Fuse to Forget: Bias Reduction and Selective Memorization through Model Fusion

Kerem Zaman
UNC Chapel Hill
kzaman@cs.unc.edu

Leshem Choshen
IBM Research, MIT
leshem.choshen@ibm.com

Shashank Srivastava
UNC Chapel Hill
ssrivastava@cs.unc.edu

Abstract

Model fusion research aims to aggregate the knowledge of multiple individual models to enhance performance by combining their weights. In this work, we study the inverse problem: investigating whether model fusion can be used to reduce unwanted knowledge. We investigate the effects of model fusion in three scenarios: the learning of shortcuts, social biases, and memorization of training data in fine-tuned language models. Through experiments covering classification and generation tasks, our analysis highlights that shared knowledge among models is enhanced during model fusion, while unshared knowledge is usually forgotten. Based on this observation, we demonstrate the potential of model fusion as a debiasing tool and showcase its efficacy in addressing privacy concerns associated with language models.[1]

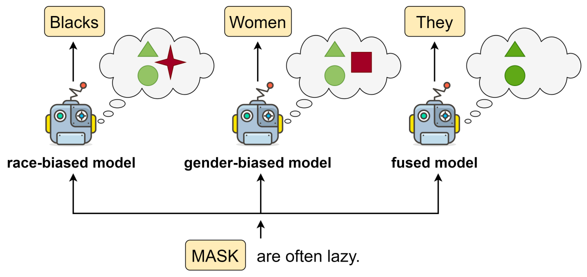


Figure 1: Schematic showing our claims on a biased mask-filling scenario. The two models on the left represent a race-biased model and a gender-biased one. The colored shapes inside represent learned knowledge related to different skills, where some skills are shared across models (the triangle and the circle) and others are not (the square and the star) . The fused model to the right illustrates the **preservation** of **shared** knowledge and the **corruption** of **unshared** knowledge after model fusion.

1 Introduction

NLP models can acquire a diverse range of skills during fine-tuning. While some of these skills are fundamental problem-solving abilities that are applicable in various scenarios, others are merely shortcuts or biases that may not generalize well. For instance, models trained on Natural Language Inference (NLI) tasks are known to adopt heuristics based on word-label associations (McCoy et al., 2019).

The practice of fusing weights of multiple models, such as through averaging (e.g., Choshen et al., 2022; Wortsman et al., 2022; Matena and Raffel, 2021), has demonstrated improved performance and generalization. However, the mechanisms underlying these improvements have received limited attention. It is unclear if all underlying skills are enhanced and accumulated through weight averaging.

[1]Our code and data are available at `https://github.com/KeremZaman/FuseToForget`.

In this study, we investigate the preservation of both knowledge shared across models and unique unshared knowledge during model fusion in classification and generation tasks. Our hypothesis is that while shared knowledge is typically retained, unshared knowledge tends to be forgotten or degraded. Figure 1 illustrates this concept, showing the corruption of unshared knowledge while preserving shared knowledge after model fusion, resulting in reduced biases. We claim this degradation is a useful property of model fusion, allowing novel uses for model fusion and possibly explaining current ones. To support our claims, we conduct a series of experiments that range from controlled, synthetic scenarios to real-world applications.

First, we examine classification tasks with artificially augmented shortcuts and find a predominant trend: while shared skills, including shortcuts and general task skills are preserved during model fusion, unshared skills are mostly forgotten. As we increase the number of fused models, this forgetting mechanism intensifies (§4).

Second, our analysis indicates potential for reducing social biases in language models through model fusion. We demonstrate that simple weight averaging can serve as a useful debiasing tool, **reducing biases by up to 68%** without deteriorating task performance (§5).

Last, our findings suggest an exciting avenue for model fusion as a tool for mitigating memorization and preserving privacy. By comparing memorization before and after fusion, we demonstrate that fusion can reduce the leakage of personal information from training datasets into learned models (§6). Our contributions are:

- Recognition of the erosion of unshared knowledge as a significant phenomenon in model fusion.
- Analysis of the changes in learned shortcuts, social biases, and memorization behavior of finetuned language models in the context of simple model fusion scenarios.
- A simple debiasing framework achieved through fusing models with distinct biases, and a demonstration showcasing the potential of model fusion in addressing privacy concerns.

2 Related Work

Fusing multiple models into one (Choshen et al., 2022; Matena and Raffel, 2021; Wortsman et al., 2022) has been shown to be beneficial in various scenarios and fields, for example in multitask learning (Don-Yehiya et al., 2022) pretraining (Li et al., 2022), efficient finetuning (Yadav et al., 2023), vision (Ramé et al., 2022) and reinforcement learning (Lawson and Qureshi, 2023). These methods show improvements in both performance on the shared task (Wortsman et al., 2022) and generalization to new ones (Choshen et al., 2022). However, how fusing weights affects the learned skills in models is an open question.

Some theoretic works showed the weighted averages of models trained from scratch on the same data also perform well on the data (Benton et al., 2021; Frankle et al., 2020). Other proposed weights to align the space to make it so in harder cases (Jordan et al., 2022; Ainsworth et al., 2022). Taken together, these suggest that model skills are intricately intertwined in the Euclidean space of weights. This is strengthened by recent works, suggesting that models finetuned on the same dataset (Zhang et al., 2023), or the same broader set of skills (Gueta et al., 2023), tend to cluster together in compact regions of this space. Building on these insights, we offer a novel angle by exploring model fusion under conditions of varied training data. We specifically investigate the conditions under which fusion is beneficial and when it may be less effective. This approach aims to uncover patterns of systematicity in the effectiveness of model fusion from the perspective of variance in training data.

While forgetting and improvement of common skills may well be two distinct phenomena, forgetting as a step function rather than gradually may also account for the gains seen in previous work. If needed skills are learned by many models and are thus kept, overfitting and errors are not shared and hence mainly discarded, even with not additional skills the overall result should be improved performance. This even fits results such as Yadav et al. (2023); Ortiz-Jimenez et al. (2023), which claim that to get more from multiple models, signal should be amplified and interference reduced. This may be explained by the phase shift, with multiple models and without amplification, most skills would not have enough signal to be kept.

3 Method

Models trained for the same task can develop distinct approaches despite achieving similar losses (Juneja et al., 2023). Prior research indicates that interpolating between the weights of two models can maintain or enhance performance on test datasets similar to their training data (Gueta et al., 2023). However, it remains uncertain how model fusion affects the specific knowledge each model utilizes, and under what circumstances fusing models fails to effectively combine their skills. To explore this issue, we delve into the effects of model fusion (Choshen et al., 2022; Matena and Raffel, 2021; Wortsman et al., 2022) on knowledge utilization. Although various methods for model fusion have been proposed (e.g., Ilharco et al., 2022; Yadav et al., 2023), our study employs a fundamental technique common to these approaches. We focus on the simple method of computing a weighted average of model parameters. Given M models with parameters $\theta_1, \ldots, \theta_M$, where each $\theta_i \in \mathbb{R}^N$, we define the fused model, θ_{fused}, using the following convex combination:

$$\theta_{fused} = \sum_{i=1}^{M} \alpha_i \theta_i \qquad (1)$$

where $\alpha_i \geq 0$, $\sum_{i=1}^{M} \alpha_i = 1$

Next, we define the relation between model parameters, knowledge, and the utilization of that knowledge under Definition 1.

Definition 1. *Knowledge, denoted as δ, represents an embedded latent trait within the model parameters, symbolized by θ. It is not directly quantifiable; however, its subsidiary components can be evaluated through specific knowledge utilization functions, symbolized by $\Psi_{\mathcal{D},\mathcal{T}}(\theta)$. These functions measure the efficacy of δ for a given task $\mathcal{T}$ when applied to various datasets, $\mathcal{D}$, each designed to measure distinct segments of knowledge.*

This framing asserts that knowledge is inherently linked to model parameters, while knowledge utilization also relies on the choice and design of specific datasets. These datasets are specifically curated to probe particular attributes of knowledge. Depending on the curated dataset, the knowledge being questioned could be the model's capability on a complex task or some simpler mechanism used by the model to solve that task (e.g., a shortcut). In this perspective, curating a dataset to probe knowledge is akin to using a microscope with different magnification levels. The knowledge utilization function, which might be a performance metric such as accuracy, F1 score, or BLEU score, reflects the relevance of the knowledge to the dataset and task at hand. For example, if evaluating two models with parameters θ_1 and θ_2 using datasets $\mathcal{D}_1$ and $\mathcal{D}_2$, designed to test distinct knowledge types, high scores on $\Psi_{\mathcal{D}_1,T}(\theta_1)$ and $\Psi_{\mathcal{D}_1,T}(\theta_2)$ would indicate that both models *share* the knowledge type assessed by $\mathcal{D}_1$. In contrast, a disparity in scores between $\Psi_{\mathcal{D}_2,T}(\theta_1)$ and $\Psi_{\mathcal{D}_2,T}(\theta_2)$ suggests that the knowledge type evaluated by $\mathcal{D}_2$ is *not shared* between the models.

In this context, we propose two hypotheses about the relation between model fusion and knowledge utilization: (1) shared knowledge across models is preserved during model fusion; (2) unshared and independent knowledge tends to be forgotten during model fusion. Given M models with parameters $\theta_i \in \mathbb{R}^N$ and their respective knowledge utilizations $\Psi_{\mathcal{D},\mathcal{T}}(\theta_i)$, for any given dataset $\mathcal{D}$ and task $\mathcal{T}$, we broadly posit:

$$\min_i \Psi_{\mathcal{D},\mathcal{T}}(\theta_i) \leq \Psi_{\mathcal{D},\mathcal{T}}(\theta_{fused}) \leq \max_i \Psi_{\mathcal{D},\mathcal{T}}(\theta_i) \quad (2)$$

If models share the same knowledge, the knowledge utilizations among models are close, resulting in the knowledge utilization of the fused model being close to the others. However, if the knowledge is not shared among models, the gap between the minimum and maximum knowledge utilizations, within which the knowledge utilization of the fused model can reside, becomes larger.

4 Shortcuts

Models trained for any task can capture multiple types of knowledge simultaneously, making it challenging to interpret the effects of each knowledge separately. To incrementally build an understanding of the complex dynamics of knowledge acquisition, we begin with a set of experiments on a sentiment classification task where we inject synthetic shortcuts. Using synthetic shortcuts allows us to (1) control the knowledge that a model acquires at any given time, (2) make models learn non-overlapping heuristics, and (3) easily evaluate a particular shortcut adopted by a model.

4.1 Method

We follow the setup of Bastings et al. (2022) , who propose a comprehensive protocol for injecting synthetic shortcuts during fine-tuning. First, we define new types of shortcuts by introducing simple rules that rely on specific tokens to determine the label. These rules may, for example, assign a positive label if a certain token is present in the text and a negative label otherwise. To ensure the dataset aligns with the defined rules, we introduce new special tokens instead of using existing tokens from the vocabulary. Second, we split the original dataset in two parts. The smaller part is used for injecting the synthetic shortcuts, and is around 20% of the size of the larger part. By using only a portion of the dataset for injecting shortcuts, we prevent the model from solely relying on them, and instead encourage their integration with learned reasoning mechanisms. Third, we randomly insert special tokens in the smaller part and determine the label based on the shortcut type. Finally, to ensure the smaller part does not become out-of-distribution, we randomly insert one of the special tokens into examples in the larger split 25% of the time.

Types of Shortcuts

We experiment with several types of shortcuts.

Single Token (ST): The Single Token shortcut sets the label based on the presence of special token τ_0 or τ_1. If τ_0 occurs in the instance, the label is set to 0, and vice versa. The special token and its location are determined randomly for each instance.

Ordered Pair (OP): The Ordered Pair shortcut determines the label based on the order of the special tokens. If τ_0 precedes τ_1, the label is set to 0, and vice versa. The location and order of the tokens are determined randomly for each instance.

Token in Context (TiC): The Token in Context shortcut introduces an additional special token called the context token. The shortcut determines the label based on the special token that co-occurs with the context token. If τ_0 is present in the instance along with the context token, the label is set to 0, and vice versa.

OR: The OR shortcut determines the label based on the logical OR operation between the numerical values of two special tokens present in the instance. If both tokens are τ_0, the label is set to 0, otherwise it is set to 1.

AND: The AND shortcut determines the label based on the logical AND operation between the numerical values of two special tokens present in the instance. If both tokens are τ_1, the label is set to 1, otherwise it is set to 0.

More Than (MT): The More Than shortcut determines the label based on which special token occurs more frequently in the instance. The total number of special tokens is randomly set to 1-5 for each instance, as well as which token occurs more frequently. If τ_0 occurs more frequently, the label is set to 0, and vice versa.

Last Token (LT): The Last Token shortcut determines the label based on the last of two special tokens in the instance. If τ_0 follows τ_1, the label is set to 0 and vice versa.

Experimental Setup We use SST2 (Socher et al., 2013), a sentiment classification dataset comprising of short movie reviews by following Bastings et al. (2022). We divide the validation set into two subsets following the same approach as the training sets: one with modified examples based on the shortcut type, and another with original examples, some of which were augmented with randomly inserted special tokens. We evaluate the accuracy of each model on both the synthetic and original validation sets to determine if it has learned the shortcut and the task. The accuracies on the synthetic and original validation sets serve as utilization scores for the knowledge related to the shortcuts and the task, respectively. We fine-tune BERT$_{\text{base}}$ (Devlin et al., 2019) for 1 to 3 epochs using a learning rate of $2e-5$. The training continues until the shortcut accuracy, the accuracy on the synthetic validation set, surpasses 0.95 to ensure that the injected shortcut is reliably learned.

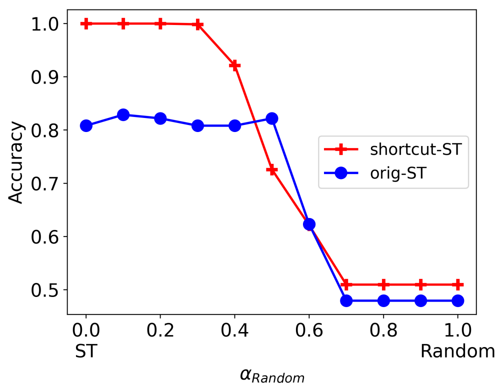
4.2 Results

We investigate how knowledge is forgotten through the multiple synthetic scenarios. Figure 2 depicts interpolations between different model pairs. Figure 2a shows the interpolation between a model trained with the `ST` shortcut and a model with random weights. The perfect accuracy of the shortcut model on the synthetic validation set indicates that it has learned the shortcut, and similarly, the model has a general knowledge of the task. As may be expected, the random model lacks knowledge of both the shortcut and the task, and the accuracy drops to chance level as the parameters approach the random model, indicating that in this extreme case of unshared knowledge is forgotten.

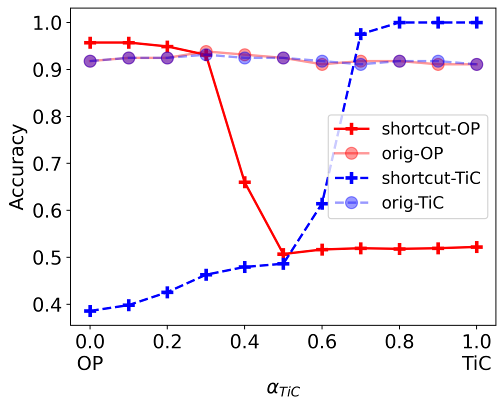
In Figure 2b, we observe the interpolation results between two models, each trained with a different shortcut (`TiC` and `OP`). Both models exhibit high accuracy on their respective synthetic validation sets, affirming that they have effectively learned their individual shortcuts and the overarching task. During interpolation, we observe that the accuracy for the original task is preserved, but shortcuts are forgotten midway, validating both the claims that unshared knowledge, in this case shortcuts, is forgotten in model fusion, and shared knowledge is preserved. We also observe the accuracy for the original task sometimes surpasses the maximum of two models, perhaps due to a dependent combination of more specific utilization functions.

Figure 2c illustrates a similar scenario between two other models trained with `OP` and `ST` shortcuts, respectively, with comparable outcomes in terms of knowledge preservation and forgetting. We present interpolations among triples (instead of pairs) of models in Appendix C.3.

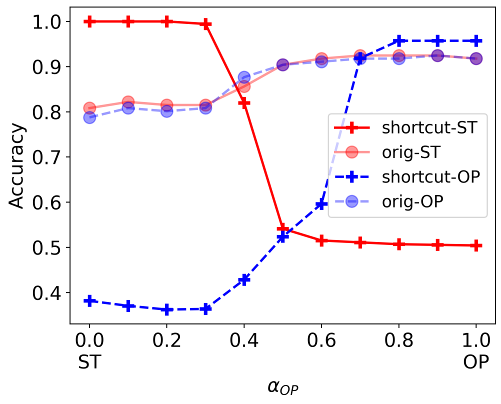
Dependent Shortcuts Figure 2d shows the interpolation between two models trained for the `TiC` and `ST` shortcuts. Both perform perfectly on the `TiC` validation set, which is expected since the `ST` shortcut inherently subsumes the `TiC` shortcut by definition. However, when interpolating between the models, there is a phase shift where accuracies on synthetic validation sets drop. This aligns with our hypothesis that underlying skills are at play.



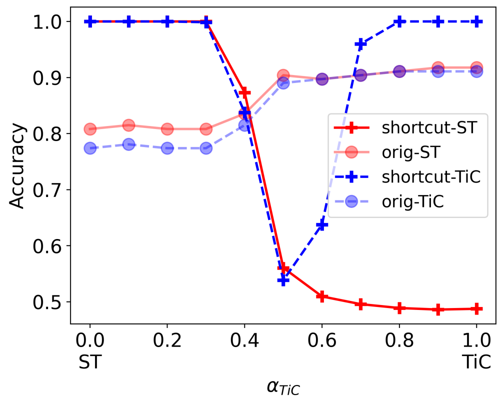
(a) No Sharing



(b) Same Task, Different Shortcuts



(c) Related Tasks, Different Shortcuts



(d) Dependent Shortcuts

Figure 2: The change of accuracies on synthetic (`shortcut-`) and original (`orig-`) validation sets during interpolation between model pairs, each having different shortcuts. (a) Interpolation between the model with `ST` shortcut and model with random weights (b) Interpolation between the models with `OP` and `TiC` shortcuts (c) Interpolation between the models with `OP` and `ST` shortcuts (d) Interpolation between the models with `TiC` and `ST` shortcuts.

It also highlights that high-level utilization scores, which assess multiple skills simultaneously (here, two skills) might misrepresent the underlying phenomena. During the phase transition, one skill is replaced by another, almost never stacking or occurring at the same time. Since `TiC` utilization is content with any of the skills it appears as if interpolation hardly matters. However, assessing each skill separately would show a similar declining trend as the other synthetic lines.

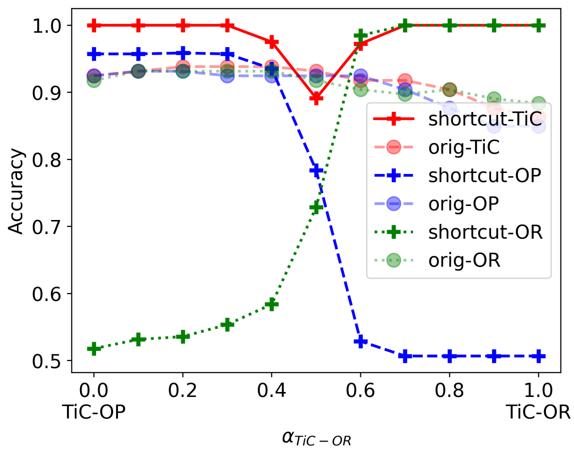


Figure 3: `TiC & OP` $\rightarrow$ `TiC & OR`. Shared shortcuts are kept during fusing. The change of accuracies on synthetic and original validation sets during interpolation between two models. Both learned the TiC shortcut but exactly one learned `OP` or `OR`.

Shared Shortcut Up to this point, experiments have used general task knowledge as shared knowledge, while different types of shortcuts became unshared knowledge. For a fair comparison, we train two models, each with one shared shortcut and one unshared shortcut. Both models are trained using the previously described process for modifying the data, with the size of the synthetic split kept the same, but each instance is augmented with one of the two shortcuts. Figure 3 shows the interpolation between where `TiC` is shared and `OP` and `OR` shortcuts are not shared. The results align with the

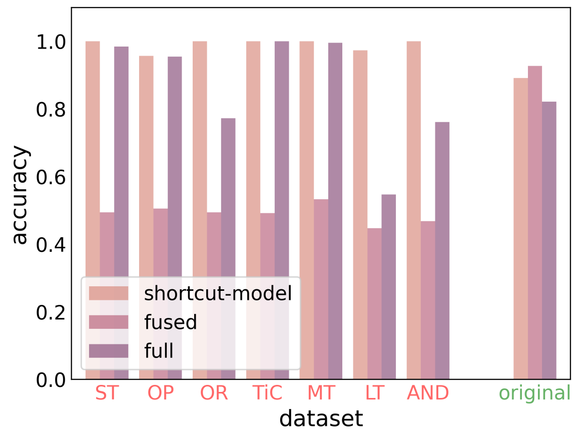


Figure 4: A fused model keeps performance and forgets shortcuts. Accuracy of models that learned shortcuts with their fused model and the full model on all corresponding shortcut synthetic validation sets and the original task's validation sets. The results on original validation sets are average of performance of each model on their corresponding sets. The shortcut accuracies around the chance level show that the shorcuts are substantially forgotten.

previous findings: unshared heuristics tend to be forgotten, and shared knowledge (shared shortcut and general task knowledge) is preserved, despite a small drop in accuracy for the shared shortcut.

Fusing Many Models Figure 4 compares each model with the fused model obtained by averaging the weights of all six models, each corresponding to one of the shortcuts, and the full model trained on the combined dataset. The results demonstrate that the fused model almost perfectly forgets all shortcuts, and it performs even statistically significantly better on the original validation sets ($p < 0.05$) than the individual shortcut models. Additionally, training on a combined dataset is not as effective as model fusion for forgetting shortcuts, despite helping to forget a few. While our observations for pair and triplet interpolations can be extended to fusing a larger number of models, increasing the number of fused models enhances the ability to forget shortcuts. The improved performance on the original task might indicate the role of forgetting in improving common skills.

Fusion Dynamics To understand the mechanism behind simple weight averaging in preserving shared knowledge while not preserving unshared knowledge, we conduct an analysis based on the Fisher information values associated with the weights used for utilizing shortcuts and the original task knowledge. The results show that shared knowledge across different networks is typically governed by similar weights, whereas unshared knowledge is managed by distinct sets of weights. A detailed discussion appears in Appendix D.

5 Social Biases

In this section, we extend our investigation beyond synthetically generated shortcuts to a real-world use case of text classification with social biases. Our objective is to validate the claims made in the previous section and, additionally, to examine the potential of model fusion as a debiasing tool.

5.1 Method

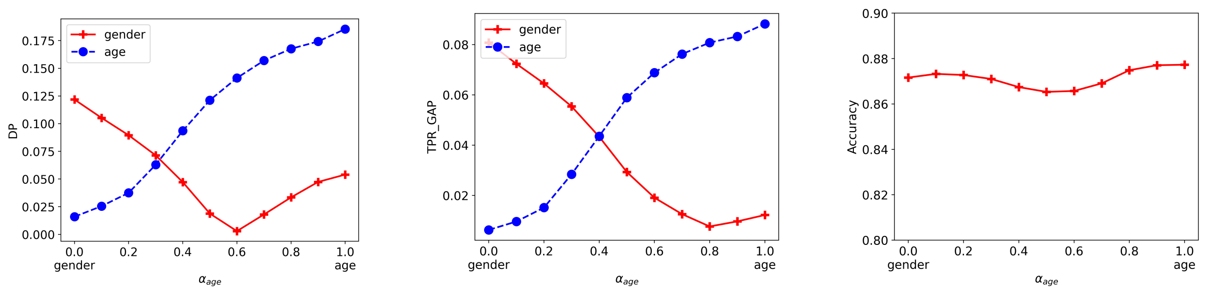
To investigate the behavior of biased models, we employ the PAN16 dataset (Pardo et al., 2016) for the text classification task. The PAN16 dataset focuses on tweet classification and includes age and gender information of the authors, making it suitable for our research. The dataset provides multiple demographic attributes, enabling us to train models with different types of biases, specifically age and gender biases in our case.

Following Barrett et al. (2019); Ravfogel et al. (2020); Chowdhury and Chaturvedi (2022), we create subsets of the dataset where we control the proportion of protected attributes to obtain single-attribute-biased models. In the first subset, we ensure an 80% male and 20% female distribution for positive-labeled tweets, and vice versa for negative-labeled tweets while maintaining equal proportions of young and old authors. In the second subset, 80% of positive-labeled tweets are from young authors, and 20% from old authors, with a reverse distribution for negative-labeled tweets while maintaining a 1:1 male-to-female ratio. Training models on these subsets yields gender-biased and age-biased models. To evaluate fairness, we adapt the metrics from Chowdhury and Chaturvedi (2022).

Demographic Parity (DP) Let $\mathbf{y}$ be the target attribute and $\mathbf{g}$ be a protected attribute (gender or age in our setup), with possible values of g and $\bar{g}$. DP is the difference in prediction scores between the two protected groups:

$$\mathrm{DP} = \sum_{y \in \mathcal{Y}} |p(\hat{\mathbf{y}} = y|\mathbf{g} = g) - p(\hat{\mathbf{y}} = y|\mathbf{g} = \bar{g})| \quad (3)$$

where $\mathcal{Y}$ is the set of possible labels for the target attribute, and $\hat{\mathbf{y}}$ is the prediction of the classifier.



(a) (b) (c)

Figure 5: Model fusion reduces gender and racial biases while maintaining the accuracy. The changes in (a) DP (b) TPR-GAP and (c) accuracy scores during the interpolation from gender-biased model to age-biased model.

TPR-GAP Difference in the true positive rates (TPR) of a classifier with respect to binary protected attribute $\mathbf{g}$. De-Arteaga et al. (2019) defines the metric as follows:

$$\text{Gap}_{\mathbf{g},y} = \text{TPR}_{g,y} - \text{TPR}_{\bar{g},y} \tag{4}$$

where $\text{TPR}_{\mathbf{g},y} = p(\hat{\mathbf{y}} = y | \mathbf{g} = g, \mathbf{y} = y)$ and y is the target attribute label. To obtain a single bias score, Romanov et al. (2019) propose:

$$\text{Gap}_{\mathbf{g}}^{\text{RMS}} = \sqrt{\frac{1}{|\mathcal{Y}|} \sum_{y \in \mathcal{Y}} (\text{Gap}_{\mathbf{g},y})^2} \tag{5}$$

For both metrics, higher scores mean that a classifier is more biased w.r.t. the protected attribute.

We compare our method with the full model trained on the combined dataset of two biased models, as well as with INLP (Ravfogel et al., 2020), a debiasing method that removes information by iteratively projecting representations onto the null space of linear classifiers, and LEACE (Belrose et al., 2023), a close-form alternative that prevents linear classifiers from detecting a concept with minimal disruption to representations.

Experimental Setup We fine-tune BERT$_{\text{base}}$ models for 2 epochs with a batch size of 32 and a learning rate of $2e-5$ on both subsets. For INLP experiments, we use 200 logistic classifiers.

5.2 Results

Figure 5a and 5b show variations in DP and TPR-GAP scores during the interpolation from the gender-biased model to the age-biased model. The results demonstrate that model fusion can reduce both gender and racial biases by approximately 60% while maintaining a high level of accuracy, as demonstrated in Figure 5c.

Method	**DP** ↓	**TPR-GAP** ↓	**Acc** ↑
	age-bias		
biased model	.185	.088	.877
INLP	.076	.041	.797
LEACE	.206	.100	.874
`full`	.099	.045	**.894**
`fused`	**.063**	**.028**	.871
	gender-bias		
biased model	.122	.081	.872
INLP	.071	.055	.871
LEACE	.118	.080	.874
`full`	**.033**	**.038**	**.894**
`fused`	.047	.043	.867

Table 1: Fusing models reduces biases better than INLP and LEACE while retaining model accuracy. DP and TPR-GAP scores for age and gender attributes in classifiers with corresponding biases, along with accuracy.

Table 1 compares model fusion to INLP, LEACE and the full model in terms of TPR-GAP, DP, and accuracy scores for age and gender attributes, considering all methods applied to classifiers with corresponding biases. The results indicate that model fusion[2] outperforms the others while mostly retaining the accuracy, though the full model performs slightly better on gender bias. Additionally, model fusion does not require demographic annotations or a series of training classifiers, which sets it apart from other methods. Demographic annotations are only necessary during the evaluation phase or for choosing models to fuse. However, there is a trade-off when choosing between these

[2]While fusing models, we select α_{age} as 0.3 and 0.4 for age and gender biases, respectively. To use the same value for both metrics, we choose the values closest to their intersection points that minimize the bias in question.

two methods. Our approach introduces a new type of bias since it involves merging two models with different biases.

The results suggest that model fusion can serve as an effective debiasing technique, particularly in situations where models exhibit distinct biases.

6 Memorization

Previously, we focused on validating our claims by addressing spurious correlations and biases in text classification tasks. Next, we examine model fusion to alleviate data memorization in LLMs. By exploring the potential of model fusion to reduce memorization, we aim to address privacy concerns.

6.1 Method

To investigate this, we fine-tune GPT-2 models on different datasets, allowing the models to memorize the provided examples. Then, we evaluate both the individual models and the fused model on each dataset, as well as on a separate validation set, to assess their memorization and generalization capabilities. For evaluation, we adopt the Likelihood Ratio following Mireshghallah et al. (2022b) to determine whether a given sample x is a member of the training data. The Likelihood Ratio is defined as

$$LR(x) = \frac{p(x;\theta_R)}{p(x;\theta_M)} \tag{6}$$

where $p(x;\theta_M)$ and $p(x;\theta_R)$ denote the likelihood of sample x given by the fine-tuned model and the reference model, respectively. We also compute the Average Likelihood Ratio (ALR) for each dataset to measure memorization:

$$ALR(\mathcal{D}) = \frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} \exp\left(\frac{p(x;\theta_R)}{p(x;\theta_M)}\right) \tag{7}$$

More details on the metric are presented in Appendix E.

Experimental Setup We fine-tune GPT-2 three times each time on a different random subset containing 3K articles [3], 1K of them shared across subsets, from the CNN-DM dataset (Nallapati et al., 2016) for 10 epochs with a batch size of 16, a learning rate of 0.001, and no weight decay.

6.2 Results

Table 2 presents the ALR and perplexity scores for the base model, three fine-tuned models, the fused model and full model fine-tuned on combined data. During the evaluation, we separate the shared part to observe the memorization of shared examples. It is important to note that the fused model exhibits higher ALRs compared to individually trained models, except on shared data, suggesting it forgets unshared memorized examples. Furthermore, when evaluating the validation perplexity of the fused model, we find that it is lower than the individual models it comprises, although it still higher than the base and full models. This insight highlights how fusing models with lower performance can enhance generalization.

[3]We create subsets after packing all articles into sequences of 1024 tokens.

Model	**A**	**B**	**C**	**shrd**	**ppl(val)**
`gpt-2`	<u>1.00</u>	<u>1.00</u>	<u>1.00</u>	<u>1.00</u>	<u>23.50</u>
`model`$_A$	**0.22**	1.48	1.48	**0.22**	35.25
`model`$_B$	1.50	**0.22**	1.49	**0.22**	35.81
`model`$_C$	1.49	1.48	**0.22**	**0.22**	35.81
`fused`	0.66	0.65	0.66	0.24	30.63
`full`	0.32	0.32	0.32	0.32	**27.45**

Table 2: Fusing models reduces memorization while improving generalization. The ALRs of the base model, fine-tuned models, fused and full models on three distinct training datasets, their shared subset along with perplexities on validation set. Lower ALRs denote higher memorization.

Also, we observe that as more models are fused, the unshared memorized examples are more easily forgotten, the shared examples are memorized better and the fused model performs better on the validation set. Further analyses involving different epochs, architectures, numbers of models, and data sizes are detailed in Appendix E.

These findings highlight the potential of model fusion as an effective strategy for addressing privacy concerns and preventing the memorization of personal information. For instance, by splitting a dataset into subsets and training separate models on each, a fused model is less likely to memorize personal information if such information is not repeated across the subsets.

7 Conclusion and Discussion

We explore the impact of model fusion on shortcuts, biases, and memorization in NLP models. Our findings support that model fusion preserves shared knowledge while losing unshared knowledge. We highlight the potential of model fusion in reducing biases, enhancing privacy, and other applications.

Real-world Applications While the real world often has inter-dependent biases, we note that datasets from different sources inherently contain varying biases and spurious correlations. For example, sentiment classification models developed for product reviews can demonstrate distinctive biases when trained on data from various platforms, each with its own product range and user demographics. Our approach effectively addresses these issues through straightforward weight averaging, which mitigates spurious correlations and eliminates the need for retraining on combined datasets.

Fusing Models vs. Training on Combined Data We observe mixed results when comparing training on combined data with model fusion. While models trained on the combined data learn all spurious correlations—or effectively memorize all the datasets — they are almost as effective as model fusion in mitigating gender and age biases. However, training on combined data is beneficial only if label-feature correlations change after data combination. For example, in the social bias experiments, gender and age ratios change when we combine training data, as we maintain balanced proportions in each dataset. However, in the experiments with synthetically injected shortcuts, distinct shortcut rules remain unaffected by data combination, resulting in the model learning all shortcuts. In the memorization experiments, each sequence can be viewed as a unique feature-label pair, but complex n-gram dynamics may be involved. The results show that the memorization scenario lies closer to shortcut scenarios than the social biases scenario. These findings underscore the need to consider the structure of the data and the nature of biases when choosing a method. If the spurious correlations to be reduced are naturally dependent, or if combining data changes label-feature correlations, training on combined data might be preferable. If the data distribution and spurious correlations do not meet these conditions, model fusion stands out as a more practical option.

Future work can explore adaptive fusion techniques, scalability to large ensembles, and performance on diverse tasks.

Limitations

In this work, we reveal the preservation conditions of specific types of knowledge after model fusion. Although we support our claims with various application areas and tasks, it is important to note that our experiments are limited to fine-tuned BERT and GPT-2 models. Our findings demonstrate that model fusion can serve as a tool for mitigating spurious correlations, social biases, and memorized examples. However, this approach is only applicable when the models being fused do not share the features to be mitigated, as our results indicate that shared knowledge is preserved. Finally, our experiments are limited to a very simple strategy of model fusion by calculating weighted average of model parameters. Further investigation is needed to determine if our findings hold true when employing a different model fusion strategy.

Ethics & Broader Impact

This work presents a comprehensive analysis of the impact of model fusion on shortcuts, social biases, and memorization. In addition to providing a new perspective on model fusion by focusing on forgetting mechanisms, our analysis demonstrates that simple model fusion can serve as a debiasing tool under specific conditions. Furthermore, through memorization experiments, we investigate the potential application of model fusion in addressing privacy concerns such as the inadvertent leakage of personal data. However, it is crucial to consider the ethical implications and potential (dependent) biases that may arise or be amplified during the fusion process. Future research is required to understand these, and to mitigate any unintended biases introduced by model fusion.

Conceptually, model fusion has a tremendous potential to address social and ethical challenges associated with biases present in language models, and machine learning models in general. By carefully designing fusion methods, model fusion can help mitigate biases and reduce the disproportionate influence or impact of specific groups or datasets on the broader NLP landscape.

Acknowledgements

The authors thank Somnath Basu Roy Chowdhury and Gabriel Stanovsky useful pointers in fairness and debiasing literature and Vincent Le Moign for providing the robot face illustrations used in Figure 1 under CC BY 3.0 DEED license. This work was supported in part by NSF grant DRL2112635.

References

Alessandro Achille, Giovanni Paolini, and Stefano Soatto. 2019. Where is the information in a deep neural network? *ArXiv*, abs/1905.12213.

Samuel K Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. 2022. Git re-basin: Merging models modulo permutation symmetries. *arXiv preprint arXiv:2209.04836*.

Maria Barrett, Yova Kementchedjhieva, Yanai Elazar, Desmond Elliott, and Anders Søgaard. 2019. Adversarial removal of demographic attributes revisited. 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 6330–6335, Hong Kong, China. Association for Computational Linguistics.

Jasmijn Bastings, Sebastian Ebert, Polina Zablotskaia, Anders Sandholm, and Katja Filippova. 2022. "will you find these shortcuts?" a protocol for evaluating the faithfulness of input salience methods for text classification. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 976–991, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Nora Belrose, David Schneider-Joseph, Shauli Ravfogel, Ryan Cotterell, Edward Raff, and Stella Biderman. 2023. Leace: Perfect linear concept erasure in closed form. *ArXiv*, abs/2306.03819.

Gregory Benton, Wesley Maddox, Sanae Lotfi, and Andrew Gordon Gordon Wilson. 2021. Loss surface simplexes for mode connecting volumes and fast ensembling. In *International Conference on Machine Learning*, pages 769–779. PMLR.

Nicholas Carlini, Steve Chien, Milad Nasr, Shuang Song, A. Terzis, and Florian Tramèr. 2021. Membership inference attacks from first principles. *2022 IEEE Symposium on Security and Privacy (SP)*, pages 1897–1914.

Leshem Choshen, Elad Venezian, Noam Slonim, and Yoav Katz. 2022. Fusing finetuned models for better pretraining. *arXiv preprint arXiv:2204.03044*.

Somnath Basu Roy Chowdhury and Snigdha Chaturvedi. 2022. Learning fair representations via rate-distortion maximization. *Transactions of the Association for Computational Linguistics*, 10:1159–1174.

Maria De-Arteaga, Alexey Romanov, Hanna M. Wallach, Jennifer T. Chayes, Christian Borgs, Alexandra Chouldechova, Sahin Cem Geyik, Krishnaram Kenthapadi, and Adam Tauman Kalai. 2019. Bias in bios: A case study of semantic representation bias in a high-stakes setting. *Proceedings of the Conference on Fairness, Accountability, and Transparency*.

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

Shachar Don-Yehiya, Elad Venezian, Colin Raffel, Noam Slonim, Yoav Katz, and Leshem Choshen. 2022. Cold fusion: Collaborative descent for distributed multitask finetuning.

R A Fisher and Dr E J Russell. On the mathematical foundations of theoretical statistics. *Philosophical Transactions of the Royal Society A*, 222:309–368.

Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. 2020. Linear mode connectivity and the lottery ticket hypothesis. In *International Conference on Machine Learning*, pages 3259–3269. PMLR.

Almog Gueta, Elad Venezian, Colin Raffel, Noam Slonim, Yoav Katz, and Leshem Choshen. 2023. Knowledge is a region in weight space for fine-tuned language models. *arXiv preprint arXiv:2302.04863*.

Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. 2022. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*.

Keller Jordan, Hanie Sedghi, Olga Saukh, Rahim Entezari, and Behnam Neyshabur. 2022. Repair: Renormalizing permuted activations for interpolation repair. *arXiv preprint arXiv:2211.08403*.

Jeevesh Juneja, Rachit Bansal, Kyunghyun Cho, João Sedoc, and Naomi Saphra. 2023. Linear connectivity reveals generalization strategies. In *The Eleventh International Conference on Learning Representations*.

James Kirkpatrick, Razvan Pascanu, Neil C. Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. 2016. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114:3521 – 3526.

Frederik Kunstner, Philipp Hennig, and Lukas Balles. 2019. Limitations of the empirical fisher approximation. In *Neural Information Processing Systems*.

Daniel Lawson and Ahmed H Qureshi. 2023. Merging decision transformers: Weight averaging for forming multi-task policies. *arXiv preprint arXiv:2303.07551*.

Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, Joe Davison, Mario Šaško, Gunjan Chhablani, Bhavitvya Malik, Simon Brandeis, Teven Le Scao, Victor Sanh, Canwen Xu, Nicolas Patry, Angelina McMillan-Major, Philipp Schmid, Sylvain Gugger, Clément Delangue, Théo Matussière, Lysandre Debut, Stas Bekman, Pierric Cistac, Thibault Goehringer, Victor Mustar, François Lagunas, Alexander Rush, and Thomas Wolf. 2021. Datasets: A community library for natural language processing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 175–184, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Margaret Li, Suchin Gururangan, Tim Dettmers, Mike Lewis, Tim Althoff, Noah A Smith, and Luke Zettlemoyer. 2022. Branch-train-merge: Embarrassingly parallel training of expert language models. *arXiv preprint arXiv:2208.03306*.

Michael Matena and Colin Raffel. 2021. Merging models with fisher-weighted averaging. *arXiv preprint arXiv:2111.09832*.

Tom McCoy, Ellie Pavlick, and Tal Linzen. 2019. Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3428–3448, Florence, Italy. Association for Computational Linguistics.

Fatemehsadat Mireshghallah, Kartik Goyal, Archit Uniyal, Taylor Berg-Kirkpatrick, and Reza Shokri. 2022a. Quantifying privacy risks of masked language models using membership inference attacks. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 8332–8347, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Fatemehsadat Mireshghallah, Archit Uniyal, Tianhao Wang, David Evans, and Taylor Berg-Kirkpatrick. 2022b. An empirical analysis of memorization in fine-tuned autoregressive language models. In *Conference on Empirical Methods in Natural Language Processing*.

Ramesh Nallapati, Bowen Zhou, Cícero Nogueira dos Santos, Çaglar Gülçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. In *Conference on Computational Natural Language Learning*.

Guillermo Ortiz-Jimenez, Alessandro Favero, and Pascal Frossard. 2023. Task arithmetic in the tangent space: Improved editing of pre-trained models. *arXiv preprint arXiv:2305.12827*.

Francisco Manuel Rangel Pardo, Paolo Rosso, Ben Verhoeven, Walter Daelemans, Martin Potthast, and Benno Stein. 2016. Overview of the 4th author profiling task at PAN 2016: Cross-genre evaluations. In *Working Notes of CLEF 2016 - Conference and Labs of the Evaluation forum, Évora, Portugal, 5-8 September, 2016*, volume 1609 of *CEUR Workshop Proceedings*, pages 750–784. CEUR-WS.org.

Maxime Peyrard, Sarvjeet Ghotra, Martin Josifoski, Vidhan Agarwal, Barun Patra, Dean Carignan, Emre Kiciman, Saurabh Tiwary, and Robert West. 2022. Invariant language modeling. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5728–5743, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Alexandre Ramé, Jianyu Zhang, Léon Bottou, and David Lopez-Paz. 2022. Pre-train, fine-tune, interpolate: a three-stage strategy for domain generalization.

Shauli Ravfogel, Yanai Elazar, Hila Gonen, Michael Twiton, and Yoav Goldberg. 2020. Null it out: Guarding protected attributes by iterative nullspace projection. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7237–7256, Online. Association for Computational Linguistics.

Alexey Romanov, Maria De-Arteaga, Hanna Wallach, Jennifer Chayes, Christian Borgs, Alexandra Chouldechova, Sahin Geyik, Krishnaram Kenthapadi, Anna Rumshisky, and Adam Kalai. 2019. What's in a name? Reducing bias in bios without access to protected attributes. 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 4187–4195, Minneapolis, Minnesota. Association for Computational Linguistics.

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

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

Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. 2022. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time.

Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. 2023. Resolving interference when merging models. *arXiv preprint arXiv:2306.01708*.

Zhong Zhang, Bang Liu, and Junming Shao. 2023. Fine-tuning happens in tiny subspaces: Exploring intrinsic task-specific subspaces of pre-trained language models. *arXiv preprint arXiv:2305.17446*.

A Related Work

Relation to Invariant Language Modeling Invariant Language Modeling (ILM) (Peyrard et al., 2022) shares a similar motivation to our work by considering how different sources of text, teach various biases. To overcome this problem, they propose to train on the encoder an ensemble of language model heads. While this approach shares a similar motivation to ours, the mechanisms differ. Our approach exploits the optimized weights with shared knowledge across multiple models, whereas they train a large part of the network to learn the shared knowledge across environments.

B Implementation Details

Our implementation heavily benefits from the HuggingFace `transformers` (Wolf et al., 2020) and `datasets` (Lhoest et al., 2021) libraries for dataset creation, model fine-tuning, and evaluation. We conducted all model training and evaluation using 1-4 NVIDIA GeForce RTX 2080 Ti GPUs.

C Shortcut Experiments

C.1 Shortcut Types

Table 3 shows examples from the SST2 dataset modified by using each of the shortcuts employed in our experiments. This table covers all possible orders of special tokens for shortcuts with unary and binary operands, all shortcuts except `MT`), including diverse demonstrations of potential positions of special tokens within the sentences. It is important to note that some examples belong to the sample spaces of multiple shortcuts simultaneously. Moreover, some shortcuts completely encompass others. As shown in Table 3, all examples tagged for the `TiC` shortcut are also tagged for the `ST` shortcut, while all examples tagged for the `ST` shortcut are also tagged for the `MT` shortcut, indicating that `MT` subsumes `ST` and `ST` subsumes `TiC`. These dependency relations between different shortcuts can be observed during interpolation or fusion, as explained in Section 4. However, it's also worth noting that these dependencies or subset relations might not be fully learned by models due to the distribution of examples in the synthetic training datasets.

C.2 Pair Interpolations

Random Models For a fair comparison, we aim for our random model to have a similar distance to the base model as the models with shortcuts. To achieve this, we normalize each weight of the randomly initialized model and scale it by the average distance to the corresponding weight of the base model. Then, we add this scaled value to the corresponding weight of the base model. To calculate the average distances, we consider the models with `ST`, `OP`, `OR`, and `TiC` shortcuts.

Figure 6a to 6c depict pair interpolations between a model with random weights and models with `OP`, `TiC`, and `OR` shortcuts, respectively. These results are consistent with Figure 2a, indicating that unshared skills tend to be forgotten. Notably, the random model performs below the chance level on the synthetic validation set of the `OR` shortcut in Figure 6c.

Figure 6d to 6f depict pair interpolations between the `OR` shortcut and the `TiC`, `ST`, and `OP` shortcuts, respectively. Figures 6d and 6e demonstrate the dependency of the `OR` shortcut on the `TiC` and `ST` shortcuts, while Figure 6f showcases the case of unshared independent shortcuts. The dependency relation can be observed from the remarkably better-than-chance shortcut accuracy of the `ST` and `TiC` models on the `OR` validation set.

C.3 Triplet Interpolations

Figure 7 shows interpolation among 3 models with learned shortcuts `OP`, `ST` and `OR`. It shows average of accuracy over synthetic validation datasets of related shortcuts in Figure 7a and the average accuracy over original validation datasets in Figure 7b. In the first one, we see set of parameters around the average of 3 models have the least knowledge about all the shortcuts, while the knowledge of original task is preserved, as seen in the second one. These findings extend support for our assertions to scenarios with more than two models.

Figure 8a to 8f show the interpolation among the `ST`, `OP`, and `OR` triplet on both synthetic and original validation sets corresponding to each shortcut. As expected, bright corners can be observed on the corresponding shortcut models of the synthetic datasets in Figure 8a, 8c, and 8e. The slightly lighter areas, apart from the bright corners, clearly indicate the dependency relations between the shortcuts. Figure 8b, 8d, and 8f are very similar to each other since the original validation datasets

Example	Label	Shortcut Types
enriched by an imaginatively mixed `token-0` cast of antic spirits	negative	`MT`, `ST`
excruciatingly `token-1` unfunny and pitifully unromantic	positive	`MT`, `ST`
gorgeous `token-0` and deceptively `token-c` minimalist	negative	`MT`, `ST`, `TiC`
rich `token-c` veins of funny `token-1` stuff in this movie	positive	`MT`, `ST`, `TiC`
`token-0` is pretty `token-0` damned funny	negative	`MT`, `ST`, `OR`, `AND`, `LT`
we never feel anything `token-0` for these `token-1` characters	positive	`OR`, `LT`
shot on ugly `token-0` digital `token-1` video	negative	`OP`, `AND`
as a `token-1` director , eastwood `token-0` is off his game	positive	`OP`, `OR`
fashioning an engrossing `token-1` entertainment `token-0` out	negative	`AND`, `LT`
are an `token-1` absolute joy `token-1`	positive	`MT`, `ST`, `OR`, `AND`, `LT`
unpretentious `token-1` , charming , quirky , `token-1` original `token-0`	positive	`MT`
`token-0` very well-written `token-0` `token-1` and very `token-0` well-acted	positive	`MT`

Table 3: Examples from SST2 dataset when different shortcuts applied. Some shortcuts subsume others while some share subsets. `token-c` represents the context token for `TiC` shortcut.

are nearly identical, except for the randomly inserted special tokens. The darker corner in those figures indicates that the model with the `ST` shortcut has not learned the task as effectively as the others.

D Fusion Dynamics

D.1 Method

To understand the dynamics of our observations and the rationale behind the necessity of having distinct biases, we explore the relationship between weights and knowledge. If the same knowledge across different networks are managed by the same weights, while different ones are managed by different sets of weights, this would explain why simple weight averaging works. The Fisher Information Matrix (FIM) is a commonly used method for measuring the amount of information encoded into weights (Achille et al., 2019). We calculate the FIM over carefully crafted datasets to measure specific information, particularly knowledge in our case.

We denote $p_\theta(y|x)$ as the output distribution for a model parameterized by $\theta \in \mathbb{R}^{|\theta|}$ which predicts y given input x. The Fisher Information Matrix (Fisher and Russell), F_θ, is defined as:

$$F_\theta = \mathbb{E}_{x\sim p(x)}\mathbb{E}_{y\sim p_\theta(y|x)}\left[s(\theta)s(\theta)^T\right] \qquad (8)$$

where $s(\theta) = \nabla_\theta \log p_\theta(y|x)$.

Given the large number of parameters, it becomes challenging to compute the full FIM with a size of $|\theta| \times |\theta|$. Similar to many previous studies, we use the Empirical Fisher Information Matrix (Kunstner et al., 2019), in which FIM is approximated as a diagonal matrix. We define the Empirical Fisher Information Matrix, $\hat{F_\theta}$, as follows:

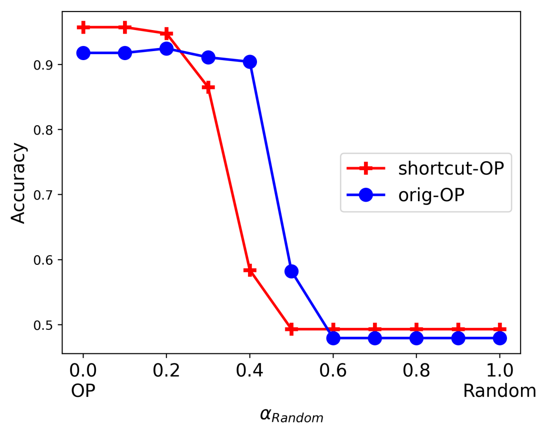
$$\hat{F_\theta} = \frac{1}{N}\sum_{i=1}^{N}\left(\nabla_\theta \log p_\theta(y|x)\right)^2 \qquad (9)$$

where $\hat{F_\theta} \in \mathbb{R}^{|\theta|}$.

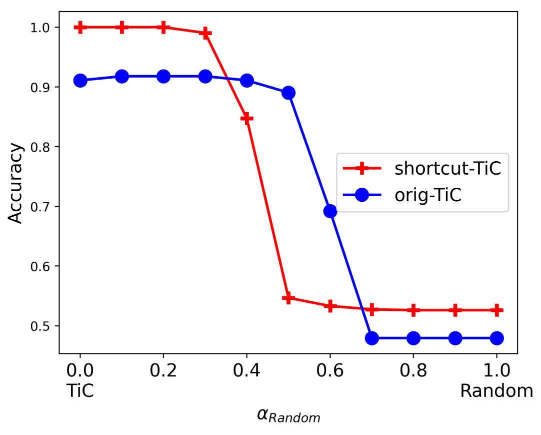
To determine whether similar weights are used in different networks for the knowledge in question, we adopt a metric called Fisher overlap, which measures the degree of overlap between two networks' FIMs by computing Fréchet distance between two networks' FIMs normalized to have a unit trace (Kirkpatrick et al., 2016). More formally, let $\hat{F_{\theta_1}}$ and $\hat{F_{\theta_2}}$ be the corresponding FIMs of the networks with parameters θ_1 and θ_2, and $\overline{F_{\theta_1}}, \overline{F_{\theta_2}}$ be the normalized FIMs to have unit traces. Then, the Fréchet distance is computed as:

$$d^2(\overline{F_{\theta_1}}, \overline{F_{\theta_2}}) = \frac{1}{2}\mathrm{tr}(\overline{F_{\theta_1}} + \overline{F_{\theta_2}} - 2(\overline{F_{\theta_1}}\,\overline{F_{\theta_2}})^{\frac{1}{2}}) \qquad (10)$$

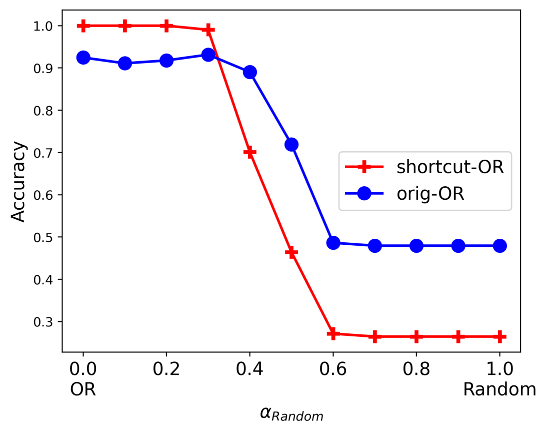
We define the Fisher overlap as $1 - d^2$, where a value of zero means two networks use non-overlapping sets of weights for the questioned knowledges.



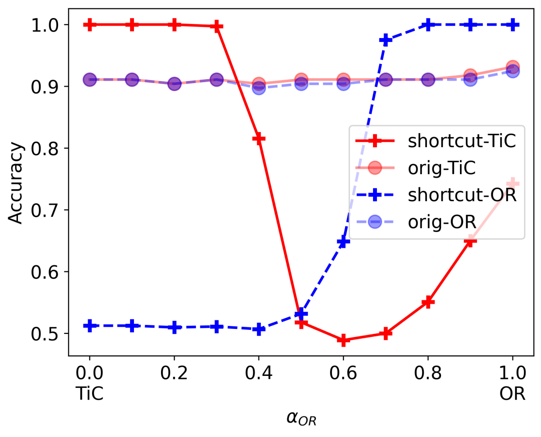
(a) OP → Random



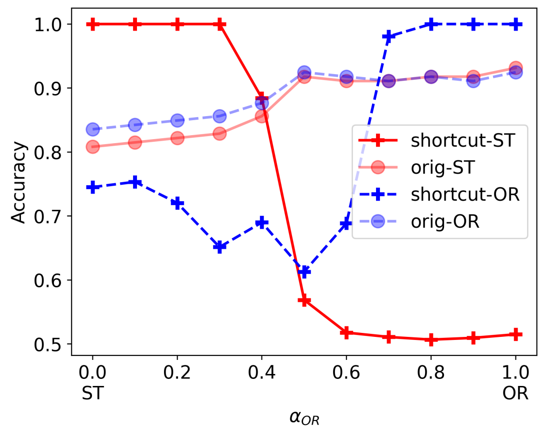
(b) TiC → Random



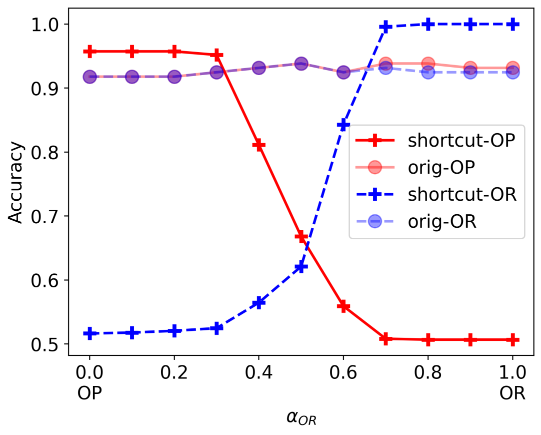
(c) OR → Random



(d) TiC → OR



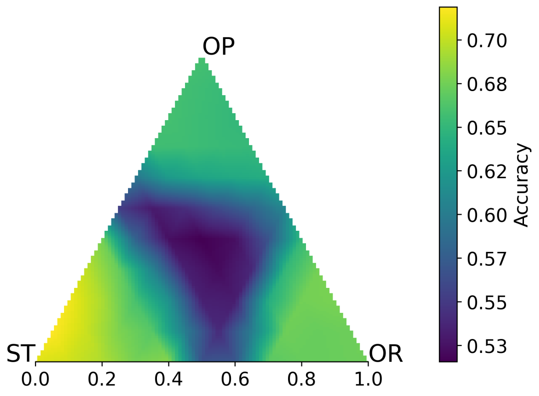
(e) ST → OR



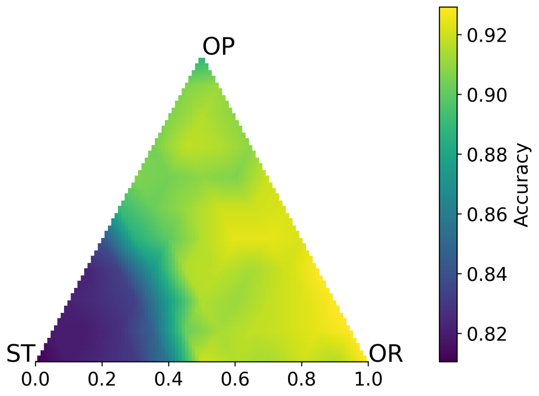
(f) OP → OR

Figure 6: The change of accuracies on synthetic and original validation sets during interpolation between model pairs, each having different shortcuts.

Experimental Setup For this experiment, we chose two model pairs with distinct shortcuts: TiC-OP and TiC-ST. They are chosen to minimize the effects of overlap between shortcuts on the results. For each pair, we independently measure the overlap between the weights used for corresponding shortcuts to determine whether unshared knowledge are administered by different weights. Additionally, we assess the overlap between the weights used for solving the task without shortcuts to investigate whether shared knowledge are administered by the same weights. We select a random subset of the SST2 validation set with $N = 200$ examples. For each shortcut, we create a copy of the selected subset by reversing the original labels and applying the shortcut corresponding to the reversed label.



(a) Results with average shortcut accuracy



(b) Results with average original accuracy

Figure 7: Fused triplets exhibit the same pattern as fused pairs across the surface. The change in accuracy during interpolation among model triplets, each having different shortcuts. (a) Change in average accuracy on synthetic datasets during the interpolation among the models with `ST`, `OP` and `OR` shortcuts (b) Change in average accuracy on original datasets during the interpolation among the models with `ST`, `OP` and `OR` shortcuts.

We reverse labels to ensure that these examples are solely solved using the shortcut knowledge. To measure overlap in the original task, we leave the random subset unchanged.

D.2 Results

Pairs	shared	unshared
`TiC-OP`	**.8077**	.6877
`TiC-ST`	**.7746**	.6819

Table 4: The Fisher overlap between model weights for shared and unshared knowledges

Table 4 reveals a notable distinction between the overlap of weights for shared knowledge and unshared knowledge representing the original task knowledge and shortcuts, respectively in our case. Since shared knowledge tends to be administered by the same set of weights, the simple weight averaging preserves the knowledge while causing unshared knowledge to be forgotten.

E Memorization

E.1 Method

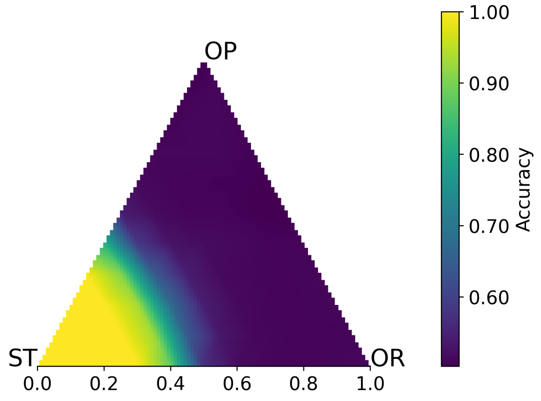
In the memorization evaluation, we adopt the Likelihood Ratio (LR) metric as previously employed by Mireshghallah et al. (2022b). However, their approach to using LR differs slightly from ours. They utilize the percentage of correctly classified training samples by a reference-based membership inference attack proposed by Mireshghallah et al. (2022a) and Carlini et al. (2021). To determine whether a sample x is a member of the training data, they first calculate the Likelihood Ratio (LR) as follows:

$$LR(x) = \frac{p(x;\theta_R)}{p(x;\theta_M)} \tag{11}$$

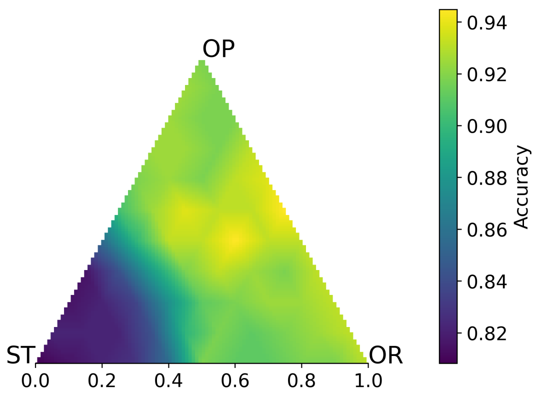
where $p(x;\theta_M)$ and $p(x;\theta_R)$ denote the likelihood of sample x given by the fine-tuned model and the reference model, respectively. Here, the reference model is a pretrained model that is not fine-tuned. They classify the sample as a training set member if $LR(x)$ is smaller than the threshold t, which is chosen by calculating LR for each sample in the validation set and selecting the highest possible threshold that maintains a false positive rate not exceeding 10%. Finally, they measure recall as the final memorization metric, which they refer to as MIA recall. In practice, selecting a threshold based on a non-training set, such as the validation set in this case, works well.

On the other hand, although fused models tend to forget the training data of their seed models, they are still memorized more than a held-out set. Therefore, deciding thresholds on the validation set and measuring MIA recall to assess the memorization of a fused model cannot effectively differentiate the memorization of the fused model from that of the seed models. Consequently, we introduce the Average Likelihood Ratio (ALR) to eliminate the need for selecting a threshold:

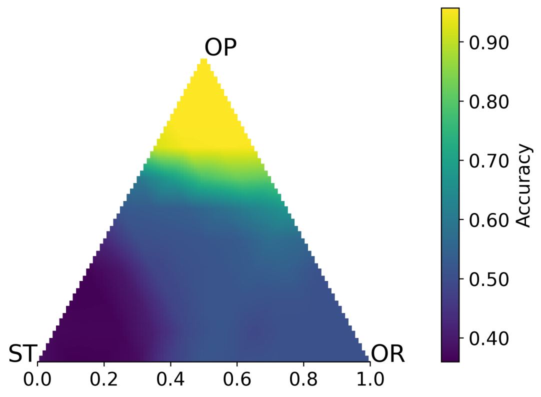
$$ALR(\mathcal{D}) = \frac{1}{|\mathcal{D}|} \exp\left(\frac{p(x;\theta_R)}{p(x;\theta_M)}\right) \tag{12}$$



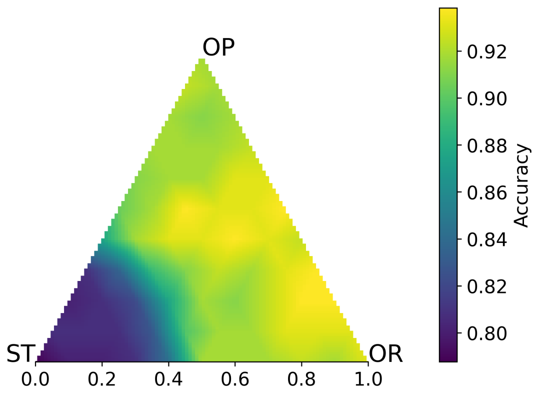
(a) Shortcut accuracy for ST shortcut



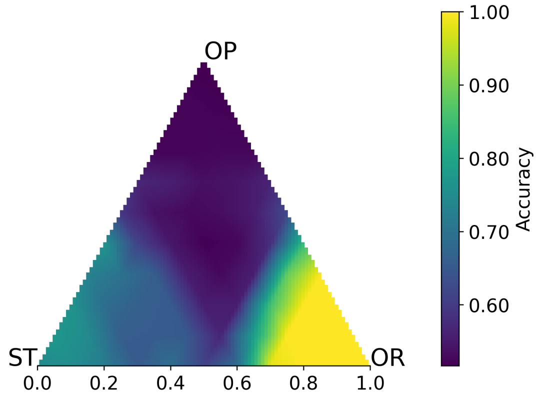
(b) Task accuracy on original validation set of ST



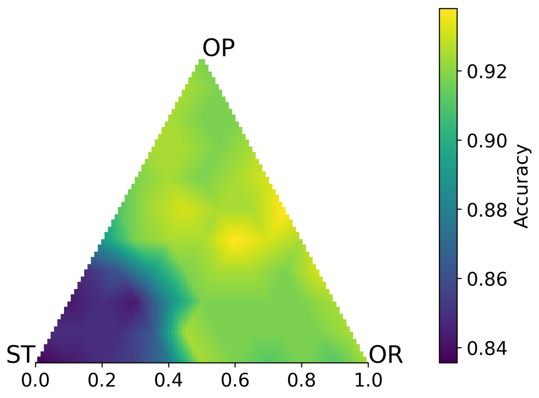
(c) Shortcut accuracy for OP shortcut



(d) Task accuracy on original validation set of OP



(e) Shortcut accuracy for OR shortcut



(f) Task accuracy on original validation set of OR

Figure 8: The change of accuracies on synthetic and original validation sets during interpolation between model pairs, each having different shortcuts.

where $\mathcal{D}$ represents the set of samples on which we test memorization of model.

While measuring LR, we adopt the reparametrization proposed by Mireshghallah et al. (2022a) and also utilized by Mireshghallah et al. (2022b), where they conceptualize pre-trained LMs as energy-based probability distributions on sequences. First, they define the Likelihood Ratio as follows:

$$LR(x) = \log\left(\frac{p(x;\theta_R)}{p(x;\theta_M)}\right) \tag{13}$$

where the target and reference models are parametrized by θ_M and θ_R.

After applying this reparametrization, LR becomes:

$$
\begin{aligned}
LR(x) &= \log\left(\frac{p(x;\theta_R)}{p(x;\theta_M)}\right) \\
&= \log\left(\frac{e^{-E(x;\theta_R)}}{Z_{\theta_R}}\right) - \log\left(\frac{e^{-E(x;\theta_M)}}{Z_{\theta_M}}\right) \\
&= E(x;\theta_M) - E(x;\theta_R) + \text{constant}
\end{aligned} \tag{14}
$$

Since the intractable term $\log(Z_{\theta_M}) - \log(Z_{\theta_R})$ is a global constant, we can ignore it during computation. This parametrization allows us to use the difference between energy values obtained for sample x from the target and reference models.

We follow Mireshghallah et al. (2022a) to determine energy values. For autoregressive language models, the energy is defined as the language modeling loss:

$$E(x;\theta) = -\sum_{t=0}^{T} \log p(x_t|x_{<t};\theta) \tag{15}$$

where T represents the sequence length.

For masked language models, they parameterize energy over 15% chunks that are masked during training. For a sequence of length T and chunk size l, where $l = s\lceil 0.15 \times T \rceil$, with the set $\mathcal{C}$ of all possible l-sized subsets:

$$E(x;\theta) = -\frac{1}{|\mathcal{C}|}\sum_{I\in\mathcal{C}}\sum_{i\in\mathcal{I}} \log p(x_i|x_{\setminus I};\theta) \tag{16}$$

where $x_{\setminus I}$ denotes the sample x with l positions in I masked. Since computing this energy value requires $\binom{T}{l}$ forward passes through the model, they approximate it by summing over $K = 10$ subsets sampled from $\mathcal{C}$.

After applying the aforementioned methods and reparametrizations, our final ALR metric becomes as follows for autoregressive LMs:

$$ALR(\mathcal{D}) = \frac{1}{|D|}\sum_{x\in\mathcal{D}} \exp\left(E(x;\theta) - E(x;\theta_R)\right) \tag{17}$$

where

$$E(x;\theta) = -\sum_{t=0}^{T} \log p(x_t|x_{<t};\theta)$$

for autoregressive models and

$$E(x;\theta) = -\frac{1}{|K|}\sum_{I\sim\mathcal{C}}\sum_{i\in\mathcal{I}} \log p(x_i|x_{\setminus I};\theta)$$

for masked language models.

E.2 Fusing Different Numbers of Models

Table 5 shows the ALRs of each model on all the training sets including the shared subset, and the perplexity scores on validation set. These models consist of the models trained separately on each dataset, fused models with a varying number of seed models and full models trained on combined datasets. Notably, the evaluation of fused models is limited to the validation set and the datasets on which their seed models are trained, while full models are exclusively evaluated on the validation set and the datasets on which they are trained. The results confirm that model fusion reduces memorization and improves generalization.

Figure 9 and 10 illustrate the effects of the number of fused models compared to the full models trained on combinations of datasets used by individual models. As shown by the ALRs in Figure 9, the fused model tends to forget more as the number of models fused increases. Conversely, the increase in data size for full models has no significant effect on the memorization of unshared datasets. Furthermore, we observe that shared datasets are memorized at similar levels, despite a slight decrease, as the number of fused models increases. As expected, they are less memorized by the full model as their percentage in the dataset decreases.

Figure 10 displays perplexity scores on the validation set for both fused and full models. While an increase in the number of models helps with generalization, corresponding full models generalize better as the total data size increases. In all scenarios, the higher perplexity scores than the base model's indicate that all models are overfitted.

Additionally, we investigate the effect of the number of fused models when the total training data size remains constant. We experiment with scenarios where we fuse 2, 3, and 4 models, each trained on 10000 examples, 1000 of which are shared across models. Figure 11 demonstrates that models forget unshared memorized data, while the shared set is increasingly memorized as the number of fused models increases. Figure 12 presents perplexity scores on the validation set for base, fused, and full models. We observe that perplexity increases as the number of fused models increases, unlike in the scenario where the total training data size is proportional to the number of models fused. This increase can be attributed to lower generalization for each model due to decreasing data size per model.

Model	**A**	**B**	**C**	**D**	**shared**	**dev$_{PPL}$**
`gpt-2`	1.0	1.0	1.0	1.0	1.0	23.48
`model`$_A$	**0.217**	1.483	1.483	1.477	0.218	35.25
`model`$_B$	1.502	**0.217**	1.488	1.486	**0.217**	35.81
`model`$_C$	1.486	1.475	**0.220**	1.475	0.219	35.81
`model`$_D$	1.484	1.476	1.478	**0.218**	0.218	35.25
`fused`$_{AB}$	0.485	0.484	-	-	0.233	31.60
`fused`$_{ABC}$	0.656	0.653	0.656	-	0.238	30.63
`fused`$_{ABCD}$	0.758	0.756	0.758	0.755	0.240	30.15
`full`$_{AB}$	0.273	0.274	-	-	0.275	30.15
`full`$_{ABC}$	0.318	0.320	0.320	-	0.321	27.45
`full`$_{ABCD}$	0.353	0.354	0.355	0.355	0.356	**25.79**

Table 5: Extended memorization results: the ALRs of a base model, four models individually fine-tuned models, fused models and full models on each training set including the shared subset along with perplexity scores of all models on the validation set. Lower ALRs denote higher memorization. Bold numbers for ALR shows the lowest ALR, hence highest memorization for a particular dataset among all models except the base model while they show the lowest perplexity for validation set. Underlined numbers represent baseline performance.

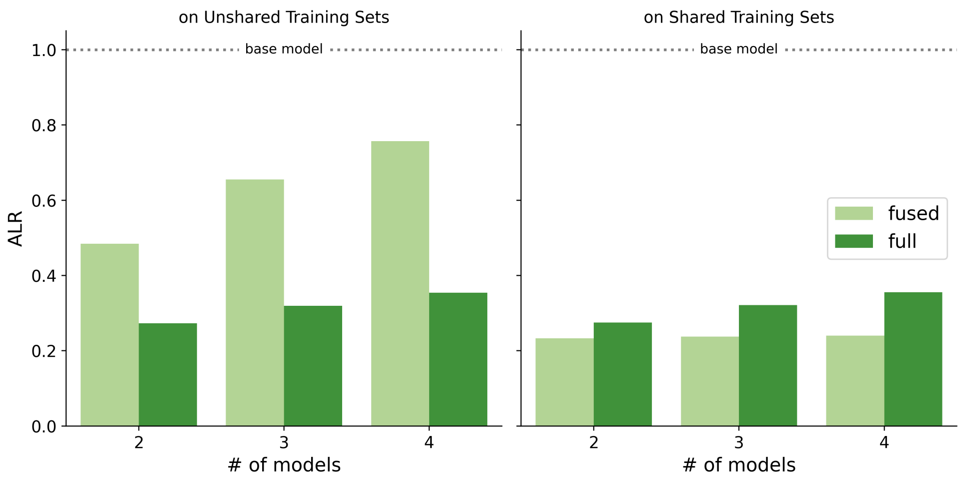


Figure 9: As the number of fused models increases, they memorize less of the unshared data and retain the shared data. The figure depicts the change in ALRs on shared and unshared training sets with respect to the number of fused models, compared to full and base models.

E.3 Fusing Models Trained for Different Numbers of Epochs

Table 6 presents the impact of different choices of number of epochs - 5, 15, and 20 epochs - on memorization. While using a lower number of epochs results in reduced memorization by models, the previously observed conclusions still hold true across all choices of epoch count. Additionally, the generalization gap between full and fused models increases as the number of epochs increases, and the memorization of the shared subset becomes more pronounced when models are not trained for too long.

E.4 Results with BERT

Experimental Setup We fine-tune BERT$_{base}$ models on randomly selected subsets with 3000 news articles (with no sequence packing), 1000 of them shared, from the CNN-DM dataset (Nallapati et al., 2016) for 20 epochs with a batch size of 16, a learning rate of $3e-4$, and no weight decay.

Table 7 replicates Table 5, but with BERT$_{base}$ models fine-tuned instead of GPT-2. The results in Table 7 align with the previous findings, indicating that our claims and observations hold across different architectures.

Model	**A**	**B**	**C**	**shared**	**dev$_{PPL}$**
`gpt-2`	1.0	1.0	1.0	1.0	23.48
			20 epochs		
`model`$_A$	**0.099**	3.263	3.257	0.100	77.00
`model`$_B$	3.309	**0.100**	3.257	**0.099**	77.00
`model`$_C$	3.256	3.219	**0.101**	0.100	77.00
`fused`	0.704	0.699	0.702	0.134	56.33
`full`	0.196	0.197	0.198	0.198	**39.33**
			15 epochs		
`model`$_A$	**0.137**	2.256	2.253	**0.138**	53.75
`model`$_B$	2.295	**0.138**	2.261	**0.138**	54.60
`model`$_C$	2.260	2.236	**0.140**	0.139	53.75
`fused`	0.670	0.667	0.670	0.168	42.52
`full`	0.242	0.243	0.244	0.244	**33.12**
			5 epochs		
`model`$_A$	**0.394**	1.025	1.024	0.396	25.00
`model`$_B$	1.030	**0.394**	1.026	**0.395**	25.00
`model`$_C$	1.028	1.024	**0.398**	0.396	25.00
`fused`	0.684	0.683	0.685	0.396	23.12
`full`	0.461	0.461	0.462	0.464	**22.76**

Table 6: Memorization results with varying number of epochs: the ALRs of a base model, three models individually fine-tuned models, fused models and full models on each training set including the shared subset along with perplexity scores of all models on the validation set. Lower ALRs denote higher memorization. Bold numbers for ALR shows the lowest ALR, hence highest memorization for a particular dataset among all models except the base model while they show the lowest perplexity for validation set. Underlined numbers represent baseline performance.

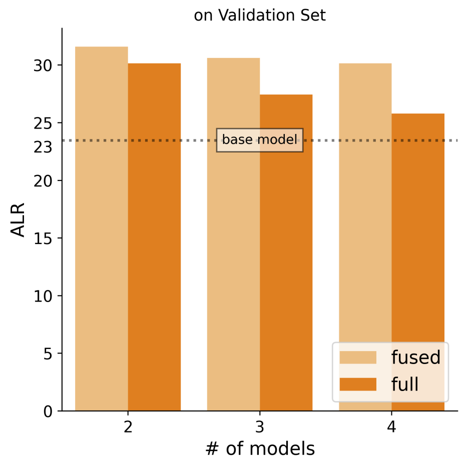


Figure 10: Fused models generalize better as the number of fused models increases but they still lag behind the full models trained on the same amount of data as the individual models trained on combined. The figure shows the perplexity scores on validation sets w.r.t. the number of fused models compared to full and base models.

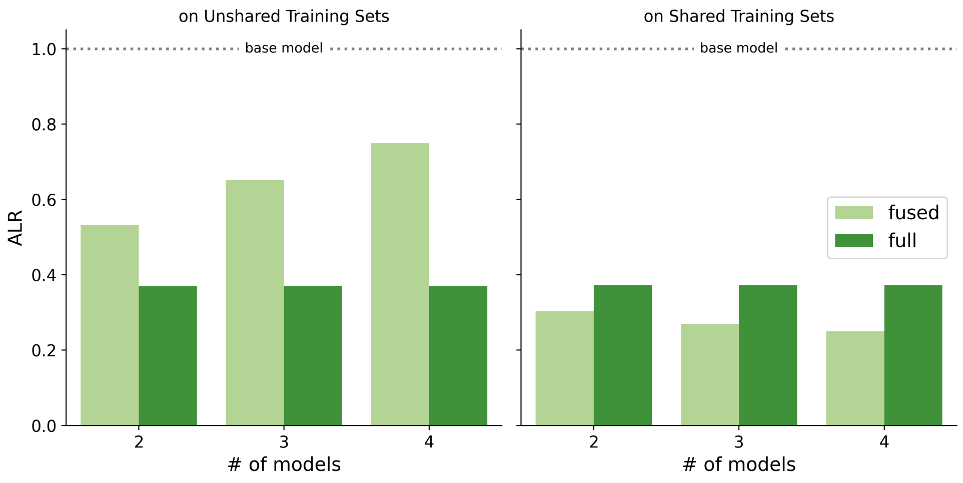


Figure 11: As the number of fused models increases while keeping the total training data size the same, they memorize less of the unshared data and more of the shared data. The figure illustrates the change in ALRs on shared and unshared training sets w.r.t. the number of fused models, compared to full and base models.

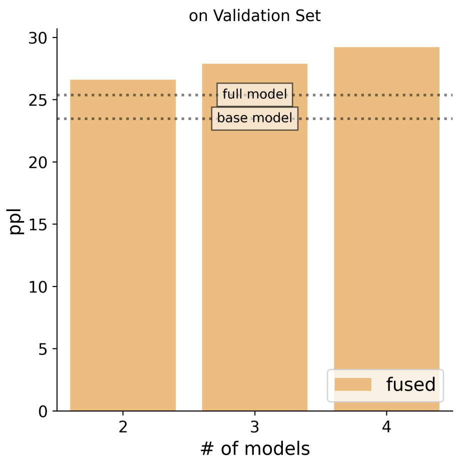


Figure 12: Fused models generalize worse as the number of fused models increases while keeping the total training data size the same. The figure displays the perplexity scores on the validation set w.r.t. the number of fused models compared to full and base models.

Model	**A**	**B**	**C**	**D**	**shared**	**dev**$_{\text{PPL}}$
`bert-base-cased`	<u>1.0</u>	<u>1.0</u>	<u>1.0</u>	<u>1.0</u>	<u>1.0</u>	<u>26.20</u>
`model`$_{\text{A}}$	**0.150**	0.247	0.245	0.249	0.151	5.98
`model`$_{\text{B}}$	0.248	**0.150**	0.244	0.249	0.150	5.98
`model`$_{\text{C}}$	0.248	0.243	**0.147**	0.248	0.149	5.98
`model`$_{\text{D}}$	0.247	0.245	0.241	**0.151**	0.149	5.98
`fused`$_{\text{AB}}$	0.194	0.192	-	-	0.158	6.17
`fused`$_{\text{ABC}}$	0.212	0.211	0.209	-	0.162	6.22
`fused`$_{\text{ABCD}}$	0.234	0.234	0.232	0.236	0.174	6.68
`full`$_{\text{AB}}$	0.153	0.154	-	-	0.155	5.67
`full`$_{\text{ABC}}$	0.158	0.158	0.157	-	0.157	5.49
`full`$_{\text{ABCD}}$	0.161	0.161	0.158	0.161	0.161	**5.36**

Table 7: Memorization results with BERT: the ALRs of a base model, four models individually fine-tuned models, fused models and full models on each training set including the shared subset along with perplexity scores of all models on the validation set. Lower ALRs denote higher memorization. Bold numbers for ALR shows the lowest ALR, hence highest memorization for a particular dataset among all models except the base model while they show the lowest perplexity for validation set. Underlined numbers represent baseline performance.