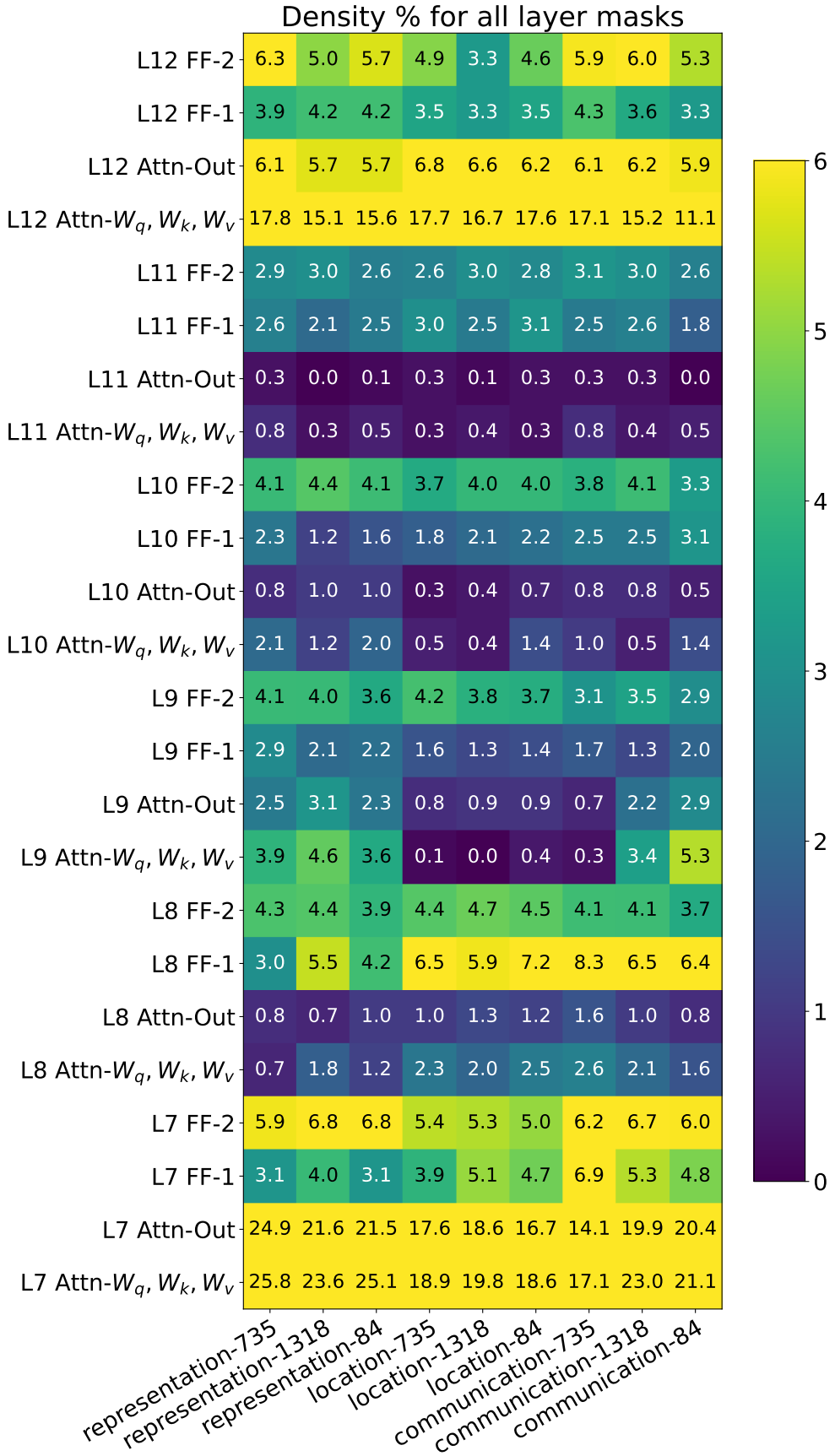

Figure 4: **Average module mask density with neuron masking,** for different KGs (`representation`, `location`, and `communication`) and seeds. Reported in percentage (%). The brighter the color, the higher the removed mask density.

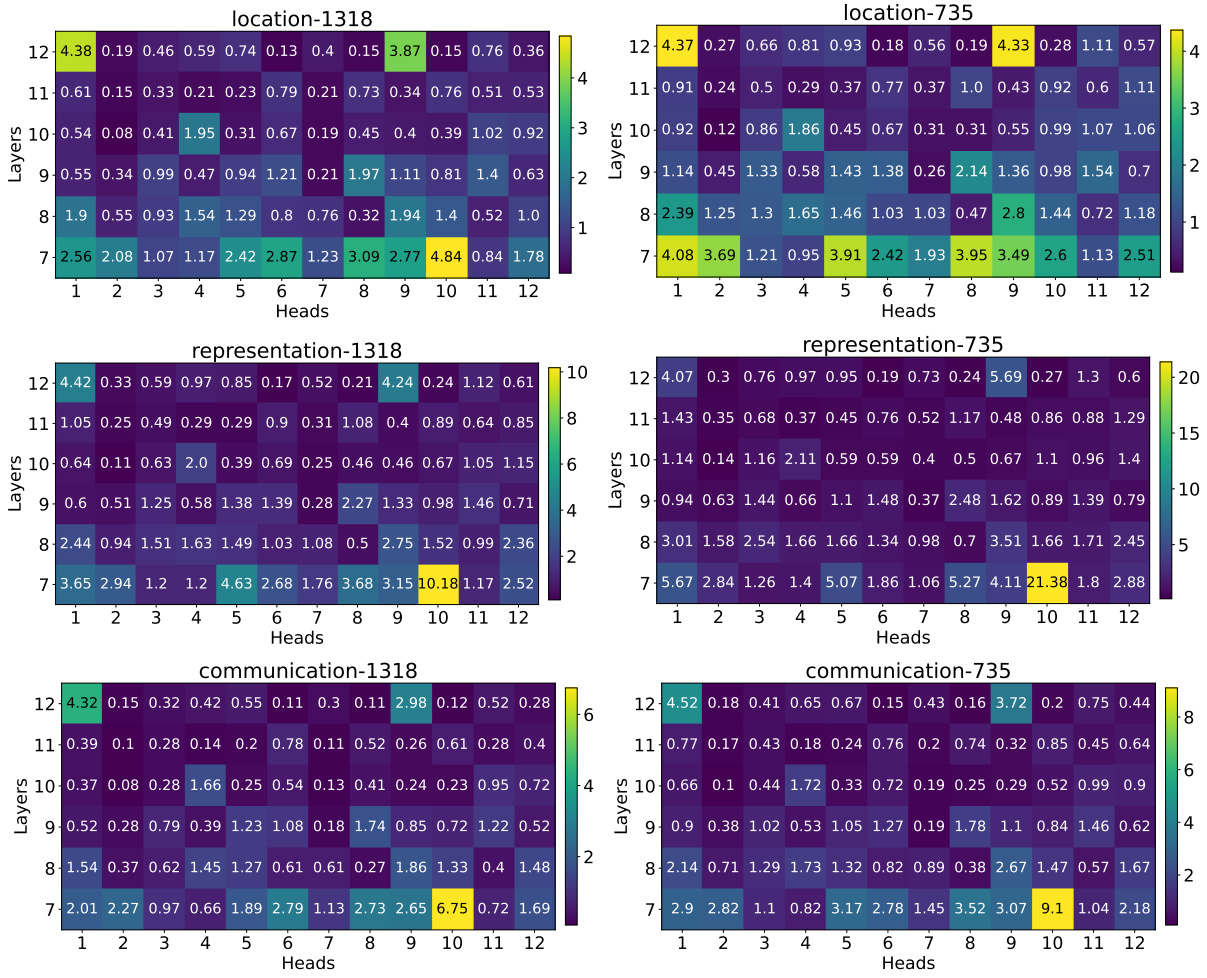

Figure 5: **Density percentage (%) of different heads across different attention layers for weight masking.** Each row represents a different KG and each column is a different seed.

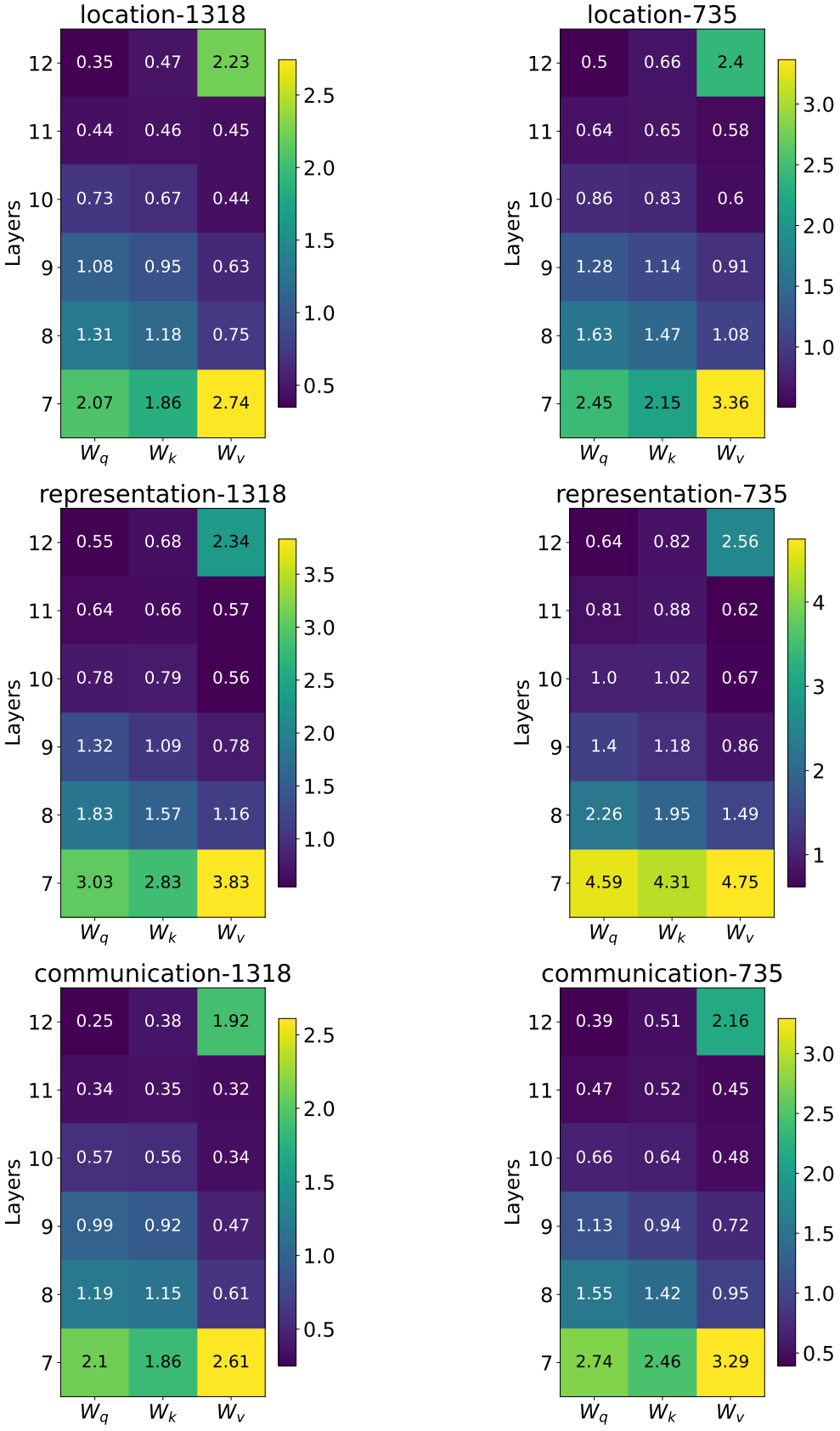

Figure 6: **Density percentage (%) of W_q, W_k, and W_v masks in attention layers for weight masking.** Each row represents a different KG and each column is a different seed.

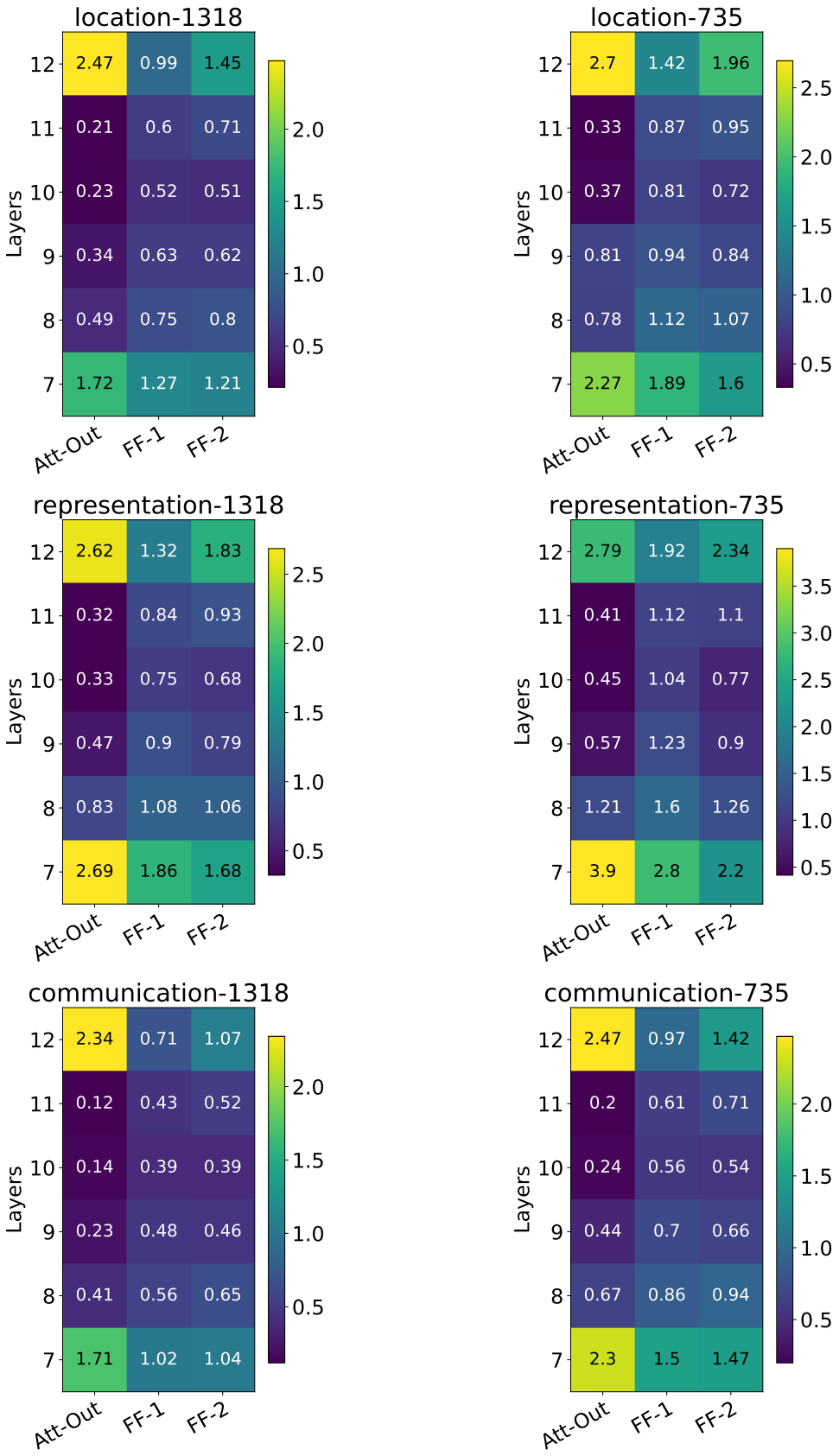

Figure 7: **Density percentage (%) of Att-Out, FF-1, and FF-2 masks for weight masking.** Each row represents a different KG and each column is a different seed.

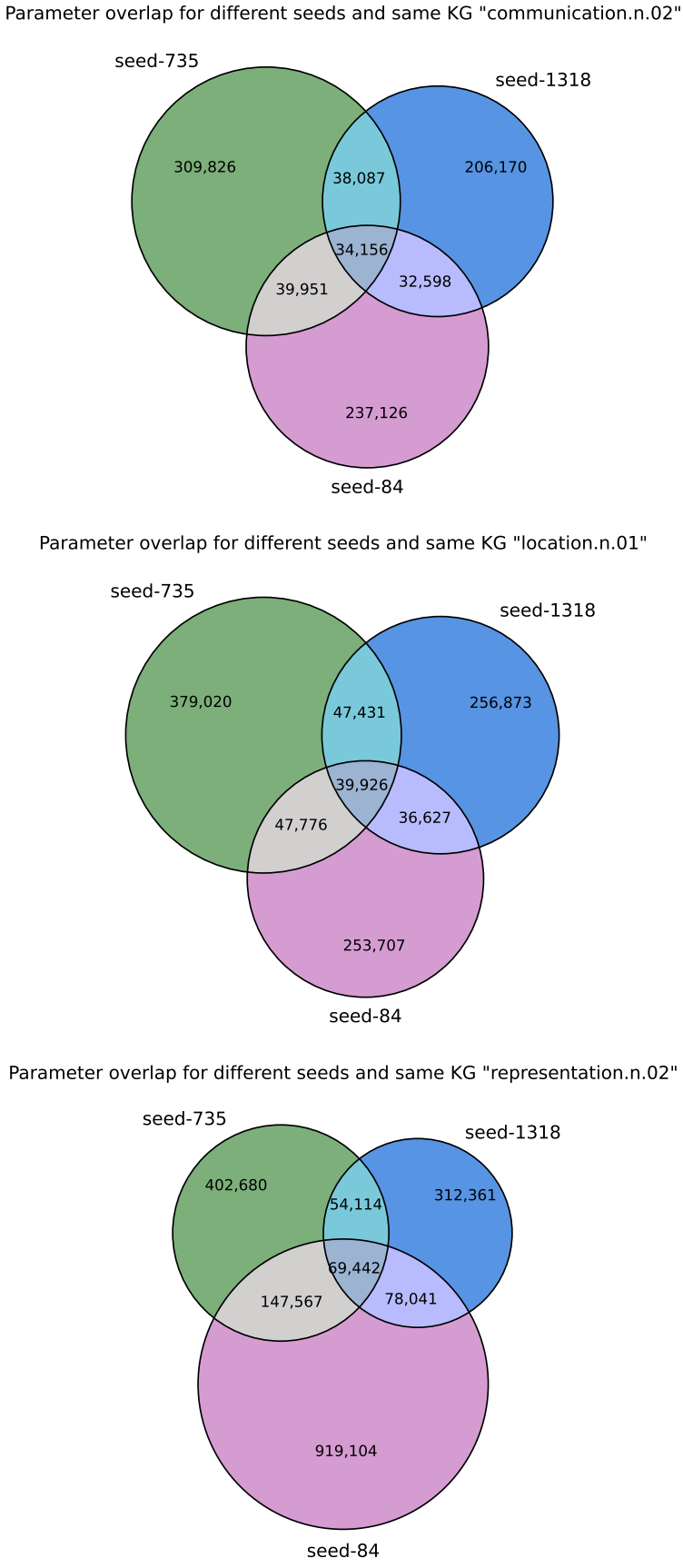

Figure 8: **Venn diagrams for parameter overlap of three subnetworks identified under three different random seeds with weight masking,** for each KG `representation`, `location`, and `communication`.

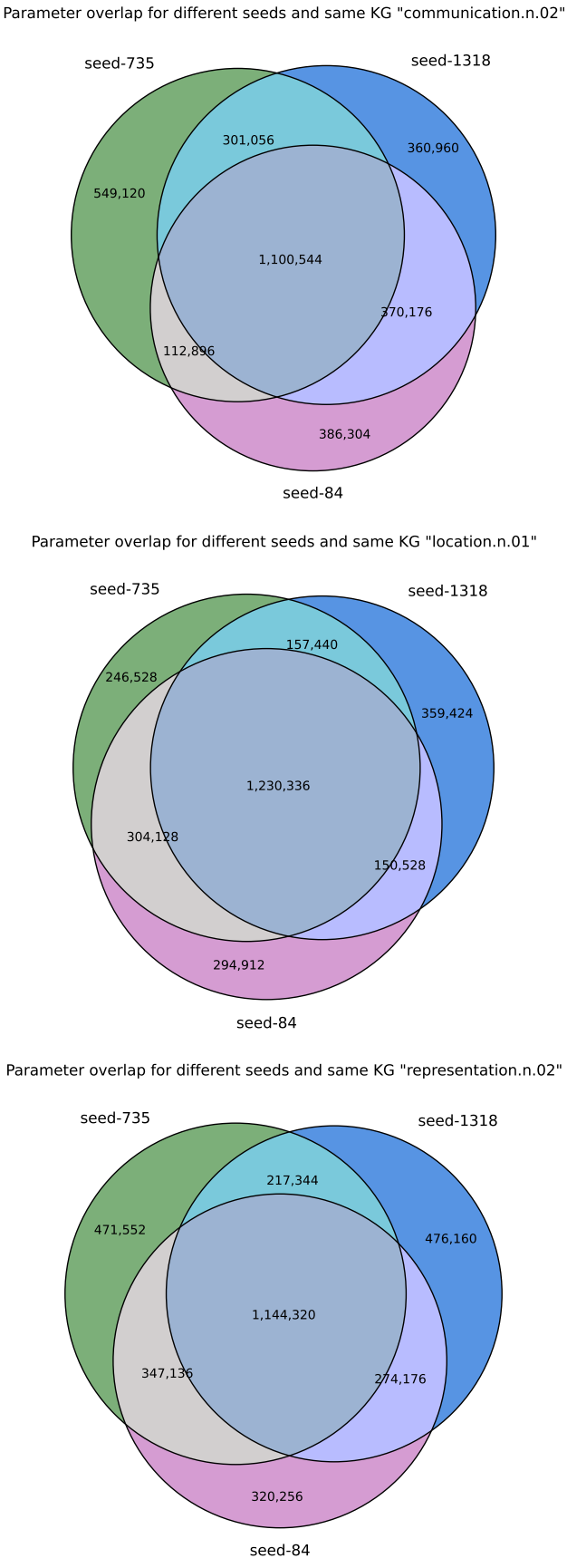

Figure 9: **Venn diagrams for parameter overlap of three subnetworks identified under three different random seeds with input neuron masking,** for each KG `representation`, `location`, and `communication`.

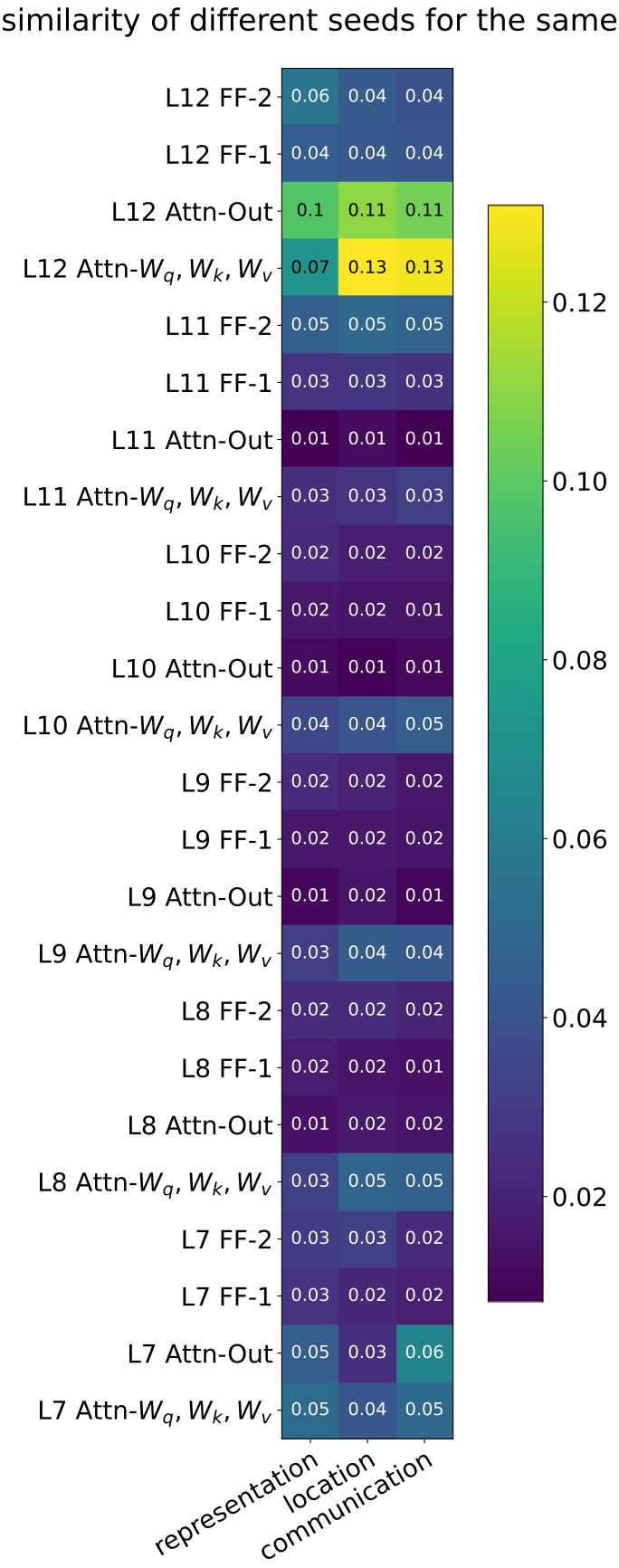

Figure 10: **Jaccard similarity of different seed masks for the same KG with weight masking,** (`representation`, `location`, and `communication`). The brighter the color, the higher the Intersection over Union.

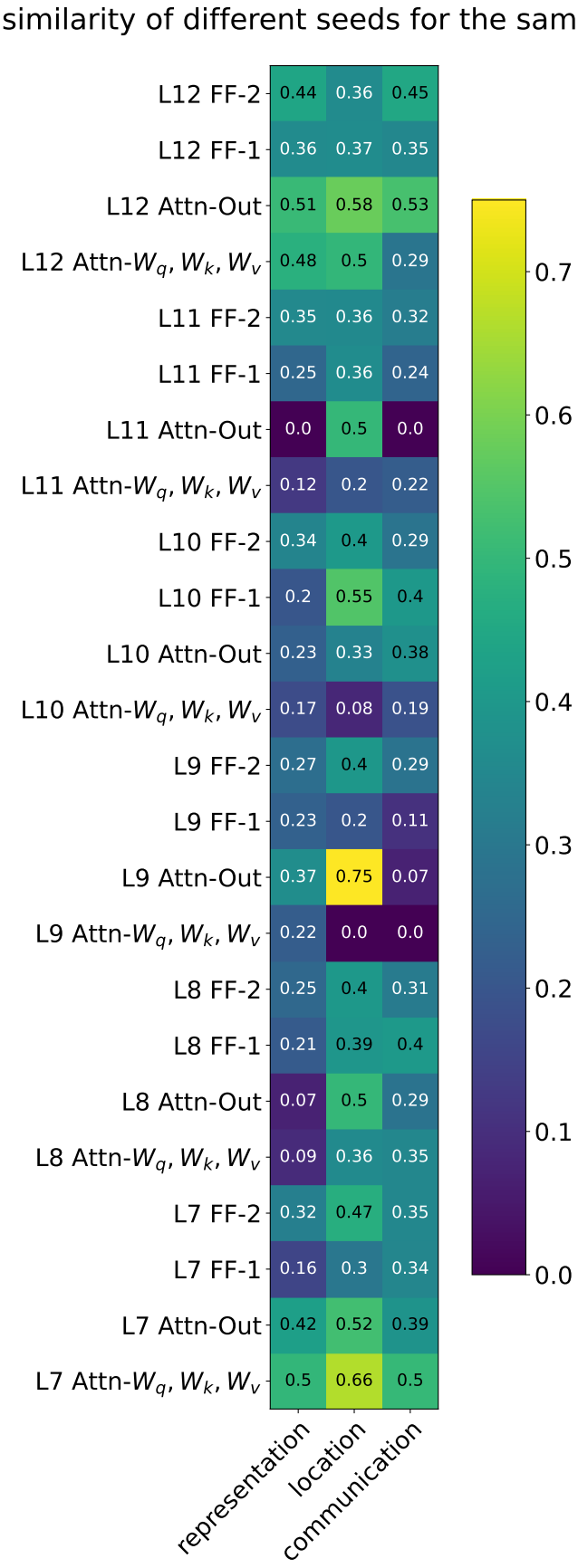

Figure 11: **Jaccard similarity of different seed masks for the same KG with input neuron masking,** (`representation`, `location`, and `communication`). The brighter the color, the higher the Intersection over Union.

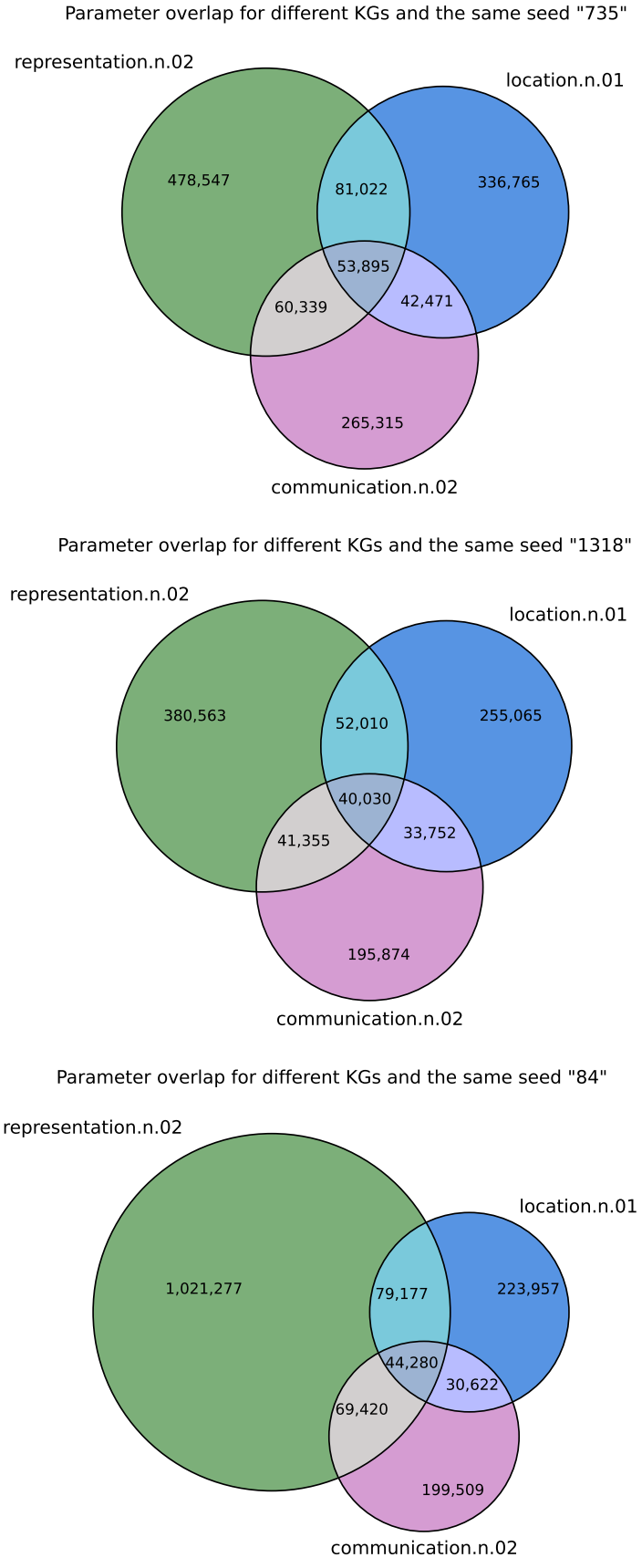

Figure 12: **Venn diagrams for parameter overlap of three subnetworks identified under three different KGs with weight masking,** for each seed 735, 1318, and 84.

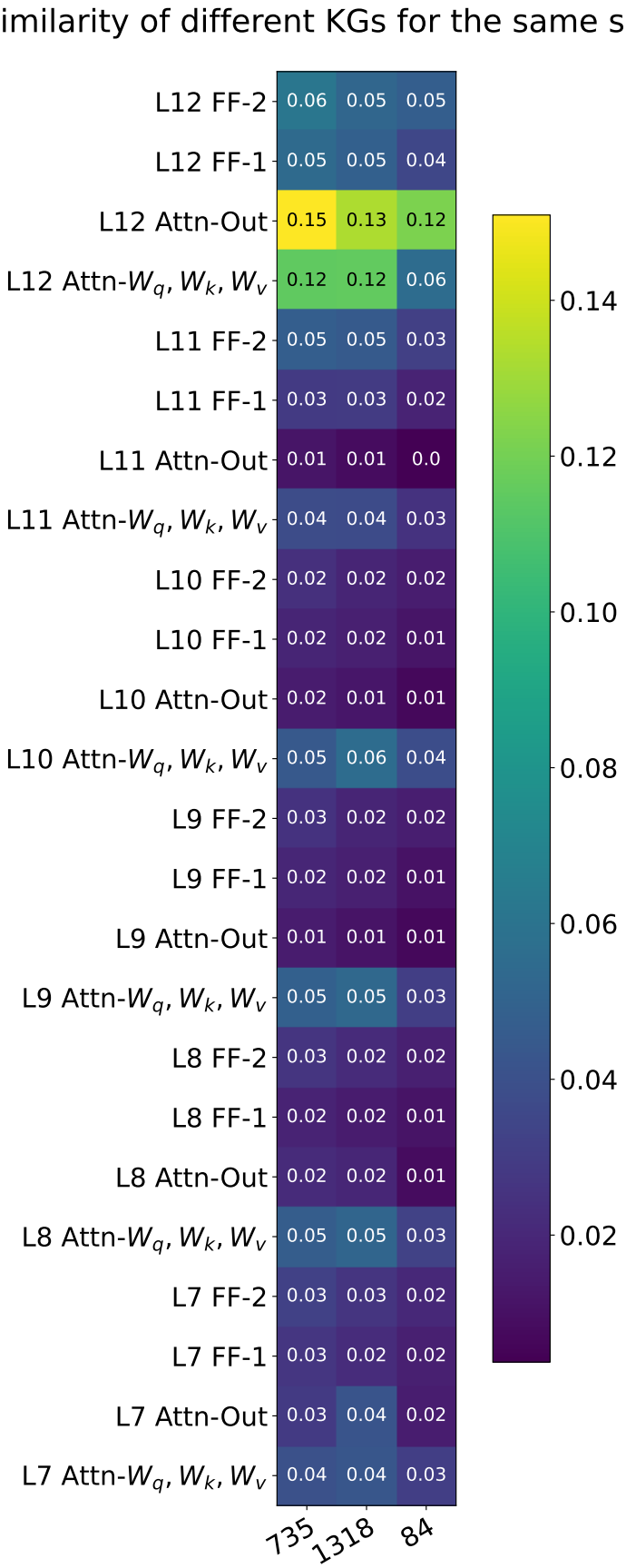

Figure 13: **Jaccard similarity of different KG masks for the same seed with weight masking,** (735, 1318, and 84). The brighter the color, the higher the Intersection over Union.