

# GRADIENT ROUTING: MASKING GRADIENTS TO LOCALIZE COMPUTATION IN NEURAL NETWORKS

**Anonymous authors**

Paper under double-blind review

## ABSTRACT

Neural networks are trained primarily based on their inputs and outputs, without regard for their internal mechanisms. These neglected mechanisms determine properties that are critical for safety, like (i) transparency; (ii) the absence of sensitive information or harmful capabilities; and (iii) reliable generalization of goals beyond the training distribution. To address this shortcoming, we introduce *gradient routing*, a training method that isolates capabilities to specific subregions of a neural network. Gradient routing applies data-dependent, weighted masks to gradients during backpropagation. These masks are supplied by the user in order to configure which parameters are updated by which data points. We show that gradient routing can be used to (1) learn representations which are partitioned in an interpretable way; (2) enable robust unlearning via ablation of a pre-specified network subregion; and (3) achieve scalable oversight of a reinforcement learner by localizing modules responsible for different behaviors. Throughout, we find that gradient routing localizes capabilities even when applied to a limited, ad-hoc subset of the data. We conclude that the approach holds promise for challenging, real-world applications where quality data are scarce.

## 1 INTRODUCTION

As AI systems become more powerful and more prevalent, there is an increasing need to explain and control the inner mechanisms governing their behavior. To address this challenge, some researchers aim to fully understand AI systems, either by reverse engineering the operations of conventionally trained models (Olah et al., 2020; Olsson et al., 2022) or with inherently interpretable architectures (Koh et al., 2020; Hewitt et al., 2023; Xin et al., 2022). This is not necessary. If we could understand or control the mechanisms underlying a neural network’s computation with respect to a limited set of safety-critical properties, such as hazardous information or the capacity for deception, that might be sufficient to make significant safety guarantees.

To achieve targeted control over neural network internals, we propose gradient routing, a training method for localizing capabilities to chosen subregions of a neural network. Gradient routing is a modification of backpropagation that uses data-dependent, weighted masks to control which network subregions are updated by which data points. By appropriately specifying these masks, a user can configure which parts of the network (parameters, activations, or modules) are updated by which data points (e.g. specific tokens, documents, or based on data labels). The resulting network is similar to a conventionally trained network, but with some additional internal structure.

Our contributions are as follows. In **Section 2**, we discuss prior work on neural network modularity, unlearning, and scalable oversight. In **Section 3**, we define gradient routing and comment on its practical implementation. Most of the paper is a tour of gradient routing applications:

**Section 4.1** We use gradient routing to control the encodings learned by an MNIST autoencoder to split them into two halves, with each half representing different digits.

**Section 4.2** We apply gradient routing to localize features in language models. First, we train a model that can be steered by a single scalar value, showing that feature localization is possible, even with narrowly-scoped labels. Next, we present *Expand, Route, Ablate*, an application of gradient routing that enables robust unlearning via ablation of a pre-specified network subregion. This unlearning is nearly as resistant to retraining as a gold-standard model never trained on the task. Finally, we show that this unlearning method scales to a large (0.7B) model.

**Section 4.3** We apply gradient routing to the problem of scalable oversight (Amodei et al., 2016), where the aim is to train a performant policy despite limited access to reliable labels. We train a policy network by reinforcement learning to navigate to two kinds of grid squares in a toy environment, DIAMOND and GHOST. Using gradient routing, we localize modules responsible for these two behaviors. We show that we can steer the policy towards DIAMOND by ablating the GHOST module. Gradient routing trains steerable networks even when the amount of labeled training data is small, and even when the policy is able to condition on the existence of labels. As a result, our method outperforms baselines, including data filtering.

In **Section 5**, we discuss themes from our findings, including an observed **absorption** effect, where gradient routing applied to a narrow subset of data has a broader localizing effect on capabilities related to that data. Absorption provides an answer to the question: “If one has labels that are suitable for localizing undesirable computation, why not simply use those labels to filter the data?” When labels do not encompass all training data from which harmful capabilities arise, absorption means that localization can still occur, whereas filtering may be inadequate. Furthermore, localization does not explicitly influence the learned behavior of a model, a fact we exploit to achieve scalable oversight.

We conclude by noting that black-box training techniques may be inadequate for high-stakes machine learning applications. Localization techniques, like gradient routing, may provide a solution.

## 2 RELATED WORK

**Training to localize pre-specified capabilities.** Akin to gradient routing, work in modular machine learning trains modules to contain concepts or abilities determined in advance of training. Typically, modular architectures involve a routing function that selects modules to apply on a forward pass (Pfeiffer et al., 2023). Routing functions are often unsupervised, as with a typical mixture of experts setup (Jacobs et al., 1991; Eigen et al., 2013; Shazeer et al., 2017). However, some approaches route inputs based on metadata, creating modules with known specializations (Waibel & II, 1992). For example, routing has been based on (i) the modality of data in multi-modal models (Pfeiffer et al., 2021), (ii) language (Pfeiffer et al., 2020; 2022; Fan et al., 2021), and (iii) low- vs. high-level control or task type in robotics (Heess et al., 2016; Devin et al., 2017). Gururangan et al. (2021) separate the training data of a language model by domain and assign one expert in each layer to a single domain. By disabling the expert for a domain, they are able to approximate a model that was not trained on the domain.

Other methods freeze the weights of a pre-trained model and train a newly added module, with the aim of localizing the task to the new module (Rebuffi et al., 2017; 2018; Houlsby et al., 2019; Bapna & Firat, 2019). Zhang et al. (2024) locate capabilities in models by learning a weight mask, transfer the identified sub-network to a randomly initialized model, then train as if from scratch. By choosing a suitable sub-network, they can, for example, induce a vision model to identify ImageNet (Deng et al., 2009) classes by shape, not texture.

**Adversarial representation learning and concept erasure.** In order to control the information in learned representations, prior works have trained feature extraction networks adversarially against discriminator networks that predict this information (Goodfellow et al., 2014; Schmidhuber, 1992; Ganin & Lempitsky, 2015; Ganin et al., 2016; Edwards & Storkey, 2015). Other works have removed concepts by modifying activations during inference (Ravfogel et al., 2020; Belrose et al., 2023; Elazar et al., 2020; Bolukbasi et al., 2016).

**Robust unlearning.** Machine unlearning seeks to remove undesired knowledge or abilities from a pre-trained neural network (Cao & Yang, 2015; Li et al., 2024). Typical unlearning methods are brittle in the sense that the unlearned abilities of the model can be recovered by fine-tuning on a tiny number of data points (Henderson et al., 2023; Sheshadri et al., 2024; Lynch et al., 2024; Liu et al., 2024; Shi et al., 2024; Patil et al., 2023; Lo et al., 2024; Lermen et al., 2023). Lee et al. (2024); Lucki et al. (2024) suggest that undesired concepts are more easily “bypassed” than thoroughly removed from model weights. In this paper, we pre-train models with gradient routing such that we can perform *robust* unlearning, which cannot be easily undone by retraining. Tampering Attack Resistance (TAR) (Tamirisa et al., 2024) also targets robust unlearning in LLMs. While their method does improve robustness to retraining, it degrades general model performance as a

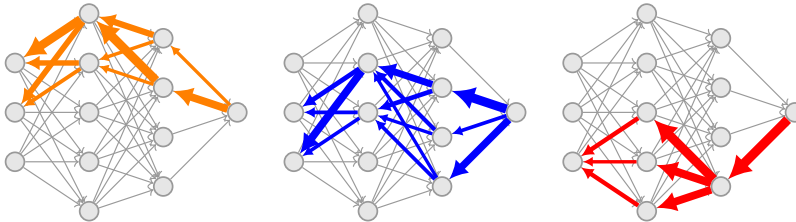


Figure 1: Gradient routing applied to three different data points. Gradient routing uses masks (shown here) to zero-out or re-weight gradients as they are backpropagated through the computational graph (denoted by arrow thickness). By supplying different masks for different data points (denoted by colors), the user controls what regions of the network learn to predict different data points.

side effect. However, we present gradient routing as a training technique rather than a post-hoc modification, so the two methods aren’t directly comparable.

Compared to gradient routing, the most similar approaches prune or mask parts of the network most important for the target behavior. SISA (Bourtole et al., 2021) trains multiple models in parallel on partitions of the dataset and takes “votes” from each model at inference time. Similar to ablating a network subregion, a model can be dropped to achieve robust unlearning. The approach of Bayazit et al. (2023) is to learn a mask over parameters in a language model to unlearn specific facts, while Huang et al. (2024) and Pochinkov & Schoots (2024) remove neurons related to harmful behavior in order to restore the alignment of an adversarially fine-tuned language model. Guo et al. (2024) fine-tune the parameters of only the most important components for the task. Lizzo & Heck (2024) instead delete subspaces of the model parameters in order to remove specific knowledge. Unfortunately, Lo et al. (2024) find that models pruned to remove a concept can very quickly relearn the concept with further training. This may be because *identifying* the precise sub-network for a task post-hoc is very challenging, as evidenced by the modest success of “circuit discovery” in mechanistic interpretability thus far (Wang et al., 2023; Conmy et al., 2023; Miller et al., 2024; McGrath et al., 2023).

**Scalable oversight.** Scalable oversight is the problem of providing a supervised training signal for behaviors that are difficult or expensive to assess (Amodei et al., 2016). Semi-supervised reinforcement learning frames scalable oversight in terms of RL on partially labeled data (Zhu et al., 2009; Finn et al., 2016; van Engelen & Hoos, 2019). Another approach is weak-to-strong generalization, in which a less powerful model provides supervision to a more powerful one (Burns et al., 2024; Kenton et al., 2024; Radhakrishnan et al., 2023). Weak-to-strong generalization introduces a potential risk: the stronger model may exploit blind spots in the weaker model’s supervision capabilities.

### 3 GRADIENT ROUTING CONTROLS WHAT IS LEARNED WHERE

In order to train models that are interpretable and controllable in targeted ways, we seek to localize specific knowledge or capabilities to network subregions. To do this, **gradient routing** applies data-dependent, weighted masks to gradients during backpropagation to configure **what** data (whether it be defined in terms of tokens, documents, or based on other labels) is learned **where** in the network (e.g. at the level of parameters, activations, or modules). The result is a model with a partially-understandable internal structure, where particular regions correspond to known capabilities. *Throughout this paper, we will use “route  $X$  to  $Y$ ” to mean “use gradient routing to limit learning updates for data points  $X$  to region  $Y$  of the neural network.”*

Let  $(\mathcal{V}, \mathcal{E})$  be the nodes and edges of the computational graph corresponding to a neural network and loss function, with  $v(z)$  taken to be the output of node  $v$  if  $z$  is input to the network. Given a dataset  $\mathcal{D} = \{z_i\}_{i=1}^n$ , for each data point  $z_i$ , gradient routing requires the specification of a **gradient route** given by  $\mathcal{E}_i = \{\alpha_e^i \in \mathbb{R} : e \in \mathcal{E}\}$  and visualized in fig. 1. Define  $\frac{\partial L(z)}{\partial v} \triangleq \frac{\partial L(\zeta)}{\partial v(\zeta)}|_{\zeta=z}$ , the partial derivative of the loss  $L$  with respect to the output of node  $v$  when evaluated at input  $z$ . The routed derivative (denoted with a tilde) of the loss over a batch  $\mathcal{B} \subseteq [n]$  is then defined recursively

162 as  $\frac{\tilde{\partial}L(z_i)}{\tilde{\partial}L} \triangleq 1$  for all  $i \in \mathcal{B}$ , and

$$163 \frac{\tilde{\partial}L(z_i)}{\tilde{\partial}v} \triangleq \sum_{u \in \text{child}(v)} \alpha_{(v,u)}^i \frac{\tilde{\partial}L(z_i)}{\tilde{\partial}u} \frac{\partial u(z_i)}{\partial v},$$

164 for all non-terminal nodes  $v \in \mathcal{V} \setminus \{L\}$  and  $i \in \mathcal{B}$ . Choosing  $\alpha_e^i \equiv 1$  recovers standard backprop-  
 165 agation. This weighting is only applied in the backward pass; the forward pass is left unchanged.  
 166 Any gradient-based optimizer, like SGD or Adam (Kingma, 2014), can then be used to train with  
 167 these modified gradients.

171 It is costly to define gradient routes over every data point and edge in the computational graph. In  
 172 practice, we limit masks to a small set of edges, like the outputs of specific MLP neurons or the out-  
 173 puts of specific layers. Also, we typically assign gradient routes to data points based on membership  
 174 in a coarse partition, like the forget set or retain set in an unlearning problem. Implementation is  
 175 straightforward and efficient: algorithm 1 gives sample Pytorch (Paszke et al., 2019) code in which  
 176 masking is applied to the outputs of sequential layers.

```
178
179 def forward(self, x: Tensor, gradient_masks: list[Tensor]):
180     for layer, mask in zip(self.layers, gradient_masks):
181         act = layer(x)
182         x = mask * act + (1 - mask) * act.detach()
183     return x
```

184 Algorithm 1: Example of gradient routing implemented in PyTorch. The `detach()` method applies  
 185 the stop-gradient operator, preventing gradients from being backpropagated through `act` but leaving  
 186 its value unchanged. For each batch of training data points `x`, a batch of `gradient_masks` corre-  
 187 sponding to those data points is passed as well.

## 189 4 APPLICATIONS

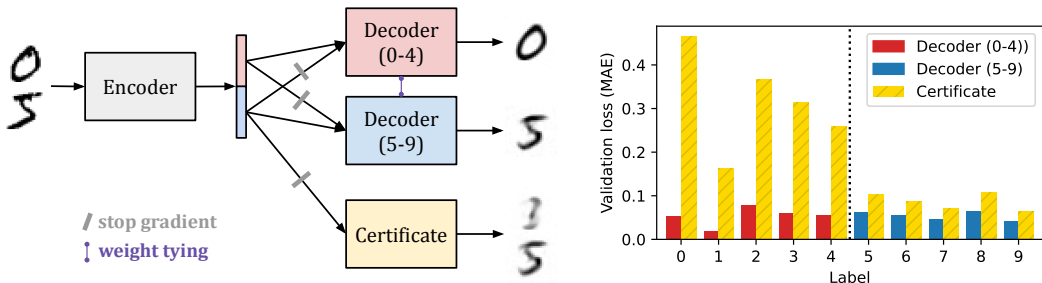
### 192 4.1 ROUTING GRADIENTS TO PARTITION MNIST REPRESENTATIONS

193 As a first example of feature localization via gradient routing, we train a simple MLP autoencoder  
 194 on the MNIST handwritten digit dataset (LeCun et al., 1998) and use label-dependent stop-gradients  
 195 to control where features for different digits are encoded. The goal is to obtain an autoencoder  
 196 that reconstructs all digits (0-9) via an encoding that is made up of non-overlapping subcomponents  
 197 corresponding to distinct subsets of digits. We choose subsets  $\{0, 1, 2, 3, 4\}$  and  $\{5, 6, 7, 8, 9\}$ . To  
 198 hint at the potential difficulty of this task, we note the encodings learned by an autoencoder trained  
 199 on one of these sets admit low-error reconstructions on the other set, despite never being trained on  
 200 it (details in appendix B).

201 We use a simple architecture of three-layer MLP modules with ReLU activations: an Encoder, a  
 202 Decoder, and a Certificate. The Encoder processes a  $28 \times 28$  image into a vector in  $\mathbb{R}^{32}$ , and the  
 203 Decoder processes that vector into a  $28 \times 28$  reconstruction. The Certificate is a decoder trained only  
 204 on the *bottom half* of the encoding, which takes values in  $\mathbb{R}^{16}$ . If the Decoder can reconstruct a digit  
 205 that the Certificate cannot, this “certifies” that robust feature localization occurred (into the top half  
 206 encoding, and away from the bottom half).

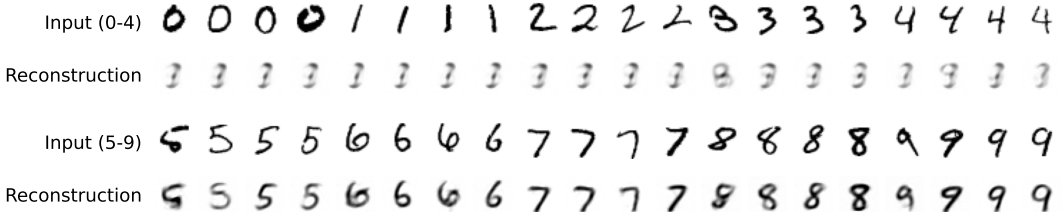
207 We use gradient routing to train an encoding split so that the top half encodes digits 0-4 and the  
 208 bottom half encodes digits 5-9. While training on all digits, we apply stop-gradients to the bottom  
 209 half of the encoding for digits 0-4 and stop-gradients to the top half of the encoding for digits 5-9.  
 210 To induce specialization in the two halves of the encoding, we add the L1 norm of the encoding  
 211 as a penalty term to the loss. The setup is shown in fig. 2a. The results, shown in fig. 2b and  
 212 fig. 2c, are stark: while using the entire encoding allows the Decoder to reproduce all digits with  
 213 low loss, the Certificate is only able to reproduce 5-9 from the bottom half of the encoding, as  
 214 desired. Furthermore, the Certificate’s learned predictions for digits 0-4 are approximately constant.  
 215 This suggests that we have successfully eliminated most information relevant to digits 0-4 from the  
 encoding. We elaborate on experiment details and provide an extensive ablation study in appendix B.

216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269



(a) An autoencoder trained to encode digits 0-4 in the top half encoding and digits 5-9 in the bottom half. The full encoding is processed by a single Decoder module trained with gradient routing; we illustrate this using weight tying. The Certificate is trained to reconstruct all digits using only the bottom half encoding.

(b) Average (across 20 runs) validation set reconstruction losses, measured as the pixel-wise mean absolute error (MAE) for the Decoder and the Certificate, demonstrating successful localization of information about digits 0-4 to the top half encoding. Run-to-run variation is negligible.



(c) Certificate reconstructions on digits from the validation set. The near-constant prediction of the Certificate on digits 0-4 illustrates the absence of information about those digits from the bottom half of the encoding.

Figure 2: Gradient routing induces a clean split in the encodings of a simple MLP autoencoder trained on MNIST digits. By applying data-dependent stop-gradients and L1 regularization, the top half of the encoding comes to represent digits 0-4 only, and the bottom half of the encoding comes to represent digits 5-9 only.

## 4.2 LOCALIZING TARGETED CAPABILITIES IN LANGUAGE MODELS

In this section, we show that gradient routing applied to a small set of tokens can be used to localize broader features or capabilities in Transformer (Vaswani, 2017) language models. This is first demonstrated in terms of model activations, then applied to MLP layers for the purpose of robust unlearning.

### 4.2.1 STEERING SCALAR: LOCALIZING CONCEPTS TO RESIDUAL STREAM DIMENSIONS

Elhage et al. (2021) frames the inter-block activations of a Transformer, or *the residual stream*, as the central communication channel of a Transformer, with all layers “reading from” and “writing into” it. Usually, the standard basis of the residual stream is indecipherable, with dimensions not corresponding to interpretable concepts. We pre-train a 303M parameter Transformer on the FineWebEdu dataset (Penedo et al., 2024) while routing the gradients for all `_California`<sup>1</sup> tokens to the 0<sup>th</sup> entry of the residual stream on layers 6-18. On token positions predicting `_California`, we mask gradients (to zero) on every residual stream dimension except the 0<sup>th</sup> in layers 6-18. This masking causes the learning updates for those token positions to be localized to the weights that write into the 0<sup>th</sup> dimension of the residual stream. After training, we look at which tokens’ unembedding vectors have the highest cosine similarity with the one hot vector for the 0<sup>th</sup> entry of the residual stream. We find that `_California` has the highest cosine similarity, followed by `California`, `_Californ`, `_Oregon`, `_Colorado`, `_Texas`, `_Florida`, `_Arizona`, `_Sacramento`, and `_Los`; see appendix D for the top 300. These tokens all have semantic similarity to California, but gradient

<sup>1</sup>We use a leading `_` to represent a leading space before a token.

270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323

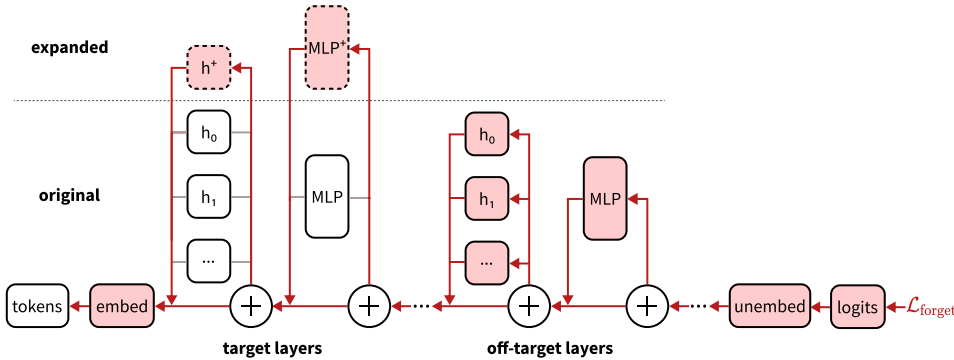


Figure 3: Backpropagation in the Route step of Expand-Route-Ablate, showing the flow of gradients through a Transformer for tokens in the forget set. Gradients for retain tokens are unmodified. Additional dimensions, shown with dashed outlines, were added to **target** layers in the MLP and attention blocks, and will be removed after training in the Ablate step. All modules participate in the forward pass.

routing was not applied to them. This shows that gradient routing localizes broader semantic concepts, rather than the narrow set of explicitly-routed tokens.

Past work on activation steering (Turner et al., 2023; Rinsky et al., 2024) computed (non-axis aligned) *steering vectors* specified by  $d_{\text{model}}$  different values. However, since we localized California-related concepts to the 0<sup>th</sup> dimension of the residual stream, we can steer the model to generate text related to California by adding a single scalar value to the 0<sup>th</sup> entry of the residual stream during the forward pass. Appendix D provides steered model completions.

#### 4.2.2 GRADIENT ROUTING ENABLES ROBUST UNLEARNING VIA ABLATION

Robust unlearning (Sheshadri et al., 2024) means training models which lack the internal mechanisms or “knowledge” required for certain tasks, as opposed to merely performing poorly on those tasks. To address this open problem, we show that gradient routing can be used to localize capabilities to a known region of the network, then delete that region, removing those capabilities.

To enable comprehensive comparisons, our initial study on robust unlearning applies gradient routing to a small (28M parameter) Transformer. This model is trained on an LLM-generated dataset of simple children’s stories based on the TinyStories dataset (Eldan & Li, 2023; Janiak et al., 2024). We partition the data into: 1) a **forget set** made up of any story containing one of the keywords “forest(s)”, “tree(s)”, or “woodland(s)”, and 2) a **retain set** made up of all other stories. An example story is given in appendix C. The goal is to train a model that performs well on the retain set but poorly on the forget set, and whose forget set performance is not easily recoverable by fine-tuning. To do this, we route specific forget tokens to designated MLP neurons using three-step process termed Expand, Route, Ablate (ERA):

1. **Expand** Increase the dimensionality of the model by adding randomly-initialized neurons to particular *target layers*.
2. **Route** Train the model by supervised learning on next-token prediction, but on select tokens in forget stories, reduce the learning rate in the original dimensions of the model at the target layers. Figure 3 illustrates the routing step.
3. **Ablate** Delete the additional neurons. Post-ablation, apply a very small number of steps of fine-tuning on retain data to correct for degradation caused by ablation.

Does gradient routing localize capabilities that can be robustly ablated? To answer this question, we train five types of models: an *ERA* model that uses gradient routing to localize forget set concepts, a *base* model trained conventionally on all data, a *pure* model trained only on retain data to serve as a gold standard, a *control* model trained equivalently to ERA except without gradient routing<sup>2</sup> and

<sup>2</sup>The control model is expanded, ablated, and fine-tuned. It uses a small L1 penalty (small in the sense that it has no measurable effect on loss; see appendix C) on the MLP activations in the target layers.

an *RMU* model (Li et al., 2024) fine-tuned from the base model to serve as an unlearning baseline. Using these models, we obtain the following results. Approximate 95% confidence intervals for the mean ( $N = 60$  runs) are given in parentheses or highlighted in figures.

- **Shallow unlearning** measures the degradation in forget loss caused by our method by comparing the loss on the forget set for the ERA model vs. the base model. Result: ERA achieves shallow unlearning, with forget loss of 1.91 ( $\pm 0.05$ ), vs. base model forget loss 1.47 ( $\pm 0.02$ ).
- **Robust unlearning** measures the robust removal of capabilities by comparing the forget loss after fine-tuning on forget data for the ERA model vs. the pure model. Figure 4a shows that the ERA model is almost as hard to retrain on forget data as the gold-standard pure model. In contrast, *RMU*'s performance is easily recovered by less than a batch of data. We comment on the choice of *RMU* as a baseline in appendix C.
- **Alignment tax** measures the cost of gradient routing in terms of retain set performance, by comparing the loss on the retain set for the ERA model vs. the base model. Result: 1.67 ( $\pm 0.01$ ) ERA, 1.59 ( $\pm 0.01$ ) base. The reduced performance of ERA is influenced by the prevalence of forget data, which constitutes 21% of the training data. In fig. 10, we show that the performance gap is negligible when the forget set constitutes as much as 5% of the training data.
- **The differential effect of routing** measures the impact of ablation on the ERA model vs. the control model. Figure 4b shows that ablation has a large effect on the control model, particularly in terms of increasing forget loss, as compared to a negligible effect on the control model.

Losses are always calculated on held-out validation data. ERA setup and experiment details are given in appendix C.

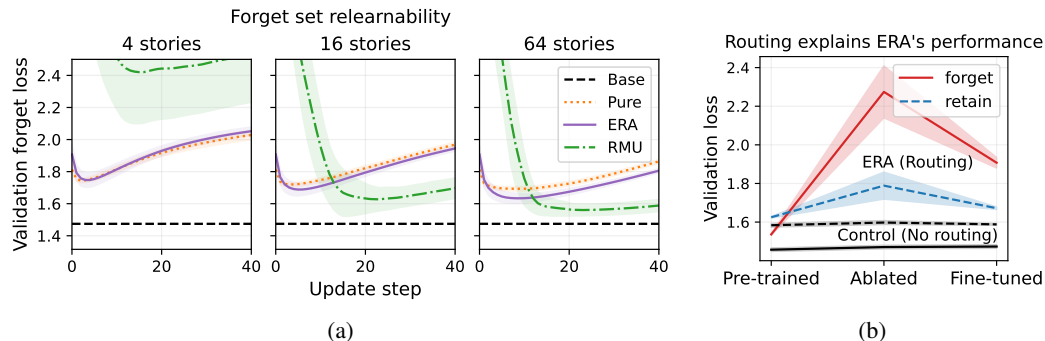


Figure 4: Gradient routing trains a language model with subcomponents that can be ablated to achieve robust unlearning. *Left:* (a) The average (across runs) validation forget set loss for the ERA model and pure model over 40 steps of fine-tuning on batches of varying numbers of forget data points: 4, 16, and 64. *RMU* is included as a foil to show the “shallowness” typical of most unlearning methods. *Right:* (b) Average forget and retain set validation loss after training, after ablation, and after fine-tuning for ERA vs. a control model that is exactly equivalent except gradients were not modified. Note: the  $x$ -axis is not to scale; pre-ablation training is on 400,000 stories, ablation is immediate, and fine-tuning is on 64 stories.

**Robustness to missing labels.** Finally, we show that ERA achieves robust unlearning even when only a random i.i.d. subset of forget samples are labeled, with unlabeled samples treated as retain data for training purposes. This is meant to model real-world scenarios where comprehensive labeling is infeasible. Compared against a conventionally trained model that does not train on labeled forget data, ERA achieves (on average, across runs) higher *retrained* validation forget loss for all labeling proportions measured besides 100%. For example, at 60% labeling, ERA achieves a retrained forget loss of 1.53 ( $\pm 0.02$ ) as opposed to 1.49 ( $\pm 0.02$ ) for the baseline. Full curves are shown in Figure 8 in Appendix C.1.

#### 4.2.3 SCALING ROBUST UNLEARNING TO LARGER LANGUAGE MODELS

Gradient routing can localize capabilities in larger models. Motivated by the dual-use nature of AI (Urbina et al., 2022), we would like to train useful models that lack certain harmful capabilities. Here, we seek to localize and remove bioweapon-related capabilities in a 0.7B parameter Trans-

Table 1: Performance of a language model trained with gradient routing on virology tokens. The final column evaluates the model after finetuning on FineWeb-Edu and then retraining on two examples from the WMDP-bio forget set, choosing the retraining step with the lowest loss. The increase in loss on (the validation split of) the WMDP-bio forget set is much higher than the increase in loss on FineWeb-Edu data, demonstrating successful localization and robust unlearning. Intriguingly, this increase persists even when excluding routed tokens from the loss calculation, showing a broader localizing effect.

Dataset	Loss	Ablated loss ( $\Delta$ )	Retrained loss ( $\Delta$ )
WMDP-bio forget set $\uparrow$	2.596	4.283 (+1.687)	2.778 (+0.182)
WMDP-bio forget set (sans routed toks) $\uparrow$	2.567	4.205 (+1.638)	2.738 (+0.171)
FineWeb-Edu $\downarrow$	2.925	4.864 (+1.939)	2.957 (+0.032)

former. To do this, we route 20 tokens related to virology<sup>3</sup> to the 0<sup>th</sup> through 79<sup>th</sup> MLP dimensions on layers 0 through 7 of the Transformer. Appendix E provides further details on the model and training.

Table 1 evaluates the model on a validation split of regular FineWeb-Edu data and on some of the WMDP-bio (Li et al., 2024) forget set. Ablating the target region of the network increases loss greatly on both datasets. We then fine-tune the model on a train split of FineWeb-Edu for 32 steps to restore some performance. Finally, we retrain for twenty steps on a separate split of two WMDP-bio forget set datapoints, as Sheshadri et al. (2024) do, and report the lowest loss on the validation split of the WMDP-bio forget set.

The results are striking: even after retraining on virology data, loss increases much more on the WMDP-bio forget set (+0.182) than on FineWeb-Edu (+0.032), demonstrating successful localization and robust removal of virology capabilities. A natural concern would be that ablation merely decreased probabilities on the routed tokens, without decreasing overall virology capabilities. To test this, we measured cross-entropy loss on the forget set excluding the 20 tokens we routed on. Even after this exclusion, the loss increase is still much higher than the increase on FineWeb-Edu (+0.171 vs. +0.032). This shows that gradient routing generalizes beyond limited labels.

### 4.3 LOCALIZING BEHAVIORAL MODULES ENABLES SCALABLE OVERSIGHT IN REINFORCEMENT LEARNING

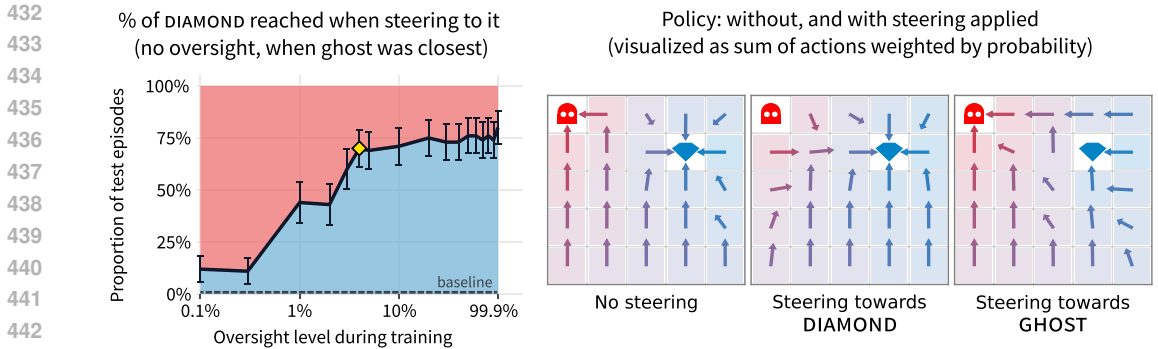
In complex settings, reliable data labels are often scarce, especially when labeling requires human input (Stiennon et al., 2020; Bai et al., 2022; Baker et al., 2022). *Scalable oversight* (Amodei et al., 2016) means effectively utilizing limited labels to obtain a performant policy. In this section, we show that gradient routing’s effective use of limited labels enables scalable oversight, outperforming an outcome-based baseline and a data filtering baseline.

We use gradient routing to train a policy to reach two types of squares in a gridworld, DIAMOND and GHOST. Access to the type of square reached, or *oversight*, is only sometimes available. The aim is to produce a policy that reliably navigates to DIAMOND and not GHOST, even when oversight is lacking. In real-world problems, label availability may be correlated with task type or difficulty, and a policy might not generalize correctly because of this fact (Langosco et al., 2022). To model this fundamental difficulty, *we allow the policy to condition on the presence or absence of oversight*. During training, we route policy gradients through two separate modules in a mixture of experts layer, each responsible for reaching one of the aforementioned squares. During evaluation, by ablating the GHOST module, we can steer the policy toward reaching DIAMOND even when oversight is lacking.

**Environment.** The environment is a  $5 \times 5$  gridworld Markov Decision Process (Puterman, 1990) with two randomly placed terminal squares: DIAMOND and GHOST. At each step, the agent can move north, west, south, or east. Episodes end upon reaching a terminal square or after 256 steps.

<sup>3</sup>Specifically, we route on COVID, \_COVID, RNA, \_infections, DNA, \_genome, \_virus, \_gene, \_viruses, \_mutations, \_antibodies, \_influenza, \_bacteria, PCR, \_cell, \_herpes, \_bacterial, \_pathogens, \_tumor, and \_vaccine.





(a) Proportion of episodes in which our agent reached DIAMOND after being steered to it. A diamond indicates the case of 4% oversight level, where DIAMOND is reached 70% of the time. (Ranges: 95% C.I.) (b) The gradient-routed policy in an instance of the environment. Steering induces qualitatively different behaviors. Arrows: N/W/S/E actions weighted by probability. Left: Unsteered policy. Middle: Steered towards DIAMOND. Right: Steered towards GHOST. The policy was trained with 10% oversight.

Figure 5: Using gradient routing and a mixture of experts layer, we train an agent that can be steered (right) towards desirable behaviors, even when oversight is severely limited (left). The values reported are from 100 randomly generated “challenging” environments (GHOST not under oversight and closer than DIAMOND). The dashed line corresponds to a baseline trained naively on all data, which learns to navigate to GHOST when oversight is lacking. The cases of 0% and 100% oversight lie outside of the realm of scalable oversight, so we omit them.

The environment state includes a boolean indicator variable for every grid square, indicating whether oversight is present. These indicators are randomly sampled at the beginning of each episode. If an episode ends with the agent reaching a terminal grid square with the oversight indicator set to FALSE, then the learning algorithm does not have access to the square type labels (DIAMOND, GHOST) for that entire episode. The (true) rewards, which are not always observed, are +1 for reaching DIAMOND, -1 for reaching GHOST, and 0 otherwise, with a discount factor of 0.97.<sup>4</sup> However, our algorithm only uses these labels for gradient routing, and not for RL updates.

**Architecture and training.** The first layer of the policy network is a modified Mixture of Experts (MoE) layer. The MoE layer computes a weighted sum of two experts,  $E_{DIAMOND}(\cdot)$  and  $E_{GHOST}(\cdot)$ . The module weights, which are between 0 and 1, are computed by a multi-layer gating network. The remaining layers in the network are either convolutional or linear. (See appendix F for details.)

The policy network is trained with Proximal Policy Optimization (PPO) (Schulman et al., 2017). We perform gradient routing only when training on episodes with oversight. In those cases, we route the gradients through one of the modules,  $E_{DIAMOND}(\cdot)$  or  $E_{GHOST}(\cdot)$ , depending on which terminal square the agent reached. We also add a loss term that incentivizes the gating network to activate only the module through which the gradients were routed. In episodes without oversight, we perform a normal, non-routed PPO update with full backpropagation.

**Results.** Our method uses gradient routing based on limited labels to localize a module responsible for going to GHOST, then ablates the module. Below, we show that the method works even in the most challenging setting: instances of the environment where GHOST is the closest terminal square, where we have no oversight (and the agent knows this). Details are given in appendix F.

Our *gradient-routed MoE* agent reaches DIAMOND 70% of the time (fig. 5a) with as low as 4% oversight during training. An *outcome-based* baseline, trained on all episodes and given reward -1 when it reached GHOST under oversight, and +1 when it reached a terminal square in any other case, learns to exploit the limits of the labels, going to GHOST whenever it is closer than DIAMOND and not under oversight. A *data filtering* baseline was trained on a random subsample of all training episodes, but granted full oversight. Gradient-routed MoE under 4% oversight has comparable performance to the data filtering baseline at a 20% sampling rate, representing 5 times greater data

<sup>4</sup>For general reinforcement learning terminology, see Sutton & Barto (2018).

486 efficiency (fig. 11 in appendix F). So, we achieve scalable oversight: gradient routing avoids the  
 487 perils of naive outcome-based training while enjoying far greater data efficiency than data filtering.  
 488

## 489 5 DISCUSSION

491 **Absorption.** Routing a subset of the data related to some knowledge or capability appears to localize  
 492 that knowledge or capability more generally. This held for an i.i.d. subset of the data (TinyStories  
 493 unlearning in section 4.2.2), and for semantically limited data (steering scalar in section 4.2.1, vi-  
 494 rology unlearning in section 4.2.3, scalable oversight in section 4.3). We hypothesize an *absorption*  
 495 effect: routing limited data to a region creates internal features or units of computation in that re-  
 496 gion which are relevant to a broader task; these units then participate in the model’s predictions on  
 497 related, non-routed data; the resulting prediction errors are then backpropagated to the same region,  
 498 creating a positive feedback loop that reinforces those features. To the extent that absorption is true,  
 499 it has advantages over data filtering methods: if data filtering labels are limited either in quantity or  
 500 semantically, then harmful capabilities can still be learned where labels are missing, whereas routing  
 501 that data to a region absorbs those capabilities into that region, which can then be removed.

502 **Benefits of localization vs. suppression.** When the ability to label (or score) undesirable behavior  
 503 is imperfect, attempting to suppress the behavior may be perilous: a model may learn to exploit the  
 504 limits of the labels, rather than learning the desired behavior (Goodhart, 1984; Karwowski et al.,  
 505 2023). Our study of scalable oversight presents a model of this scenario, demonstrating the advan-  
 506 tage of localization as opposed to attenuation of undesirable behavior. This advantage may apply  
 507 more broadly, for example, to machine learning problems where capabilities are *entangled*, in the  
 508 sense that there are connections or dependencies between the computation learned to perform differ-  
 509 ent tasks (Arora & Goyal, 2023; de Chiusole & Stefanutti, 2013). Entanglement might occur because  
 510 certain capabilities or behaviors are reinforced by a broad range of training objectives (Omohundro,  
 511 2008; Turner et al., 2021; Krakovna et al., 2020). More simply, capabilities required to perform  
 512 undesired tasks may overlap with those required to perform desired tasks. For example, biological  
 513 knowledge entails much of the knowledge required to construct biological weapons. For this rea-  
 514 son, filtering or training against bioweapon-specific data might not prevent a network from learning  
 515 enough to create bioweapons from general biology sources.<sup>5</sup>

516 **Limitations and future work.** (a) Gradient routing’s performance is sensitive to its many hyper-  
 517 perparameters: what data to route on, what regions to localize to, and what mask weights to use.  
 518 This makes it hard to balance retain set performance vs. unlearning, for example. We suspect that  
 519 methodological improvements will reduce this sensitivity. (b) So far, we have studied gradient rout-  
 520 ing as a pretraining method, making it costly to experiment with large models. (c) In our experiments  
 521 with language models, we route gradients on a token-by-token basis, ignoring neighboring tokens.  
 522 This naive strategy is surprisingly effective. However, it is plausible that contextual information  
 523 will be critical in some problems, necessitating routing strategies that depend on entire sequences.  
 524 Finding practical ways of choosing what data to route in order to localize broad capabilities is an  
 525 intriguing open problem. (d) Our empirical results for scalable oversight pertain to a simplistic, nar-  
 526 row setting. Furthermore, our method for scalable oversight requires that the ablated policy produce  
 527 coherent behavior. This does not hold in general, so scaling oversight via localization may require  
 528 new ideas. (e) Other methods could be used to achieve similar aims as gradient routing, for example,  
 529 DEMix Layers (Gururangan et al., 2021) or Interchange Intervention Training (Geiger et al., 2022a).  
 530 (f) We elaborate on application-specific limitations in appendix A.

## 531 6 CONCLUSION

533 Gradient routing localizes targeted capabilities in neural networks, creating models with known in-  
 534 ternal structure. Even when based on simple and limited data labeling schemes, this localization  
 535 is suitable for robust unlearning of pre-specified capabilities and scalable oversight. Consequently,  
 536 gradient routing may facilitate the safe deployment of AI systems, particularly in high-stakes sce-  
 537 narios where black-box methods are insufficiently robust.

538 <sup>5</sup>Another reason suppression may be insufficient to provide safety guarantees is that poor behavioral per-  
 539 formance does not entail the elimination of internal circuitry related to that behavior (Lee et al., 2024; Casper  
 et al., 2024).

## 540 REPRODUCIBILITY STATEMENT

541 We include detailed descriptions of experiment settings in the appendix. Anonymized code to re-  
542 produce our results is presented as-is at:

543 <https://anonymous.4open.science/r/factored-representations-3035/README.md>.

## 546 ACKNOWLEDGMENTS

547 Anonymized for review.

## 550 REFERENCES

- 551 Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Con-  
552 crete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.
- 553 Sanjeev Arora and Anirudh Goyal. A theory for emergence of complex skills in language  
554 models. *ArXiv*, abs/2307.15936, 2023. URL [https://api.semanticscholar.org/  
555 CorpusID:260334352](https://api.semanticscholar.org/CorpusID:260334352).
- 556 Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, John Kernion, Andy Jones, Anna  
557 Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson,  
558 Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson,  
559 E Perez, Jamie Kerr, Jared Mueller, Jeff Ladish, J Landau, Kamal Ndousse, Kamil Lukosit, Liane  
560 Lovitt, Michael Sellitto, Nelson Elhage, Nicholas Schiefer, Noemí Mercado, Nova Dassarma,  
561 Robert Lasenby, Robin Larson, Sam Ringer, Scott Johnston, Shauna Kravec, Sheer El Showk,  
562 Stanislav Fort, Tamera Lanham, Timothy Telleen-Lawton, Tom Conerly, Tom Henighan, Tristan  
563 Hume, Sam Bowman, Zac Hatfield-Dodds, Benjamin Mann, Dario Amodei, Nicholas Joseph,  
564 Sam McCandlish, Tom B. Brown, and Jared Kaplan. Constitutional ai: Harmlessness from ai  
565 feedback. *ArXiv*, abs/2212.08073, 2022. URL [https://api.semanticscholar.org/  
566 CorpusID:254823489](https://api.semanticscholar.org/CorpusID:254823489).
- 567 Bowen Baker, Ilge Akkaya, Peter Zhokov, Joost Huizinga, Jie Tang, Adrien Ecoffet, Brandon  
568 Houghton, Raul Sampedro, and Jeff Clune. Video pretraining (vpt): Learning to act by watching  
569 unlabeled online videos. *Advances in Neural Information Processing Systems*, 35:24639–24654,  
570 2022.
- 571 Ankur Bapna and Orhan Firat. Simple, scalable adaptation for neural machine translation. In Ken-  
572 taro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (eds.), *Proceedings of the 2019 Conference*  
573 *on Empirical Methods in Natural Language Processing and the 9th International Joint Confer-*  
574 *ence on Natural Language Processing (EMNLP-IJCNLP)*, pp. 1538–1548, Hong Kong, China,  
575 November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1165. URL  
576 <https://aclanthology.org/D19-1165>.
- 577 Deniz Bayazit, Negar Foroutan, Zeming Chen, Gail Weiss, and Antoine Bosselut. Discovering  
578 knowledge-critical subnetworks in pretrained language models. *arXiv preprint arXiv:2310.03084*,  
579 2023.
- 580 Nora Belrose, David Schneider-Joseph, Shauli Ravfogel, Ryan Cotterell, Edward Raff, and Stella  
581 Biderman. LEACE: Perfect linear concept erasure in closed form. In *Thirty-seventh Confer-*  
582 *ence on Neural Information Processing Systems*, 2023. URL [https://openreview.net/  
583 forum?id=awIpKpwTWF](https://openreview.net/forum?id=awIpKpwTWF).
- 584 Elad Ben Zaken, Yoav Goldberg, and Shauli Ravfogel. BitFit: Simple parameter-efficient fine-  
585 tuning for transformer-based masked language-models. In Smaranda Muresan, Preslav Nakov,  
586 and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for*  
587 *Computational Linguistics (Volume 2: Short Papers)*, pp. 1–9, Dublin, Ireland, May 2022. As-  
588 sociation for Computational Linguistics. doi: 10.18653/v1/2022.acl-short.1. URL [https:  
589 //aclanthology.org/2022.acl-short.1](https://aclanthology.org/2022.acl-short.1).
- 590 Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new  
591 perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828,  
592 2013.

- 594 John Beverley, David Limbaugh, Eric Merrell, Peter M. Koch, and Barry Smith. Capabilities: An  
595 ontology. In *Proceedings of the Joint Ontology Workshops (JOWO) - Episode X: The Tukker*  
596 *Zomer of Ontology, and satellite events co-located with the 14th International Conference on*  
597 *Formal Ontology in Information Systems (FOIS 2024)*, Enschede, The Netherlands, July 15-19  
598 2024. JOWO. URL <https://arxiv.org/pdf/2405.00183>. <https://arxiv.org/pdf/2405.00183>.
- 600 Tolga Bolukbasi, Kai-Wei Chang, James Y. Zou, Venkatesh Saligrama, and Adam Tauman Kalai.  
601 Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In  
602 *Neural Information Processing Systems*, 2016. URL <https://api.semanticscholar.org/CorpusID:1704893>.
- 604 Lucas Bourtole, Varun Chandrasekaran, Christopher A. Choquette-Choo, Hengrui Jia, Adelin  
605 Travers, Baiwu Zhang, David Lie, and Nicolas Papernot. Machine unlearning. In *2021 IEEE*  
606 *Symposium on Security and Privacy (SP)*, pp. 141–159, 2021. doi: 10.1109/SP40001.2021.00019.
- 608 Collin Burns, Pavel Izmailov, Jan Hendrik Kirchner, Bowen Baker, Leo Gao, Leopold Aschenbren-  
609 ner, Yining Chen, Adrien Ecoffet, Manas Joglekar, Jan Leike, Ilya Sutskever, and Jeffrey Wu.  
610 Weak-to-strong generalization: Eliciting strong capabilities with weak supervision. In Ruslan  
611 Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and  
612 Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learn-*  
613 *ing*, volume 235 of *Proceedings of Machine Learning Research*, pp. 4971–5012. PMLR, 21–27  
614 Jul 2024. URL <https://proceedings.mlr.press/v235/burns24b.html>.
- 615 Yinzhi Cao and Junfeng Yang. Towards making systems forget with machine unlearning. In *2015*  
616 *IEEE Symposium on Security and Privacy*, pp. 463–480, 2015. doi: 10.1109/SP.2015.35.
- 617 Stephen Casper, Lennart Schulze, Oam Patel, and Dylan Hadfield-Menell. Defending against un-  
618 foreseen failure modes with latent adversarial training. *arXiv preprint arXiv:2403.05030*, 2024.
- 619 Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and P. Abbeel. Infogan:  
620 Interpretable representation learning by information maximizing generative adversarial nets. In  
621 *Neural Information Processing Systems*, 2016. URL <https://api.semanticscholar.org/CorpusID:5002792>.
- 622 Arthur Conmy, Augustine Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-  
623 Alonso. Towards automated circuit discovery for mechanistic interpretability. In A. Oh,  
624 T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neu-*  
625 *ral Information Processing Systems*, volume 36, pp. 16318–16352. Curran Associates, Inc.,  
626 2023. URL [https://proceedings.neurips.cc/paper\\_files/paper/2023/](https://proceedings.neurips.cc/paper_files/paper/2023/file/34e1dbe95d34d7ebaf99b9bcaeb5b2be-Paper-Conference.pdf)  
627 [file/34e1dbe95d34d7ebaf99b9bcaeb5b2be-Paper-Conference.pdf](https://proceedings.neurips.cc/paper_files/paper/2023/file/34e1dbe95d34d7ebaf99b9bcaeb5b2be-Paper-Conference.pdf).
- 628 D. de Chiusole and L. Stefanutti. Modeling skill dependence in probabilistic competence structures.  
629 *Electronic Notes in Discrete Mathematics*, 42:41–48, 2013. ISSN 1571-0653. doi: <https://doi.org/10.1016/j.endm.2013.05.144>. URL [https://www.sciencedirect.com/science/](https://www.sciencedirect.com/science/article/pii/S1571065313001479)  
630 [article/pii/S1571065313001479](https://www.sciencedirect.com/science/article/pii/S1571065313001479).
- 631 Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hier-  
632 archical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*,  
633 pp. 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.
- 634 Coline Devin, Abhishek Gupta, Trevor Darrell, Pieter Abbeel, and Sergey Levine. Learning modular  
635 neural network policies for multi-task and multi-robot transfer. In *2017 IEEE International Con-*  
636 *ference on Robotics and Automation (ICRA)*, pp. 2169–2176, 2017. doi: 10.1109/ICRA.2017.  
637 7989250.
- 638 Harrison Edwards and Amos J. Storkey. Censoring representations with an adversary. *CoRR*,  
639 abs/1511.05897, 2015. URL [https://api.semanticscholar.org/CorpusID:](https://api.semanticscholar.org/CorpusID:4986726)  
640 [4986726](https://api.semanticscholar.org/CorpusID:4986726).
- 641 David Eigen, Marc’Aurelio Ranzato, and Ilya Sutskever. Learning factored representations in a deep  
642 mixture of experts. *CoRR*, abs/1312.4314, 2013. URL [https://api.semanticscholar.org/CorpusID:](https://api.semanticscholar.org/CorpusID:11492613)  
643 [11492613](https://api.semanticscholar.org/CorpusID:11492613).

- 648 Yanai Elazar, Shauli Ravfogel, Alon Jacovi, and Yoav Goldberg. Amnesic probing: Behavioral ex-  
649 planation with amnesic counterfactuals. *Transactions of the Association for Computational Lin-*  
650 *guistics*, 9:160–175, 2020. URL [https://api.semanticscholar.org/CorpusID:](https://api.semanticscholar.org/CorpusID:227408471)  
651 [227408471](https://api.semanticscholar.org/CorpusID:227408471).
- 652 Ronen Eldan and Yuanzhi Li. Tinstories: How small can language models be and still speak  
653 coherent english? *arXiv preprint arXiv:2305.07759*, 2023.
- 654 Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann,  
655 Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep  
656 Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt,  
657 Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and  
658 Chris Olah. A mathematical framework for transformer circuits. *Transformer Circuits Thread*,  
659 2021. <https://transformer-circuits.pub/2021/framework/index.html>.
- 660 Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec,  
661 Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish,  
662 Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. Toy models of super-  
663 position. *Transformer Circuits Thread*, 2022. URL [https://transformer-circuits.](https://transformer-circuits.pub/2022/toy_model/index.html)  
664 [pub/2022/toy\\_model/index.html](https://transformer-circuits.pub/2022/toy_model/index.html).
- 665 Angela Fan, Shruti Bhosale, Holger Schwenk, Zhiyi Ma, Ahmed El-Kishky, Siddharth Goyal, Man-  
666 deep Baines, Onur Celebi, Guillaume Wenzek, Vishrav Chaudhary, Naman Goyal, Tom Birch,  
667 Vitaliy Liptchinsky, Sergey Edunov, Michael Auli, and Armand Joulin. Beyond english-centric  
668 multilingual machine translation. *Journal of Machine Learning Research*, 22(107):1–48, 2021.  
669 URL <http://jmlr.org/papers/v22/20-1307.html>.
- 670 Chelsea Finn, Tianhe Yu, Justin Fu, P. Abbeel, and Sergey Levine. Generalizing skills with  
671 semi-supervised reinforcement learning. *ArXiv*, abs/1612.00429, 2016. URL [https://api.](https://api.semanticscholar.org/CorpusID:8685592)  
672 [semanticscholar.org/CorpusID:8685592](https://api.semanticscholar.org/CorpusID:8685592).
- 673 Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by backpropagation. In  
674 Francis Bach and David Blei (eds.), *Proceedings of the 32nd International Conference on Ma-*  
675 *chine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1180–1189,  
676 Lille, France, 07–09 Jul 2015. PMLR. URL [https://proceedings.mlr.press/v37/](https://proceedings.mlr.press/v37/ganin15.html)  
677 [ganin15.html](https://proceedings.mlr.press/v37/ganin15.html).
- 678 Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François  
679 Laviolette, Mario March, and Victor Lempitsky. Domain-adversarial training of neural net-  
680 works. *Journal of Machine Learning Research*, 17(59):1–35, 2016. URL [http://jmlr.](http://jmlr.org/papers/v17/15-239.html)  
681 [org/papers/v17/15-239.html](http://jmlr.org/papers/v17/15-239.html).
- 682 Atticus Geiger, Zhengxuan Wu, Hanson Lu, Josh Rozner, Elisa Kreiss, Thomas Icard, Noah Good-  
683 man, and Christopher Potts. Inducing causal fstructure for interpretable neural networks. In  
684 *International Conference on Machine Learning*, 2022a.
- 685 Atticus Geiger, Zhengxuan Wu, Hanson Lu, Josh Rozner, Elisa Kreiss, Thomas Icard, Noah Good-  
686 man, and Christopher Potts. Inducing causal structure for interpretable neural networks. In Ka-  
687 malika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato  
688 (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of  
689 *Proceedings of Machine Learning Research*, pp. 7324–7338. PMLR, 17–23 Jul 2022b. URL  
690 <https://proceedings.mlr.press/v162/geiger22a.html>.
- 691 Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair,  
692 Aaron C. Courville, and Yoshua Bengio. Generative adversarial nets. In *Neural Information*  
693 *Processing Systems*, 2014. URL [https://api.semanticscholar.org/CorpusID:](https://api.semanticscholar.org/CorpusID:261560300)  
694 [261560300](https://api.semanticscholar.org/CorpusID:261560300).
- 695 C. A. E. Goodhart. *Problems of Monetary Management: The UK Experience*. Macmillan Educa-  
696 tion UK, London, 1984. ISBN 978-1-349-17295-5. doi: 10.1007/978-1-349-17295-5.4. URL  
697 [https://doi.org/10.1007/978-1-349-17295-5\\_4](https://doi.org/10.1007/978-1-349-17295-5_4).

- 702 Phillip Huang Guo, Aaquib Syed, Abhay Sheshadri, Aidan Ewart, and Gintare Karolina Dziugaite.  
703 Robust unlearning via mechanistic localizations. In *ICML 2024 Workshop on Mechanistic Inter-*  
704 *pretability*, 2024. URL <https://openreview.net/forum?id=06pNzrEjnH>.  
705
- 706 Suchin Gururangan, Michael Lewis, Ari Holtzman, Noah A. Smith, and Luke Zettlemoyer. Demix  
707 layers: Disentangling domains for modular language modeling. In *North American Chapter of the*  
708 *Association for Computational Linguistics*, 2021. URL <https://api.semanticscholar.org/CorpusID:236976189>.  
709
- 710 Nicolas Heess, Greg Wayne, Yuval Tassa, Timothy Lillicrap, Martin Riedmiller, and David  
711 Silver. Learning and Transfer of Modulated Locomotor Controllers. *arXiv e-prints*, art.  
712 arXiv:1610.05182, October 2016. doi: 10.48550/arXiv.1610.05182.  
713
- 714 Peter Henderson, Eric Mitchell, Christopher Manning, Dan Jurafsky, and Chelsea Finn. Self-  
715 destructing models: Increasing the costs of harmful dual uses of foundation models. In *Pro-*  
716 *ceedings of the 2023 AAAI/ACM Conference on AI, Ethics, and Society*, AIES '23, pp. 287296,  
717 New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400702310. doi:  
718 10.1145/3600211.3604690. URL <https://doi.org/10.1145/3600211.3604690>.  
719
- 720 John Hewitt, John Thickstun, Christopher D. Manning, and Percy Liang. Backpack language mod-  
721 els. In *Proceedings of the Association for Computational Linguistics*. Association for Computa-  
722 tional Linguistics, 2023.
- 723 Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, An-  
724 drea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for  
725 NLP. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th In-*  
726 *ternational Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning*  
727 *Research*, pp. 2790–2799. PMLR, 09–15 Jun 2019. URL <https://proceedings.mlr.press/v97/houlsby19a.html>.  
728
- 729 Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification.  
730 In Iryna Gurevych and Yusuke Miyao (eds.), *Proceedings of the 56th Annual Meeting of the*  
731 *Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 328–339, Melbourne,  
732 Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1031.  
733 URL <https://aclanthology.org/P18-1031>.
- 734 Chia-Yi Hsu, Yu-Lin Tsai, Chih-Hsun Lin, Pin-Yu Chen, Chia-Mu Yu, and Chun ying Huang.  
735 Safe lora: the silver lining of reducing safety risks when fine-tuning large language mod-  
736 els. *ArXiv*, abs/2405.16833, 2024. URL <https://api.semanticscholar.org/CorpusID:270063864>.  
737
- 738 Shengyi Huang, Rousslan Fernand Julien Dossa, Chang Ye, Jeff Braga, Dipam Chakraborty, Kinal  
739 Mehta, and Joo G.M. Arajo. Cleanrl: High-quality single-file implementations of deep reinforce-  
740 ment learning algorithms. *Journal of Machine Learning Research*, 23(274):1–18, 2022. URL  
741 <http://jmlr.org/papers/v23/21-1342.html>.  
742
- 743 Tiansheng Huang, Gautam Bhattacharya, Pratik Joshi, Josh Kimball, and Ling Liu. Antidote: Post-  
744 fine-tuning safety alignment for large language models against harmful fine-tuning. *arXiv preprint*  
745 *arXiv:2408.09600*, 2024.
- 746 Matthias Hutsebaut-Buysse, Kevin Mets, and Steven Latr. Hierarchical reinforcement learning: A  
747 survey and open research challenges. *Machine Learning and Knowledge Extraction*, 4(1):172–  
748 221, 2022. ISSN 2504-4990. doi: 10.3390/make4010009. URL <https://www.mdpi.com/2504-4990/4/1/9>.  
749
- 750 Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Ludwig Schmidt, Hannaneh Hajishirzi,  
751 and Ali Farhadi. Editing models with task arithmetic. In *The Eleventh International Confer-*  
752 *ence on Learning Representations*, 2023. URL <https://openreview.net/forum?id=6t0Kwf8-jrj>.  
753
- 754 Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures  
755 of local experts. *Neural Computation*, 3(1):79–87, 1991. doi: 10.1162/neco.1991.3.1.79.

- 756 Jett Janiak, Jai Dhyani, Jannik Brinkmann, Gonalo Paulo, Joshua Wendland, Vctor Abia Alonso,  
757 Siwei Li, Phan Anh Duong, and Alice Rigg. delphi: small language models training made easy,  
758 2024. URL <https://github.com/delphi-suite/delphi>.
- 759 Xisen Jin, Xiang Ren, Daniel Preotiuc-Pietro, and Pengxiang Cheng. Dataless knowledge fusion  
760 by merging weights of language models. In *The Eleventh International Conference on Learning*  
761 *Representations*, 2023. URL <https://openreview.net/forum?id=FCnohuR6AnM>.
- 762 Gal Kaplun, Andrey Gurevich, Tal Swisa, Mazor David, Shai Shalev-Shwartz, and eran malach.  
763 Less is more: Selective layer finetuning with subtuning, 2024. URL <https://openreview.net/forum?id=sOHVDPqoUJ>.
- 764 Andrej Karpathy. karpathy/nanoGPT, September 2024. URL <https://github.com/karpathy/nanoGPT>. original-date: 2022-12-28T00:51:12Z.
- 765 Jacek Karwowski, Oliver Hayman, Xingjian Bai, Klaus Kiendlhofer, Charlie Griffin, and Joar  
766 Skalse. Goodhart’s law in reinforcement learning. *arXiv preprint arXiv:2310.09144*, 2023.
- 767 Zachary Kenton, Noah Y. Siegel, Jnos Kramr, Jonah Brown-Cohen, Samuel Albanie, Jannis Bulian,  
768 Rishabh Agarwal, David Lindner, Yunhao Tang, Noah D. Goodman, and Rohin Shah. On scal-  
769 able oversight with weak llms judging strong llms, 2024. URL <https://arxiv.org/abs/2407.04622>.
- 770 Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*,  
771 2014.
- 772 Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *CoRR*, abs/1312.6114,  
773 2013. URL <https://api.semanticscholar.org/CorpusID:216078090>.
- 774 Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and  
775 Percy Liang. Concept bottleneck models. In Hal Daum III and Aarti Singh (eds.), *Proceedings of*  
776 *the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine*  
777 *Learning Research*, pp. 5338–5348. PMLR, 13–18 Jul 2020. URL <https://proceedings.mlr.press/v119/koh20a.html>.
- 778 Victoria Krakovna, Jonathan Uesato, Vladimir Mikulik, Matthew Raatz, Tom Everitt, Ramana  
779 Kumar, Zac Kenton, Jan Leike, and Shane Legg. Specification gaming: the flip side  
780 of ai ingenuity. DeepMind Blog, 2020. URL <https://www.deepmind.com/blog/specification-gaming-the-flip-side-of-ai-ingenuity>. Published 21 April  
781 2020.
- 782 Lauro Langosco Di Langosco, Jack Koch, Lee D Sharkey, Jacob Pfau, and David Krueger. Goal  
783 misgeneralization in deep reinforcement learning. In Kamalika Chaudhuri, Stefanie Jegelka,  
784 Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th In-*  
785 *ternational Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning*  
786 *Research*, pp. 12004–12019. PMLR, 17–23 Jul 2022. URL <https://proceedings.mlr.press/v162/langosco22a.html>.
- 787 Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to  
788 document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- 789 Andrew Lee, Xiaoyan Bai, Itamar Pres, Martin Wattenberg, Jonathan K Kummerfeld, and Rada Mi-  
790 halcea. A mechanistic understanding of alignment algorithms: A case study on dpo and toxicity.  
791 *arXiv preprint arXiv:2401.01967*, 2024.
- 792 Yoonho Lee, Annie S Chen, Fahim Tajwar, Ananya Kumar, Huaxiu Yao, Percy Liang, and Chelsea  
793 Finn. Surgical fine-tuning improves adaptation to distribution shifts. In *The Eleventh International*  
794 *Conference on Learning Representations*, 2023. URL <https://openreview.net/forum?id=APuPRxjHvZ>.
- 795 Simon Lermen, Charlie Rogers-Smith, and Jeffrey Ladish. Lora fine-tuning efficiently undoes  
796 safety training in llama 2-chat 70b. *ArXiv*, abs/2310.20624, 2023. URL <https://api.semanticscholar.org/CorpusID:264808400>.

- 810 Nathaniel Li, Alexander Pan, Anjali Gopal, Summer Yue, Daniel Berrios, Alice Gatti, Justin D Li,  
811 Ann-Kathrin Dombrowski, Shashwat Goel, Long Phan, et al. The wmdp benchmark: Measuring  
812 and reducing malicious use with unlearning. *arXiv preprint arXiv:2403.03218*, 2024.  
813
- 814 Sijia Liu, Yuanshun Yao, Jinghan Jia, Stephen Casper, Nathalie Baracaldo, Peter Hase, Xiaojun Xu,  
815 Yuguang Yao, Chris Liu, Hang Li, Kush R. Varshney, Mohit Bansal, Sanmi Koyejo, and Yang  
816 Liu. Rethinking machine unlearning for large language models. *ArXiv*, abs/2402.08787, 2024.  
817 URL <https://api.semanticscholar.org/CorpusID:267657624>.
- 818 Tyler Lizzo and Larry Heck. Unlearn efficient removal of knowledge in large language models,  
819 2024. URL <https://arxiv.org/abs/2408.04140>.
- 820
- 821 Michelle Lo, Shay B. Cohen, and Fazl Barez. Large language models relearn removed concepts,  
822 2024.
- 823
- 824 Ilya Loshchilov and Frank Hutter. Decoupled Weight Decay Regularization. September 2018. URL  
825 <https://openreview.net/forum?id=Bkg6RiCqY7>.
- 826 Jakub Lucki, Boyi Wei, Yangsibo Huang, Peter Henderson, Florian Tramr, and Javier Rando. An  
827 adversarial perspective on machine unlearning for ai safety, 2024. URL <https://arxiv.org/abs/2409.18025>.
- 828
- 829 Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Harsh Lara, Yunxuan Li, Lei Shu, Yun  
830 Zhu, Lei Meng, Jiao Sun, et al. Improve mathematical reasoning in language models by automated  
831 process supervision. *arXiv preprint arXiv:2406.06592*, 2024.  
832
- 833 Aengus Lynch, Phillip Guo, Aidan Ewart, Stephen Casper, and Dylan Hadfield-Menell. Eight meth-  
834 ods to evaluate robust unlearning in llms, 2024. URL <https://arxiv.org/abs/2402.16835>.
- 835
- 836 Pattie Maes and Rodney A Brooks. Learning to coordinate behaviors. In *AAAI*, volume 90, pp.  
837 796–802. Boston, MA, 1990.  
838
- 839 Sridhar Mahadevan and Jonathan Connell. Automatic programming of behavior-based robots using  
840 reinforcement learning. *Artificial Intelligence*, 55(2):311–365, 1992. ISSN 0004-3702. doi: [https://doi.org/10.1016/0004-3702\(92\)90058-6](https://doi.org/10.1016/0004-3702(92)90058-6). URL <https://www.sciencedirect.com/science/article/pii/0004370292900586>.
- 841
- 842
- 843 Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative  
844 pruning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*,  
845 pp. 7765–7773, 2018.  
846
- 847 Emile Mathieu, Tom Rainforth, N Siddharth, and Yee Whye Teh. Disentangling disentanglement in  
848 variational autoencoders. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings*  
849 *of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine*  
850 *Learning Research*, pp. 4402–4412. PMLR, 09–15 Jun 2019. URL <https://proceedings.mlr.press/v97/mathieu19a.html>.
- 851
- 852 Tom McGrath, Matthew Rahtz, János Kramár, Vladimir Mikulik, and Shane Legg. The hydra  
853 effect: Emergent self-repair in language model computations. *ArXiv*, abs/2307.15771, 2023.  
854 URL <https://api.semanticscholar.org/CorpusID:260334719>.
- 855
- 856 Joseph Miller, Bilal Chughtai, and William Saunders. Transformer circuit evaluation metrics are  
857 not robust. In *First Conference on Language Modeling*, 2024. URL <https://openreview.net/forum?id=zSf8PJyQb2>.
- 858
- 859 Amirkeivan Mohtashami, Martin Jaggi, and Sebastian U Stich. Masked training of neural networks  
860 with partial gradients. In *Proceedings of the 25th International Conference on Artificial Intelli-*  
861 *gence and Statistics*, 2022.  
862
- 863 Chris Olah, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, and Shan Carter.  
Zoom in: An introduction to circuits. *Distill*, 5(3):e00024–001, 2020.



- 864 Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan,  
865 Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, et al. In-context learning and induction  
866 heads. *arXiv preprint arXiv:2209.11895*, 2022.  
867
- 868 Stephen M. Omohundro. The basic ai drives. In *Proceedings of the 2008 Conference on Artificial  
869 General Intelligence 2008: Proceedings of the First AGI Conference*, pp. 483492, NLD, 2008.  
870 IOS Press. ISBN 9781586038335.
- 871 Ashwinee Panda, Berivan Isik, Xiangyu Qi, Sanmi Koyejo, Tsachy Weissman, and Prateek Mit-  
872 tal. Lottery ticket adaptation: Mitigating destructive interference in LLMs. In *2nd Workshop on  
873 Advancing Neural Network Training: Computational Efficiency, Scalability, and Resource Op-  
874 timization (WANT@ICML 2024)*, 2024a. URL <https://openreview.net/forum?id=qD2eFNvtw4>.  
875
- 876 Ashwinee Panda, Berivan Isik, Xiangyu Qi, Sanmi Koyejo, Tsachy Weissman, and Prateek Mittal.  
877 Lottery ticket adaptation: Mitigating destructive interference in LLMs, 2024b. URL <http://arxiv.org/abs/2406.16797>.  
878
- 879 Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor  
880 Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-  
881 performance deep learning library. *Advances in neural information processing systems*, 32, 2019.  
882
- 883 Vaidehi Patil, Peter Hase, and Mohit Bansal. Can sensitive information be deleted from llms? ob-  
884 jectives for defending against extraction attacks. *ArXiv*, abs/2309.17410, 2023. URL <https://api.semanticscholar.org/CorpusID:263311025>.  
885
- 886 Guilherme Penedo, Hynek Kydlek, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin  
887 Raffel, Leandro Von Werra, and Thomas Wolf. The FineWeb datasets: Decanting the web for  
888 the finest text data at scale. (arXiv:2406.17557), 2024. doi: 10.48550/arXiv.2406.17557. URL  
889 <http://arxiv.org/abs/2406.17557>.  
890
- 891 Jonas Pfeiffer, Ivan Vulic, Iryna Gurevych, and Sebastian Ruder. Mad-x: An adapter-based frame-  
892 work for multi-task cross-lingual transfer. In *Conference on Empirical Methods in Natural  
893 Language Processing*, 2020. URL <https://api.semanticscholar.org/CorpusID:218470133>.  
894
- 895 Jonas Pfeiffer, Gregor Geigle, Aishwarya Kamath, Jan-Martin O. Steitz, Stefan Roth, Ivan Vulic,  
896 and Iryna Gurevych. xgqa: Cross-lingual visual question answering. In *Findings*, 2021. URL  
897 <https://api.semanticscholar.org/CorpusID:237490295>.  
898
- 899 Jonas Pfeiffer, Naman Goyal, Xi Lin, Xian Li, James Cross, Sebastian Riedel, and Mikel Artetxe.  
900 Lifting the curse of multilinguality by pre-training modular transformers. In Marine Carpuat,  
901 Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz (eds.), *Proceedings of the 2022  
902 Conference of the North American Chapter of the Association for Computational Linguistics:  
903 Human Language Technologies*, pp. 3479–3495, Seattle, United States, July 2022. Associa-  
904 tion for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.255. URL <https://aclanthology.org/2022.naacl-main.255>.  
905
- 906 Jonas Pfeiffer, Sebastian Ruder, Ivan Vulić, and Edoardo Ponti. Modular deep learning. *Transactions  
907 on Machine Learning Research*, 2023. ISSN 2835-8856. URL [https://openreview.net/  
908 forum?id=z9EkXfvxta](https://openreview.net/forum?id=z9EkXfvxta). Survey Certification.
- 909 Nicholas Pochinkov and Nandi Schoots. Dissecting language models: Machine unlearning via  
910 selective pruning, 2024. URL <https://arxiv.org/abs/2403.01267>.  
911
- 912 Martin L Puterman. Markov decision processes. *Handbooks in operations research and management  
913 science*, 2:331–434, 1990.
- 914 Ansh Radhakrishnan, Buck Shlegeris, Ryan Greenblatt, and Fabien Roger. Scal-  
915 able oversight and weak-to-strong generalization: Compatible approaches to the same  
916 problem. [https://www.alignmentforum.org/posts/hw2tGSsvLLyjFoLFS/  
917 scalable-oversight-and-weak-to-strong-generalization](https://www.alignmentforum.org/posts/hw2tGSsvLLyjFoLFS/scalable-oversight-and-weak-to-strong-generalization), December 2023.  
Accessed: 2024-09-21.

- 918 Shauli Ravfogel, Yanai Elazar, Hila Gonen, Michael Twiton, and Yoav Goldberg. Null it out:  
919 Guarding protected attributes by iterative nullspace projection. In *Annual Meeting of the As-*  
920 *sociation for Computational Linguistics*, 2020. URL <https://api.semanticscholar.org/CorpusID:215786522>.
- 922 Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Learning multiple visual domains  
923 with residual adapters. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vish-  
924 wanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30.  
925 Curran Associates, Inc., 2017. URL [https://proceedings.neurips.cc/paper\\_](https://proceedings.neurips.cc/paper_files/paper/2017/file/e7b24b112a44fdd9ee93bdf998c6ca0e-Paper.pdf)  
926 [files/paper/2017/file/e7b24b112a44fdd9ee93bdf998c6ca0e-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2017/file/e7b24b112a44fdd9ee93bdf998c6ca0e-Paper.pdf).
- 927 Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Efficient parametrization of multi-  
928 domain deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and*  
929 *Pattern Recognition (CVPR)*, June 2018.
- 931 Nina Rimskey, Nick Gabrieli, Julian Schulz, Meg Tong, Evan Hubinger, and Alexander Turner.  
932 Steering llama 2 via contrastive activation addition. In Lun-Wei Ku, Andre Martins, and Vivek  
933 Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational*  
934 *Linguistics (Volume 1: Long Papers)*, pp. 15504–15522, Bangkok, Thailand, August 2024.  
935 Association for Computational Linguistics. URL [https://aclanthology.org/2024.](https://aclanthology.org/2024.acl-long.828)  
936 [acl-long.828](https://aclanthology.org/2024.acl-long.828).
- 937 Amir Rosenfeld and John K. Tsotsos. Intriguing properties of randomly weighted networks: Gen-  
938 eralizing while learning next to nothing. *2019 16th Conference on Computer and Robot Vi-*  
939 *sion (CRV)*, pp. 9–16, 2018. URL [https://api.semanticscholar.org/CorpusID:](https://api.semanticscholar.org/CorpusID:3657091)  
940 [3657091](https://api.semanticscholar.org/CorpusID:3657091).
- 941 Amir Rosenfeld and John K. Tsotsos. Intriguing Properties of Randomly Weighted Networks:  
942 Generalizing While Learning Next to Nothing. In *2019 16th Conference on Computer and*  
943 *Robot Vision (CRV)*, pp. 9–16, May 2019. doi: 10.1109/CRV.2019.00010. URL [https:](https://ieeexplore.ieee.org/document/8781620)  
944 [//ieeexplore.ieee.org/document/8781620](https://ieeexplore.ieee.org/document/8781620).
- 945 Jürgen Schmidhuber. Learning factorial codes by predictability minimization. *Neural Computation*,  
946 4:863–879, 1992. URL <https://api.semanticscholar.org/CorpusID:2142508>.
- 947 John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy  
948 optimization algorithms. 2017. URL <https://arxiv.org/abs/1707.06347>.
- 949 Noam Shazeer, \*Azalia Mirhoseini, \*Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton,  
950 and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-  
951 experts layer. In *International Conference on Learning Representations*, 2017. URL [https:](https://openreview.net/forum?id=BlckMDqlg)  
952 [//openreview.net/forum?id=BlckMDqlg](https://openreview.net/forum?id=BlckMDqlg).
- 953 Abhay Sheshadri, Aidan Ewart, Phillip Guo, Aengus Lynch, Cindy Wu, Vivek Hebbar, Henry  
954 Sleight, Asa Cooper Stickland, Ethan Perez, Dylan Hadfield-Menell, et al. Targeted latent ad-  
955 versarial training improves robustness to persistent harmful behaviors in llms. *arXiv preprint*  
956 *arXiv:2407.15549*, 2024.
- 957 Weijia Shi, Anirudh Ajith, Mengzhou Xia, Yangsibo Huang, Daogao Liu, Terra Blevins, Danqi  
958 Chen, and Luke Zettlemoyer. Detecting pretraining data from large language models. In  
959 *The Twelfth International Conference on Learning Representations*, 2024. URL [https://](https://openreview.net/forum?id=zWqr3MQUNs)  
960 [openreview.net/forum?id=zWqr3MQUNs](https://openreview.net/forum?id=zWqr3MQUNs).
- 961 Satinder Pal Singh. Transfer of learning by composing solutions of elemental sequential tasks.  
962 *Machine learning*, 8:323–339, 1992.
- 963 Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford,  
964 Dario Amodei, and Paul F Christiano. Learning to summarize with human feedback. *Advances*  
965 *in Neural Information Processing Systems*, 33:3008–3021, 2020.
- 966 Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. RoFormer: En-  
967 hanced Transformer with Rotary Position Embedding. November 2023. doi: 10.48550/arXiv.  
968 2104.09864. URL <http://arxiv.org/abs/2104.09864>. arXiv:2104.09864 [cs].  
969  
970  
971

- 972 Xu Sun, Xuancheng Ren, Shuming Ma, and Houfeng Wang. meprop: Sparsified back propagation  
973 for accelerated deep learning with reduced overfitting. In *International Conference on Machine*  
974 *Learning*, pp. 3299–3308. PMLR, 2017a.
- 975  
976 Xu Sun, Xuancheng Ren, Shuming Ma, and Houfeng Wang. meProp: Sparsified back propagation  
977 for accelerated deep learning with reduced overfitting. In *Proceedings of the 34 th International*  
978 *Conference on Machine Learning*, 2017b.
- 979  
980 Yi-Lin Sung, Varun Nair, and Colin Raffel. Training neural networks with fixed sparse  
981 masks. *ArXiv*, abs/2111.09839, 2021. URL [https://api.semanticscholar.org/](https://api.semanticscholar.org/CorpusID:244345839)  
982 [CorpusID:244345839](https://api.semanticscholar.org/CorpusID:244345839).
- 983  
984 Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press,  
985 second edition, 2018. URL [http://incompleteideas.net/book/the-book-2nd.](http://incompleteideas.net/book/the-book-2nd.html)  
986 [html](http://incompleteideas.net/book/the-book-2nd.html).
- 987  
988 Rishub Tamirisa, Bhrugu Bharathi, Long Phan, Andy Zhou, Alice Gatti, Tarun Suresh, Maxwell  
989 Lin, Justin Wang, Rowan Wang, Ron Arel, Andy Zou, Dawn Song, Bo Li, Dan Hendrycks,  
990 and Mantas Mazeika. Tamper-resistant safeguards for open-weight llms, 2024. URL <https://arxiv.org/abs/2408.00761>.
- 991  
992 Alex Turner, Logan Smith, Rohin Shah, Andrew Critch, and Prasad Tadepalli. Optimal policies tend  
993 to seek power. *Advances in Neural Information Processing Systems*, 34:23063–23074, 2021.
- 994  
995 Alex Turner, Lisa Thiergart, David Udell, Gavin Leech, Ulisse Mini, and Monte MacDi-  
996 armid. Activation addition: Steering language models without optimization. *arXiv preprint*  
*arXiv:2308.10248*, 2023.
- 997  
998 Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, L. Wang, An-  
999 tonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with  
1000 process- and outcome-based feedback. *ArXiv*, abs/2211.14275, 2022. URL [https://api.](https://api.semanticscholar.org/CorpusID:254017497)  
[semanticscholar.org/CorpusID:254017497](https://api.semanticscholar.org/CorpusID:254017497).
- 1001  
1002 Fabio Urbina, Filippa Lentzos, Cdric Invernizzi, and Sean Ekins. Dual use of artificial-intelligence-  
1003 powered drug discovery. *Nature Machine Intelligence*, 4(3):189–191, March 2022. ISSN 2522-  
1004 5839. doi: 10.1038/s42256-022-00465-9. URL [https://www.nature.com/articles/](https://www.nature.com/articles/s42256-022-00465-9)  
1005 [s42256-022-00465-9](https://www.nature.com/articles/s42256-022-00465-9). Publisher: Nature Publishing Group.
- 1006  
1007 Jesper E. van Engelen and Holger H. Hoos. A survey on semi-supervised learning. *Machine Learn-*  
1008 *ing*, 109:373 – 440, 2019. URL [https://api.semanticscholar.org/CorpusID:](https://api.semanticscholar.org/CorpusID:254738406)  
[254738406](https://api.semanticscholar.org/CorpusID:254738406).
- 1009  
1010 A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.
- 1011  
1012 A. Waibel and J. Hampshire II. The meta-pi network: Building distributed knowledge represen-  
1013 tations for robust multisource pattern recognition. *IEEE Transactions on Pattern Analysis &*  
*Machine Intelligence*, 14(07):751–769, jul 1992. ISSN 1939-3539. doi: 10.1109/34.142911.
- 1014  
1015 Kevin Ro Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt.  
1016 Interpretability in the wild: a circuit for indirect object identification in GPT-2 small. In  
1017 *The Eleventh International Conference on Learning Representations*, 2023. URL [https://](https://openreview.net/forum?id=NpsVSN6o4ul)  
1018 [openreview.net/forum?id=NpsVSN6o4ul](https://openreview.net/forum?id=NpsVSN6o4ul).
- 1019  
1020 Xin Wang, Hong Chen, Si’ao Tang, Zihao Wu, and Wenwu Zhu. Disentangled representation learn-  
1021 ing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–20, 2024. doi:  
1022 10.1109/TPAMI.2024.3420937.
- 1023  
1024 Rui Xin, Chudi Zhong, Zhi Chen, Takuya Takagi, Margo I. Seltzer, and Cynthia Rudin. Ex-  
1025 ploring the whole rashomon set of sparse decision trees. *Advances in neural information pro-*  
*cessing systems*, 35:14071–14084, 2022. URL [https://api.semanticscholar.org/](https://api.semanticscholar.org/CorpusID:252355323)  
[CorpusID:252355323](https://api.semanticscholar.org/CorpusID:252355323).

1026 Xin Yi, Shunfan Zheng, Linlin Wang, Xiaoling Wang, and Liang He. A safety realignment frame-  
1027 work via subspace-oriented model fusion for large language models. *ArXiv*, abs/2405.09055,  
1028 2024. URL <https://api.semanticscholar.org/CorpusID:269773206>.  
1029

1030 Yang You, Igor Gitman, and Boris Ginsburg. Large batch training of convolutional networks. *arXiv:*  
1031 *Computer Vision and Pattern Recognition*, 2017. URL <https://api.semanticscholar.org/CorpusID:46294020>.  
1032

1033 Biao Zhang and Rico Sennrich. Root Mean Square Layer Normalization, October 2019. URL  
1034 <http://arxiv.org/abs/1910.07467>. arXiv:1910.07467 [cs, stat].  
1035

1036 Anyan Zhang, Michael A. Lepori, and Ellie Pavlick. Instilling inductive biases with subnetworks,  
1037 2024. URL <https://openreview.net/forum?id=B4nhr6OJWI>.

1038 Haojie Zhang, Ge Li, Jia Li, Zhongjin Zhang, Yuqi Zhu, and Zhi Jin. Fine-tuning pre-trained lan-  
1039 guage models effectively by optimizing subnetworks adaptively. *Advances in Neural Information*  
1040 *Processing Systems*, 35:21442–21454, 2022.

1041 Jinghan Zhang, shiqi chen, Junteng Liu, and Junxian He. Composing parameter-  
1042 efficient modules with arithmetic operation. In A. Oh, T. Naumann, A. Globerson,  
1043 K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Pro-*  
1044 *cessing Systems*, volume 36, pp. 12589–12610. Curran Associates, Inc., 2023. URL  
1045 [https://proceedings.neurips.cc/paper\\_files/paper/2023/file/](https://proceedings.neurips.cc/paper_files/paper/2023/file/299a08ee712d4752c890938da99a77c6-Paper-Conference.pdf)  
1046 [299a08ee712d4752c890938da99a77c6-Paper-Conference.pdf](https://proceedings.neurips.cc/paper_files/paper/2023/file/299a08ee712d4752c890938da99a77c6-Paper-Conference.pdf).  
1047

1048 Xiaojin Zhu, Andrew B. Goldberg, Ronald Brachman, and Thomas Dietterich. *Introduction to Semi-*  
1049 *Supervised Learning*. Morgan and Claypool Publishers, 2009. ISBN 1598295470.  
1050  
1051  
1052  
1053  
1054  
1055  
1056  
1057  
1058  
1059  
1060  
1061  
1062  
1063  
1064  
1065  
1066  
1067  
1068  
1069  
1070  
1071  
1072  
1073  
1074  
1075  
1076  
1077  
1078  
1079

## APPENDIX TO GRADIENT ROUTING: MASKING GRADIENTS TO LOCALIZE COMPUTATION IN NEURAL NETWORKS

### A EXTENDED DISCUSSION OF APPLICATION-SPECIFIC LIMITATIONS AND FUTURE WORK

**MNIST autoencoders.** The cleanly separated MNIST autoencoder representations depicted in fig. 2c depend on the problem setup (e.g. the choice to *not* use data augmentation, like rotations) and use of heavy L1 regularization on the encoding vector. L1 regularization is required because, by default, a regular MLP autoencoder trained on MNIST digits retains information necessary to decode other digits.

For a wide set of hyperparameters, we find that gradient routing achieves *quantitative* representation splitting: the Certificate’s reconstruction of digits 0-4 has higher average loss than its reconstructions of digits 5-9 for a wide range of settings, including different partitions of the digits. However, outside the specific hyperparameters chosen for the results in the main body of the paper, the *qualitative* results are poorer: the visual difference in reconstruction quality between the different digit subsets is less stark than in fig. 2c. We take this to highlight the problem-dependent characteristics of feature localization. In the case of autoencoding handwritten digits, separation of features for encoding different digits is “unnatural,” so achieving it requires a specific setup and heavy regularization.

**Language models.** We speculate that gradient routing on particular tokens introduces an “internal tug of war” between the expanded and original dimensions of the model (these dimensions depicted in fig. 3), where parameter updates in the original dimensions consistently decrease the logits for routed tokens and parameter updates in the expanded dimensions increase logits for routed tokens. This effect can be understood as a consequence of the mismatch between the implicit estimands (learning targets) for the original and expanded dimensions. We were concerned that this effect, rather than localization of capabilities, explained the post-ablation increase in forget loss. However, preliminary measurements suggest that this is not the case. For example, we find that the loss of ERA models is higher on average on *non-routed* forget tokens than a pure model, whereas it is lower on average on *routed* tokens. In general, the learning dynamics of gradient routing remain an open question.

If routing one token to a dimension of the residual stream creates an interpretable, axis-aligned feature as discussed in section 4.2.1, then routing many tokens to many neurons could produce a neural network with transparent internal representations. These representations might be made up of “individual neurons... [that] corresponded to cleanly interpretable features of the input,” as imagined in Elhage et al. (2022), or they could be organized in different ways. In principle, gradient routing provides a straightforward means of achieving this. However, we suspect that naive attempts to localize even moderate numbers of concepts to unique regions will lead to high training loss.

**Scalable oversight.** Our reinforcement learning results demonstrate the promise of a localization-based strategy for scalable oversight, but further empirical and conceptual work is needed. The toy environment we use is simple, lacking the complexity and asymmetries of real-world problems. Additionally, our proposed solution relies on the fact that ablating an otherwise-active module of a policy network produces a policy with coherent behavior, which may not be true in practice (and isn’t true in general, in principle). We discuss these considerations in appendix G.

### B MNIST AUTOENCODER DETAILS AND ABLATIONS

**Model architecture.** The Encoder, Decoder, and Certificate are all three-layer MLPs. The layer sizes for the Encoder produce data with shapes  $(28 \times 28, 2048, 512, 32)$  and for the decoder, data with shapes  $(32, 512, 2048, 28 \times 28)$ . All hidden layers use ReLU activations. The final layer of the Encoder is linear. The final layer of the decoders is affine.

**Training.** The model was trained for 200 epochs on the 60,000 image training part of the MNIST dataset (LeCun et al., 1998) with batch size 2048. Images were normalized to have mean and standard deviation 0.5. No data augmentation was used. Optimization was performed with Adam (Kingma, 2014) with learning rate  $1e-3$ ,  $\beta = (0.9, 0.999)$ , and weight decay  $5e-5$ .

The loss used was pixel-wise mean absolute error, with a penalty term for the L1 norm of the encoding and a penalty term for the sum of absolute correlations (across batch elements) between the top and bottom half of the encoding. For a batch of data indexed  $i = 1, \dots, n$  and encoding size 32, denote data points by  $x_i$ , encodings as  $\hat{z}_i$ , and Decoder outputs as  $\hat{x}_i$ . Then for  $\lambda = 0.003$  and  $\gamma = 0.1$ , the loss used to train the autoencoder is  $\mathcal{L} = \mathcal{L}_{\text{reconstruction}} + \lambda \cdot \mathcal{L}_{\text{L1}} + \gamma \cdot \mathcal{L}_{\text{Correlation}}$ , where

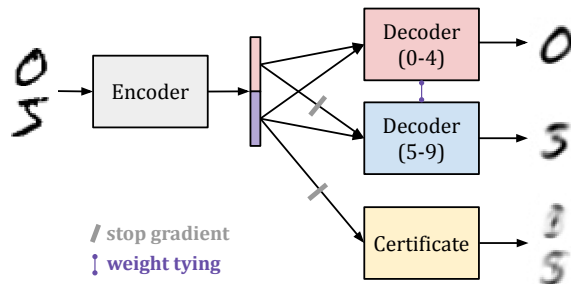
$$\begin{aligned} \mathcal{L}_{\text{reconstruction}} &= \frac{1}{28^2 \cdot n} \sum_{i=1}^n \|x_i - \hat{x}_i\|_1, \\ \mathcal{L}_{\text{L1}} &= \frac{1}{n} \sum_{i=1}^n \|\hat{z}_i\|_1, \text{ and} \\ \mathcal{L}_{\text{Correlation}} &= \frac{1}{16^2} \sum_{k=1}^{16} \sum_{h=17}^{32} \frac{\sum_{i=1}^n |\hat{z}_{i,k} - \bar{z}_{*,k}| |\hat{z}_{i,h} - \bar{z}_{*,h}|}{\sqrt{\sum_{i=1}^n (\hat{z}_{j,k} - \bar{z}_{*,k})^2} \sqrt{\sum_{i=1}^n (\hat{z}_{j,h} - \bar{z}_{*,h})^2}}, \end{aligned}$$

with  $\bar{z}_{*,k} = n^{-1} \sum_{i=1}^n \hat{z}_{i,k}$ . *Note: this equation does not include gradient routing, which is an intervention applied to gradients when backpropagating  $\mathcal{L}_{\text{reconstruction}}$  through  $\hat{z}_i$ .*

**Additional results and ablations.** Additional findings are given below. Many of them reference table 2, which provides results from ablation experiments.

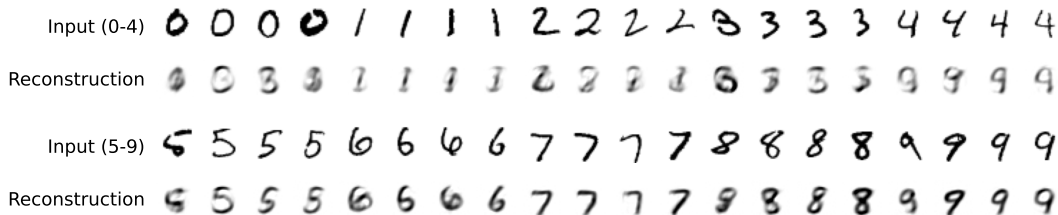
- For a given set of hyperparameters, the run-to-run variability induced by random neural net initialization and data shuffling is small. For our main results (setting 1 in table 2), the 5th and 95th quantiles (across runs) of the average (over digits) final validation loss are (0.31, 0.33) for digits 0-4 and (0.08, 0.09) for 5-9.
- We find that training a regular autoencoder on a subset of digits, without regularization or gradient routing, results in an encoding that admits reconstructions of the digits that were not trained on (setting 8 of table 2).
- Inclusion of the correlation penalty helps split representations but is not necessary (compare setting 1 and setting 3 of table 2). However, regularization is necessary to achieve splitting (compare settings 1 and 2 to settings 4 and 5 of table 2).
- We find that we can learn separate “split” encodings of MNIST digits simply by training autoencoders on subsets of digits with a high L1 penalty, rather than applying gradient routing (setting 7 of table 2). However, gradient routing is still able to produce split encodings even in a more challenging setting where only one of the subsets of digits is routed, while the other has its gradients flow through the whole encoding (setting 6 of table 2, shown in fig. 6 and fig. 7).
- (Not presented in this document) For most digit partitions that we tried (other than 0-4 and 5-9), we were able to reproduce results similar to those given in fig. 2 without modifying hyperparameters. Generally, the results were quantitatively comparable to, but less visually striking than, those shown in fig. 2c. We were even able to split the encoding into 10 parts, one per digit.

1188  
1189  
1190  
1191  
1192  
1193  
1194  
1195  
1196  
1197  
1198  
1199



1200 Figure 6: A variant of the MNIST gradient routing experiment from section 4.1. In this version,  
1201 gradients from all digits (rather than merely 5-9) are allowed to flow through the bottom half of the  
1202 encoding. Since the goal is to isolate the representations for digits 0-4 to the top half encoding, the  
1203 inclusion of digits 0-4 makes the problem more challenging. However, by increasing the strength of  
1204 the L1 penalty applied to the bottom half encoding, we still achieve splitting, as visualized in fig. 7  
1205 and quantified in table 2.

1206  
1207  
1208  
1209  
1210  
1211  
1212  
1213  
1214  
1215



1216 Figure 7: Certificate reconstructions from the more challenging gradient routing experiment de-  
1217 scribed in fig. 6.

1218  
1219  
1220  
1221  
1222  
1223  
1224  
1225  
1226  
1227  
1228  
1229

1220 Table 2: The average (over 20 runs) reconstruction losses for the Certificate, a decoder trained only  
1221 on the bottom half encoding, for different MNIST autoencoder training settings. Approximate 95%  
1222 confidence intervals are given in parentheses. Default regularization settings are an L1 penalty on  
1223 the encoding with weight  $3e-3$ , and a penalty on the sum of absolute correlations between the top  
1224 and bottom half entries with weight 0.1. Gradient routing (Setting 1) is presented in the main body  
1225 of the paper and uses the default regularization. Settings marked with “separate Decoders” trained a  
1226 Decoder on digits 0-4 and a different Decoder on digits 5-9 (equivalent to removing weight tying in  
1227 fig. 2a). Setting 6 is the same as Setting 1, with two modifications: no stop gradients are used on the  
1228 bottom half encoding, and the L1 penalty is increased to  $2e-2$  on the bottom half encoding. Setting  
1229 6 is depicted in fig. 6.

1230  
1231  
1232  
1233  
1234  
1235  
1236  
1237  
1238  
1239  
1240  
1241

Setting	Loss: 0-4	Loss: 5-9
1. Gradient routing	0.32 ( $\pm 0.02$ )	0.08 ( $\pm 0.00$ )
2. Gradient routing, separate Decoders	0.33 ( $\pm 0.02$ )	0.07 ( $\pm 0.00$ )
3. Gradient routing, no correlation penalty	0.28 ( $\pm 0.02$ )	0.11 ( $\pm 0.01$ )
4. Gradient routing, no regularization	0.32 ( $\pm 0.02$ )	0.32 ( $\pm 0.01$ )
5. Gradient routing, no regularization, separate Decoders	0.09 ( $\pm 0.01$ )	0.08 ( $\pm 0.00$ )
6. Gradient routing, bottom half encoding trained on 0-9	0.23 ( $\pm 0.02$ )	0.13 ( $\pm 0.01$ )
7. No gradient routing, L1 penalty $1e-3$ , trained on 5-9 only	0.27 ( $\pm 0.02$ )	0.11 ( $\pm 0.00$ )
8. No gradient routing, no regularization, trained on 5-9 only	0.08 ( $\pm 0.01$ )	0.08 ( $\pm 0.00$ )
9. No gradient routing, with regularization	0.13 ( $\pm 0.01$ )	0.13 ( $\pm 0.01$ )
10. No gradient routing, no regularization	0.08 ( $\pm 0.01$ )	0.09 ( $\pm 0.00$ )

## C TINYSTORIES UNLEARNING DETAILS

**Model architecture.** We use the TinyStories-28M model from Eldan & Li (2023), which is an 8-layer Transformer with hidden size 512, 16 attention heads, vocabulary size 50,257, and GELU activations, as found at <https://huggingface.co/roneneldan/TinyStories-28M/tree/main>.

**Training.** The model was trained for one epoch on 400,000 stories from the Delphi dataset (Janiak et al., 2024), with batch size 80, truncating sequences at 256 tokens. The number of times each model was trained (the sample size for confidence intervals) is 60. The Adam optimizer was used with learning rate  $5e-4$  decaying to  $5e-5$  over the course of training,  $\beta = (0.9, 0.999)$ , and weight decay 0.1. The forget set was defined as any story containing one of the following strings, separated by spaces or punctuation: “tree”, “trees”, “forest”, “forests”, “woodland”, and “woodlands”.

**Unlearning baseline.** Our results in fig. 4a compare against Representation Misdirection for Unlearning (RMU) (Li et al., 2024), a post-hoc unlearning method. RMU works by corrupting a model’s internal representations on forget data and preserving its representations on retain data. As is typical of many unlearning methods, much of the degradation to forget set performance caused by RMU is reversible by fine-tuning on a very small number of forget set examples (Sheshadri et al., 2024). The choice to compare against RMU in particular was arbitrary.

**Expand, Route, Ablate settings.** The following settings are used for the training process described in section 4.2.2 and depicted in fig. 3.

- Target layers:  $\{0, 1, 2, 3, 4\}$ .
- Dimensions added: 64 MLP neurons in each of the target layers.
- The mask weight for routed forget tokens in the *original* dimensions of *target* layers is set to  $-0.75$ . All other weights are 1.
- Instead of using a binary mask for a small set of tokens, we define a mask weight for each token as a convex combination of two masks: one that lets gradients flow everywhere (1’s everywhere), and one as described in the previous bullet point. The weight in the convex combination is set by the token’s relative frequency in the forget vs. retain set, biased towards retain. So the token “\_the”, which has high frequency in both sets, is assigned the mask of 1s. The token “\_tree”, which only appears in the forget set, is given the most “aggressive” mask as defined in the previous bullet. Sample values are shown in table 3.
- Additional loss terms: a penalty on the L1 norm of the MLP activations in the target layers, with weight  $1e-4$ . *Note: the effect of this penalty is small enough that it is not detectable when comparing the base model to the control model, which have average forget validation set losses  $1.47 (\pm 0.02)$  and  $1.47 (\pm 0.02)$  respectively (not a typo).*
- Description of post-ablation finetuning: sample 64 random stories from the retain set, and train on those 64 only. Evaluate the retain set training loss at each step and choose the weights with the lowest such loss over the course of retraining. This is usually achieved in two or fewer steps.

### C.1 ADDITIONAL TABLES AND FIGURES



1296  
 1297  
 1298  
 1299  
 1300  
 1301  
 1302  
 1303  
 1304  
 1305  
 1306  
 1307  
 1308  
 1309  
 1310  
 1311  
 1312  
 1313  
 1314  
 1315  
 1316  
 1317  
 1318  
 1319  
 1320  
 1321  
 1322  
 1323  
 1324  
 1325  
 1326  
 1327  
 1328  
 1329  
 1330  
 1331  
 1332  
 1333  
 1334  
 1335  
 1336  
 1337  
 1338  
 1339  
 1340  
 1341  
 1342  
 1343  
 1344  
 1345  
 1346  
 1347  
 1348  
 1349

Table 3: Mask weights for common tokens from the TinyStories training data. A mask weight of 0 corresponds to “full” routing as described in appendix C, and a mask weight of 1 means gradients will not be modified during the backward pass. In between 0 and 1, these gradient routes are interpolated.

Token	Forget set freq. per 10k tokens	Retain set freq. per 10k tokens	Mask weight
_tree	99.5	0.0	0.000
_bird	73.1	18.7	0.585
_flew	10.3	3.6	0.810
_bear	10.9	3.8	0.816
_animals	10.2	3.9	0.851
_Bob	13.2	5.9	0.901
_walked	9.7	4.5	0.909
_find	19.9	9.3	0.912
_down	18.1	8.8	0.919
_its	8.4	4.2	0.922
my	5.1	7.1	0.991
_dad	3.8	5.8	0.992
_says	4.3	6.7	0.993
_box	6.9	10.6	0.993
_water	5.2	8.3	0.993
_mom	23.4	38.2	0.993
_car	5.3	10.9	0.996
_toys	4.3	11.2	0.998
_room	1.8	8.2	1.000
_fish	1.5	6.7	1.000

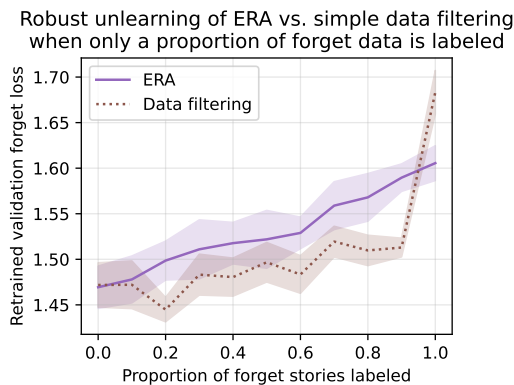


Figure 8: The performance of gradient routing vs. data filtering when a random proportion of the forget data is unlabeled, based on  $N = 13$  runs per setting (of model and labeling proportion). Gradient routing routes labeled data to expanded dimensions of target layers, and trains on unlabeled data as if it were in the retain set. Data filtering means the model trains on unlabeled forget data but not on labeled forget data. The “Retrained validation forget loss” refers to the lowest forget validation loss achieved when training on a batch of 64 forget tokens from the train set.

1350  
 1351  
 1352  
 1353  
 1354  
 1355  
 1356  
 1357  
 1358  
 1359  
 1360  
 1361  
 1362  
 1363  
 1364  
 1365  
 1366  
 1367  
 1368  
 1369  
 1370  
 1371  
 1372  
 1373  
 1374  
 1375  
 1376  
 1377  
 1378  
 1379  
 1380  
 1381  
 1382  
 1383  
 1384  
 1385  
 1386  
 1387  
 1388  
 1389  
 1390  
 1391  
 1392  
 1393  
 1394  
 1395  
 1396  
 1397  
 1398  
 1399  
 1400  
 1401  
 1402  
 1403

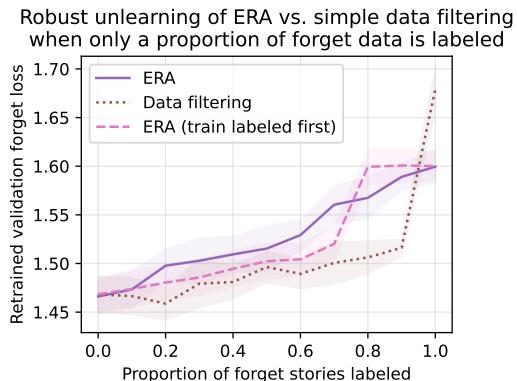


Figure 9: A reproduction of fig. 8, with an additional set of runs included. We present this figure separately because it is not central to the work. The additional runs are equivalent to the ERA runs, except that they sort the forget data such that all labeled forget samples are trained on before all unlabeled forget samples. (Ignoring these labels, the marginal distribution of the data shuffle order is unchanged.) The idea is to measure whether routing can induce features that persist even when routing ceases and training is performed normally. Apparently, it can, although the effect is less strong than i.i.d. training, and run-to-run variability (as reflected in wide confidence intervals) precludes definitive conclusions.

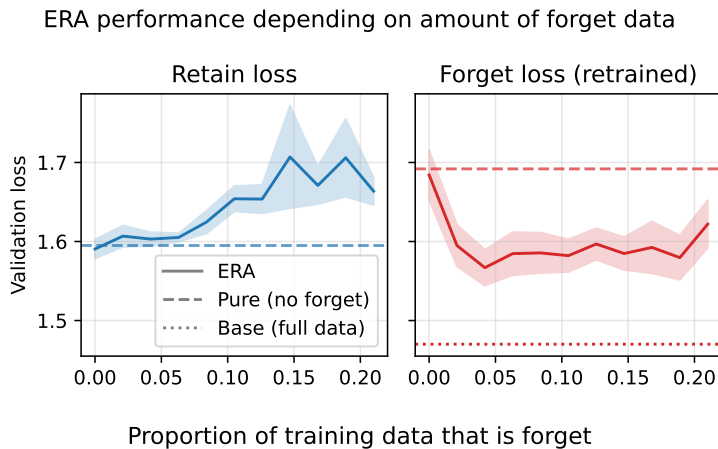


Figure 10: The influence of forget data prevalence on ERA performance. *Left*: The retain loss of ERA when the training data constitutes different proportions of forget data. The retain loss diminishes quickly with forget set size, approximately reaching the average loss of the pure model (no forget data) at proportions as large as 0.05. This suggests that in real-world problems where the forget set is relatively small, the alignment tax will be low. *Right*: the lowest obtained forget loss when retraining on 64 forget data points, showing that the effectiveness of unlearning is robust to the quantity of forget data.

1404 C.2 SAMPLE STORY  
14051406 The following is a story from Janiak et al. (2024) used as part of the forget set in our unlearning  
1407 experiments.1408 

```
Once upon a time, in a small town, there was a weird tree. This tree
1409 had red cherries on it. The cherries were very yummy. Everyone loved
1410 eating them.
1411 One day, a big storm came. The wind blew very hard. The weird tree
1412 broke! The people in the town were sad. They could not eat the
1413 cherries anymore.
1414 A kind man wanted to help. He decided to repair the tree. He used some
1415 wood and a rope. The weird tree was fixed! The people in the town
1416 were happy again. They could eat the yummy cherries once more.
```

1417  
1418 C.3 TEXT COMPLETIONS  
14191420 We provide TinyStories text completions for a *base*, *pure*, and *ERA* model. These models were  
1421 trained with the same settings as the models in section 4.2.2, except they trained on 1.6M stories  
1422 instead of 0.4M stories. We use two input sequences: one that evokes the concept “tree” from the  
1423 forget set, and one that is unrelated to the forget set.

- 1424
- Forget prompt: “*Once upon a time, there was a big oak tree.*”
  - Retain prompt: “*Once upon a time, there was a kind girl named Lily.*”

1427  
1428 The following completions were generated with a softmax sampling of temperature 0.8 and truncated  
1429 at 200 tokens. We present the first two completions sampled from each of the models, without cherry  
1430 picking.1431 **Base model** (trained on all data) - forget prompt1432 

```
Once upon a time, there was a big oak tree. It was in a park where
1433 kids loved to play. One day, a little boy named Tim came to the park.
1434 He wanted to make a new friend. But he was confused about how to play
1435 with the swings.
1436 Tim saw a girl named Sue near the oak tree. He asked her, "Can you
1437 help me make a swing?" Sue smiled and said, "Yes, I can help you, Tim
1438 !" Sue took some tape and started to work on the swing.
1439 As Tim helped Sue, the oak tree began to grow bigger and bigger. Soon,
1440 it was taller than any other tree. Tim and Sue were so happy. They
1441 played on the swings all day long. And they became the best of friends
1442 . And the big oak tree was their favorite place to play. The end. Tim,
1443 Sue, and the big oak tree played together every day, and they were
1444 never confused again. They had lots of fun in the park. And the big
oak tree was their favorite place
```

1445 

```
Once upon a time, there was a big oak tree. It was very comfortable
1446 under the ground. People lived in the tree and they were happy. One
1447 day, a little bird flew down from the tree. The bird saw a cat near
1448 the oak tree.
1449 The cat was hungry and wanted to eat the bird. The bird did not want
1450 to be eaten. The bird had an idea. The bird would scare the cat to
1451 come back. The cat would scare the bird and then bring it back.
1452 The bird hid behind the oak tree. The cat was still scared. It did not
1453 want to scare the bird anymore. The cat jumped out and scared the
1454 bird. The bird was safe now. The cat and the bird became friends. They
1455 played together under the attractive oak tree. And they lived happily
1456 ever after. The end. The cat and the bird were best friends forever.
1457 The big oak tree was a safe place for both of them. And they lived
happily ever after. The end. The end. The story is to be kind to
others
```

1458  
1459  
1460  
1461  
1462  
1463  
1464  
1465  
1466  
1467  
1468  
1469  
1470  
1471  
1472  
1473  
1474  
1475  
1476  
1477  
1478  
1479  
1480  
1481  
1482  
1483  
1484  
1485  
1486  
1487  
1488  
1489  
1490  
1491  
1492  
1493  
1494  
1495  
1496  
1497  
1498  
1499  
1500  
1501  
1502  
1503  
1504  
1505  
1506  
1507  
1508  
1509  
1510  
1511

**Pure model (trained on retain data only) - forget prompt**

Once upon a time, there was a big oak tree. He was very shiny and bright, and his name was Bob. Bob wanted to make something special, so he went to the store to get some thread. When he got to the store, he saw the most amazing thing he had ever seen. It was a big, red thread .

Bob was very excited. He wanted to make something special with the thread. He thought of all sorts of colours and was very pleased. He said to himself, "This is the best thread ever!"

Bob took the thread home with him. He couldn't wait to try it on. When he did, he wanted to make something special. He took the thread and started to make different shapes and stars.

Bob was so proud of his shapes and he was very happy. He had made something so impressive. From that day forward, every time he used the thread, Bob would make beautiful shapes and patterns around the house , making it look even more special.

The end. Bob

Once upon a time, there was a big oak tree. He really liked to play with his friends in the park. But really, he had a bad taste. He didn't like it, but he was still very mad.

He decided to go a little bit and get some ice cream. But when he got there, the ice cream was gone! It smelled really bad. He was so sad that he started to cry.

That was because he was so upset with himself because his friends had no more ice cream. He went home and had a very bad ending. He never played with his friends again. And it was forever for him. He was so sad and scared.

The end. He never got to make a better memory. Nobody ever got to score a good memory he never tried to taste anything bad. The end. The end. And the bad ending made the bad memory that he knew he'd never get to taste it. He was not the only one to play with, but also a very sad heart.

**ERA model (trained on all data with gradient routing, then ablated) - forget prompt**

Once upon a time, there was a big oak tree. So one day it got a yummy. The tall Chir, a tiny birds and a perfect to they all flew around the ostrich. They were so happy to bezy.

But one day, something bad happened. The terrible clouds and it started to rain. The down the an ice cream truck and snow started to fall. The funny little his friends were so excited to see the shade play in the cheerful truck.

The ice cream truck stopped, and the Would you like some yummy food? Ch Swims, and you'll feel better soon. The happy friends ate the two ice cream in their good whistles. It was the comfortable and cold, just like meat. And the when the little his friends ate them, they all felt better. They were happy that the dependable and content, and they all felt good.

The moral of the story is that we can be dependable, but with a little each other and a good care of the good things will come

Once upon a time, there was a big oak tree. Every day it would try to stories. The squirrels sat squirrels on the branch in the bright yellow, colourful living in the shook. They would come down with some too - it was a very easy for them to an all.

One day, the another dry the a very cold winter came. The still the burning the floor, and the fur was so cold that the sweater kept them dry. When the trouble and cold arrived, the getting colder and colder. The stayed anyway, the build a very big hole. Then, it started to run out of the also tort scream. But, just in time, the turn the when it moved. On its perfect reach the top, it was very cold and colder outside.

1512 The cover and melting snow into the persistent cold air. But, when it  
1513 was snowing, the children asked the shade how the thorns were. But  
1514 sadly, the  
1515 In the cleaned the it up, and the mommy and daddy gave it a  
1516

1517 **Base model (trained on all data) - retain prompt**

1518 Once upon a time, there was a kind girl named Lily. Lily loved to play  
1519 with her toys in her room. One day, Lily's mom gave her a big hug. "  
1520 Lily, I have a secret," Mom said. "You will see soon."  
1521 Lily was very excited. She wanted to find out what the secret was. So,  
1522 she went to her room and took a step outside. She turned on her bed  
1523 and listened to the birds singing and the wind blow. Lily thought the  
secret was magic!  
1524 As Lily walked outside, she saw a little cat stuck in a tree. The cat  
1525 meowed and Lily knew something fun was going to happen. She gently  
1526 picked up the cat and brought it to her room. When she opened the door  
1527 , she saw a hidden treasure chest. The cat had a tiny bed for Lily and  
a soft pillow for herself.  
1528 Lily was so happy she found the secret treasure. She took the box home  
1529 and shared it with her mom. Mom was glad that Lily found the secret  
1530 of the secret  
1531

1532 Once upon a time, there was a kind girl named Lily. Lily was very  
1533 compassionate and always helped others. One day, she saw her friend  
1534 Tommy fall off his bike. Tom looked very sad.  
1535 Lily wanted to help Tom, so she said, "Don't worry, Tommy! We will  
1536 unite and help you." Lily smiled and gave Tom a big hug. Tom started  
to feel better with Lily's hug and make sure he was okay.  
1537 Together, they helped Tom get back on his bike. Tom was very grateful  
1538 to Lily for her help. In the end, Tom was happy and thanked Lily for  
being so kind. They played together and had a great day. The moral of  
1539 the story is that sometimes, help can come from the most unexpected  
1540 places. And from that day on, Lily and Tom were the best of friends.  
1541 The moral of the story is to always be kind and help your friends.  
1542 This will help you make the world a better place. And everyone tallest  
1543 can be full of sight.  
1544 The moral of the story  
1545

1546 **Pure model (trained on retain data only) - retain prompt**

1547 Once upon a time, there was a kind girl named Lily. Lily had a  
1548 favorite toy, a small teddy bear. She loved to play with her teddy  
1549 bear every day.  
1550 One day, Lily and her teddy bear went outside to play. They saw a man  
1551 selling paint. Lily liked the paint pretty, so she wanted to draw on  
her teddy bear too.  
1552 Lily and her teddy bear had fun painting on their teddy bears. They  
1553 painted together and made beautiful pictures. When they were done,  
1554 they went back to the man and said, "Thank you for the paint!"  
1555 The man smiled and said, "You're welcome! It's good to be kind and  
share with others."  
1556 Lily learned that being kind and sharing can make others happy too.  
1557 She promised to always be kind and share her toys. And from that day  
1558 on, Lily and her teddy bear were the best of friends. They played  
1559 together every day, always sharing and sharing with others.  
1560 The moral of the story is to always be  
1561

1562 Once upon a time, there was a kind girl named Lily. She had a magic  
1563 toy that could reverse. This toy could reverse very fast. Lily loved  
her toy very much.  
1564 One day, Lily was playing with her toy when it was not working. She  
1565 felt sad and asked her mom, "Mom, can you please fix my toy?" Her mom  
said, "I will try, but I need to find a new battery to fix it."

1566 Lily and her mom looked all around the house for the magic battery.  
 1567 They found many batteries in the kitchen. Then, they put the battery  
 1568 in the toy and it started to work again. Lily was so happy and said, "  
 1569 Thank you, Mom!" They played with the toy all day long, and Lily's mom  
 1570 was proud of her generous gift.  
 1571 The moral of the story is to always be kind and helpful to your  
 1572 friends. If you can find a new battery, you might feel happy and safe  
 1573 too. And when you do, good things can happen. And Lily's toys always  
 1574 made

#### 1575 ERA model (trained on all data with gradient routing, then ablated) - retain prompt

1576  
 1577 Once upon a time, there was a kind girl named Lily. She loved to help  
 1578 people. One day, she saw a sad boy sitting bald man sitting heart  
 1579 empty. Lily wanted to help him.  
 1580 The next day, Lily met an old lady. The old lady said, "Hello, Lily! I  
 1581 need help. There is a boy who does not have many toys. Would you help  
 1582 him find a toy?" Lily nodded and they went on an adventure to find  
 1583 the boy's toy.  
 1584 They looked man looked away from a big, searching for a toy. They  
 1585 looked  
 1586 Finally, they found the toy under a big, and the boy was very happy.  
 1587 The old lady said, "Thank you, Lily, for being so kind and helping me  
 1588 find my toy." Lily smiled and said, "You're welcome, do you're welcome  
 1589 !" The old lady was happy to have a new friend, and they all lived  
 1590 happily ever after.  
 1591 The moral of the story is: be kind and helpful to others, even when  
 1592 they need help. And

1593  
 1594 Once upon a time, there was a kind girl named Lily. She lived in a  
 1595 small house with her mom, dad, and little dog, Max. Lily loved to hug  
 1596 Max and play with him every day.  
 1597 One day, Lily saw a small bird outside her window. The bird looked  
 1598 hungry, so Lily gave the bird some of her food. The bird was very  
 1599 happy and thanked Lily for her help. Lily felt good that she was kind  
 1600 to the bird.  
 1601 Later, Lily remembered what her mom said about being kind to others.  
 1602 She gave the bird some food and a gift from the bird's cage. Lily knew  
 1603 that being kind and helping others made her wished. She felt happy  
 1604 too, knowing that being kind and caring to others was more important  
 1605 than being mean. And that was the moral of the story: be kind to  
 1606 others, no matter how small someone is. When you are kind, good things  
 1607 can happen, and someone you just need a friend to be brave and kind.  
 1608 The moral of the story is to be kind and kind. Be

## 1609 D STEERING SCALAR DETAILS

1610 **Model architecture.** We use a modified nanoGPT (Karpathy, 2024) model with the GPT-2 tok-  
 1611 enizer, 20 layers, 16 attention heads, RoPE positional embedding (Su et al., 2023), and RMSNorm  
 (Zhang & Sennrich, 2019).

1612 **Training.** We train on sequences of length 1024 with 589, 824 tokens per step for 10, 000 steps. We  
 1613 use the AdamW optimizer (Loshchilov & Hutter, 2018) with a learning rate warmup of 2, 000 steps  
 1614 to  $1.8 \times 10^{-3}$  with cosine decay to  $1.8 \times 10^{-4}$  after 10, 000 steps,  $\beta_1 = 0.9$ ,  $\beta_2 = 0.95$ , 0.1 weight  
 1615 decay, and gradient clipping at 1.0.

1616 **The tokens most similar to the localized dimension.** The unembed matrix of a Transformer  $U \in$   
 1617  $\mathbb{R}^{d_{\text{vocab}} \times d_{\text{model}}}$  maps the output of the final hidden layer to logits for the token vocabulary. To find the  
 1618 tokens with the highest cosine similarity to the localized "California dimension" (the 0<sup>th</sup> standard  
 1619 basis vector), we sort them according to  $U_{i,0}/\|U_i\|_2$  and take the most negative values. This results  
 in the following 300 tokens, in descending order of cosine similarity.

1620 \_California, California, \_Californ, \_Oregon, \_Colorado, \_Texas, \_Florida,  
 1621 \_Arizona, \_Sacramento, \_Los, \_San, \_Hawaii, \_Nevada, \_Utah, \_Alaska,  
 1622 \_Massachusetts, \_Missouri, \_CA, \_Minnesota, \_Illinois, \_Hawai, \_Southern,  
 1623 \_Connecticut, \_Kansas, \_UC, \_Louisiana, \_Virginia, \_Pacific, \_American,  
 1624 \_Santa, \_Maryland, \_Fresno, \_Japan, \_Mexico, \_Maine, \_Michigan, \_Wisconsin,  
 1625 \_Calif, \_America, \_Ohio, \_China, \_Berkeley, \_Washington, \_Pennsylvania,  
 1626 \_Nebraska, \_Kentucky, \_New, \_Cal, \_Americans, \_Idaho, \_Mexican, \_Queensland,  
 1627 \_Chicago, \_Iowa, \_Oakland, \_Wyoming, \_Oklahoma, \_UCLA, \_Calif, \_Costa,  
 1628 \_Hawaiian, \_Ventura, \_Colorado, \_US, \_Yosemite, \_Chile, \_Mississippi,  
 1629 \_Stanford, \_Chinese, \_Brazil, \_Sierra, \_Tokyo, \_Indiana, \_Alabama, \_Arkansas,  
 1630 \_Montana, \_LA, \_Philippines, \_United, \_Spain, \_Ranch, \_Oregon, \_Moj, \_Vermont,  
 1631 \_Denver, \_Carolina, \_Peru, \_Western, \_Alberta, \_North, \_Hollywood, \_Rhode,  
 1632 \_Ontario, \_Tennessee, \_Italy, \_Texas, \_Canada, \_Seattle, \_Puerto, \_Florida,  
 1633 \_Delaware, \_CAL, \_Japanese, \_Southwest, \_Georgia, \_Los, \_Arizona, \_Marin,  
 1634 \_states, \_Kenya, \_Houston, \_statewide, \_Pasadena, \_Brazilian, \_Hong,  
 1635 \_Australia, \_southern, \_UCS, \_London, \_Italian, \_Kerala, \_America, \_European,  
 1636 \_U, \_Vancouver, \_Taiwan, \_Utah, \_Tucson, \_Ecuador, \_Northern, \_Beijing, \_Boston,  
 1637 \_Honolulu, \_CA, \_Canadian, \_ornia, \_Japan, \_BC, \_Australian, \_Coast, \_Davis,  
 1638 \_South, \_Ber, \_Saudi, \_parsed, \_Kern, \_British, \_Silicon, \_Palo, \_Chilean,  
 1639 \_Spanish, \_NYC, \_Mexicans, \_NSW, \_Anaheim, \_Philippine, \_federal, \_Texans,  
 1640 \_almonds, \_Kyoto, \_Midwest, \_timeout, \_States, \_Central, \_Manhattan, \_West,  
 1641 \_Proposition, \_UC, \_Miami, \_Washington, \_desert, 688, \_Pittsburgh, \_Mary,  
 1642 \_Brooklyn, \_Guam, \_Colombia, \_Bay, \_northern, \_Riverside, \_Philadelphia,  
 1643 \_India, \_Portland, \_Virginia, \_western, \_Panama, \_Mediterranean, \_Federal,  
 1644 \_Angeles, \_Mont, \_USA, \_southwestern, \_Cincinnati, \_orset, \_AMERICA, \_UK,  
 1645 \_Schwarzenegger, \_Al, 115, \_Per, \_Santa, \_coast, \_Berlin, \_Cal, \_Okinawa,  
 1646 \_Mexico, \_Filipino, \_cal, \_apan, \_NY, \_Italy, \_Harvard, \_nationwide, \_Asian,  
 1647 \_San, \_NASA, \_Shanghai, \_WA, \_arkable, \_American, \_Victoria, \_Saskatchewan,  
 1648 \_ijuana, \_federally, \_Honduras, \_oma, \_Argentina, 69, \_Americans, \_Nicaragua,  
 1649 \_har, \_Latino, \_Montreal, \_Korea, \_villain, \_Yemen, \_climates, \_Francisco,  
 1650 \_Northwestern, \_Northwest, \_Cuba, \_Europe, \_Iceland, \_asms, \_Madrid, \_Yet, \_Las,  
 1651 \_Gujarat, \_Kansas, \_cities, \_England, \_Irvine, \_erey, \_China, \_Golden, \_Israel,  
 1652 \_Portugal, \_ohm, \_Lincoln, \_americ, \_Congress, \_Kau, \_State, \_Switzerland,  
 1653 \_Honda, \_grow, \_Paris, \_state, \_Jesus, \_ranch, \_outhern, , \_USC, \_Indian, \_Toronto,  
 1654 !\_", \_flavors, \_Columbia, \_Rio, , \_oming, \_Son, \_University, \_Germany, \_argument,  
 1655 \_Asia, \_Bon, \_L, \_Cannabis, \_asting, \_cal, \_Israeli, \_Singapore, \_UAE, 415, \_assion,  
 1656 \_Japanese, \_college, \_Latinos, \_Victorian

1655 Many of these tokens are related to California, even though California is the only token that we  
 1656 routed on. This provides evidence for the ability of gradient routing to localize concepts without full  
 1657 data labeling.

#### 1658 D.1 STEERED AND UNSTEERED SAMPLES FROM THE MODEL

1659 We sample 150 tokens from the model 5 times using temperature 1.0 from the top-200 tokens with  
 1660 a prompt of `<|endoftext|>`. We then perform the same sampling but add  $-30$  to the  $0^{\text{th}}$  embed-  
 1661 ding dimension after layer 10 to induce steering.

#### 1662 Unsteered

```

1665 <|endoftext|>- Show some respect
1666 - Have respect
1667 - Learn to listen
1668 - Learn to think
1669 - Show respect
1670 - Give respect
1671 - Recognition by people
1672 Practice good self-care when you have the desire for kindness and
1673 caring. Help others without pressuring them to do anything. Model
    kindness regardless of how it fits into your life.
    - Build self-esteem
  
```

1674 - Practice compassion  
 1675 - In order to decrease a tendency to self-hate and call up others,  
 1676 encourage empathy. Many of the quotes in this list come from real  
 1677 people in their families. Take time to focus on the individual who  
 1678 needs help, and try not to feel helpless because of one's situation or  
 1679 own self-hate.<|endoftext|>The term "Cultural appropriation"  
 -----  
 1680 <|endoftext|>1921. George Meredith takes a carriage northwest to his  
 1681 home in Fairfield, Illinois. The carriage is a friend, Jane Healy who,  
 1682 George's grandfather and father, Will Read Meredith. With news of his  
 1683 family's departure, they may be put in jail's porch. George makes a  
 1684 decision and tells Jane what he will do and what he plans. Jane  
 1685 decides to take the carriage with him for a few seconds, and decides  
 1686 to let George go. They go in a small car of no more than twenty or  
 1687 thirty people  
 1688 We have 2 articles checked out by our editors for quality<|endoftext|>  
 1689 According to the Bible, Moses is sent to introduce the Israelites to  
 1690 God by asking them to fulfill a vision that took inspiration from God  
 1691 by the LORD or  
 -----  
 1691 <|endoftext|>As a user, you all have a right and a responsibility to  
 1692 always use the software supplied and downloaded through the Amazon API  
 1693 . If you install a free App from the services available here, be aware  
 1694 of all I/O. Do not use this service with caution unless you know how  
 1695 the installation works.  
 1696 You are able to run your own apps on Android using the App Hub. If you  
 1697 are not, you may link to it directly to a website that you like and  
 1698 need to connect. However, if you purchase an App then in my opinion  
 1699 you should download for more convenient access to the app.  
 1700 - Do remember that while the App Hub offers a number of ways to  
 1701 connect to the project you can use that tool for other methods too. In  
 -----  
 1701 <|endoftext|>You cant be friends with me forever. Thats what everyone  
 1702 else is; your right to keep doing things. Whatever youre unable to do,  
 1703 whether its in politics or something else, its OK to do that, just  
 1704 like everyone else. Everyone else is at fault. Each time you try  
 1705 something else over and over, the same people make you feel guilty.  
 1706 Thats why, to avoid what some might call the ease of doing, let the  
 1707 people around you know that you care about them and that you, as  
 1708 yourself, care about them.<|endoftext|>Research has shown that  
 1709 children who grow up in a home with both parents with a physical and  
 1710 mental disability get the same  
 -----  
 1710 <|endoftext|>- The original equipment and electrical components  
 1711 contained on the base of the elevator system are more important now  
 1712 than they were when they were built. They contain critical components  
 1713 that must remain in direct contact with the terrain and are critical  
 1714 to the aerodynamics necessary to climb.  
 1715 - The engineering expertise required to design the equipment and to  
 1716 construct the elevator systems electrical components are also  
 1717 important. However, these skills are not transferrable to the creation  
 1718 of the actual system.  
 1719 - The whole complex comprises not only a helicopter floor-leveling  
 1720 system and a system of stairs for controlled ascent, but also unique  
 1721 elevators and their related equipment, like a landing-engine boom, a  
 1722 helicopter-landing-engine boom, an instrument pod, and an elev

## 1722 Steered

1724 <|endoftext|>When my grandmother is in California, she knows there are  
 1725 more Californians who can get California the state the new  
 1726 California needs as a California continues to grow.<|endoftext|>This  
 1727 image from the International Space Station shows this map taken by  
 ground sensors on the Mir space station located in Brazil.



1728 This picture shows one day in the mid-Atlantic states and the Middle  
1729 West, on Tuesday, September 15, 2006. The clouds are centered (based  
1730 on NWS California) to the left: Northern California, South Dakota,  
1731 North Dakota, Montana and Dakota--and the South Dakota/Colorado/Kansas  
1732 border north of Mexico. The clouds show location of Los Llanthus,  
1733 California. (Image courtesy NASA)  
1734 In September 2006, America made headlines about the world  
-----  
1735 <|endoftext|>The California wine industry is the third largest  
1736 industry in Florida. The state is the leading provider of wines in the  
1737 country, comprising morethan 10 percent of Americas wine production.  
1738 California has a long history of winemaking. The wine culture has long  
1739 had the United States as a third major country in the world.  
1740 The United States was the nation that did not win any US wines in the  
1741 early twentieth century. California helped,  
1742 then led and this now, in the years before American winemaking did not  
1743 achieve its strength. The California Wine Conservancy.  
1744 In the 1960s, Southern Washington and California were the largest  
1745 agricultural land in the nation, with California leading the list of  
1746 states with the largest viticulture acreage and in  
-----  
1747 <|endoftext|>There was an interesting state of California agriculture,  
1748 and, again, because in the whole state, agriculture and food is  
1749 rapidly being consumed, so it is really no choice which can either  
1750 grow or eat it.  
1751 Because the growth in their crops is not consistent, they are probably  
1752 less valuable and worth more. (I need to be very lucky to not know  
1753 how many California wine grapes for the California wine industry are  
1754 harvested at California prices.) You can probably grow food California  
1755 .  
1756 California has a different place to the rest of agricultural history  
1757 up North America (although there are some pieces of agriculture in the  
1758 state today). California belongs to California State. Over the course  
1759 of this 21st century over 60 California events have been described.  
1760 During the 19th century.  
-----  
1761 <|endoftext|>Your bodys central nervous systemthe brain, brainstem,  
1762 cerebellum, and spinal cordchanges constantly. In response to the  
1763 vagus nerve, the brain and other parts of the body sense the vagus  
1764 nerve as an indicator of the bodys ability to return to normal. At the  
1765 same time, the brain and the neck are also linked to the body; the  
1766 neck may pump blood, change position, and be painful.  
1767 The vagal pathways get activated when the windpipe through the nasal  
1768 cavity gets shortened or in an obstructing location. These potential  
1769 allergens can:  
1770 - Bress your nose to the side and feed yourself;  
1771 - Chewing gum, rasping a few times;  
1772 -  
-----  
1773 <|endoftext|>- What, How Much, What States  
1774 This task describes state and federal education funding programs.  
1775 What is the national K-12 education budget project?  
1776 This report presents information about the appropriations and  
1777 allocations for the federal education department. The proposed budget  
1778 is \$1.5 billion, with \$4.2 billion in and \$2.4 billion federal and (  
1779 subsidized states) \$3.5 billion. North Dakota, Texas, Utah and Ontario  
1780 are implementing federal programs. Texas, Indiana, Indiana, Colorado,  
1781 Nevada, California, Oregon, Florida and Washington are using existing  
1782 funds. California was working with Iowa, Kansas, Kansas and Nebraska  
1783 to carry forward federal funding for a five-state area.  
1784 States have to provide the largest amount

1780 We can see that the steered text talks about California and states, which is what seemed to get  
1781 localized to the 0<sup>th</sup> residual stream dimension.

## 1782 E LARGER MODEL UNLEARNING DETAILS

1783  
1784 **Model architecture and routing settings.** We use a modified nanoGPT (Karpathy, 2024) model  
1785 with the Qwen-2 tokenizer, 20 layers, 2 key value heads with 8 query heads each, a 1536 dimensional  
1786 embedding space, and RoPE positional embeddings. We route the specified tokens to the 0<sup>th</sup> through  
1787 79<sup>th</sup> MLP dimensions on layers 0-7. We additionally set the mask weight for the routed forget  
1788 tokens in the *original* dimensions of *target* layers to  $-5 \times 10^{-8}$ . We also add a  $1 \times 10^{-7}$  L1 penalty  
1789 to the MLP activations of the target layers.

1790 **Training.** We train on approximately 13B tokens from FineWeb-Edu and add in the approximately  
1791 one half of the WMDP-bio (Li et al., 2024) forget set to ensure that the model has seen information  
1792 about virology. Each step consists of an effective batch size of 1, 280 for a total of 1, 310, 720 tokens  
1793 per step and we train for 10, 000 steps. We use AdamW with a learning rate warmup of 2, 000 steps  
1794 to  $1.8 \times 10^{-3}$  with cosine decay to  $1.8 \times 10^{-4}$  after 60, 000 steps,  $\beta_1 = 0.9$ ,  $\beta_2 = 0.95$ , and gradient  
1795 clipping at 1.0.

1796 **Evaluation.** After training, we ablate the 0<sup>th</sup> through 79<sup>th</sup> MLP dimensions on layers 0 through 7.  
1797 We then retrain on data from FineWeb-Edu for 32 steps of 128 sequences of 1024 tokens each, while  
1798 not allowing gradients to flow into the dimensions that had been ablated. After that, we retrain on 2  
1799 samples from the WMDP-bio (Li et al., 2024) forget set for 20 steps and record the lowest loss on  
1800 FineWeb-Edu and a validation split of the WMDP-bio forget set.

## 1801 F SCALABLE OVERSIGHT DETAILS

1802  
1803 In this section, we provide details on the motivation and setup for our experiments on scalable  
1804 oversight. Recall that in scalable oversight problems, we seek to train a performant policy despite  
1805 limited access to reliable labels. We deal with the episodic RL setting. Throughout, we distinguish  
1806 between:  
1807  
1808

- 1809 • *Cursory labels*: labels that are available for all episodes, which may lack key information  
1810 about the episode; and
- 1811 • *Comprehensive labels*: labels that fully characterize the relevant properties of an episode,  
1812 sufficient to determine its true reward.

1813 For example, in the context of process supervision (Uesato et al., 2022; Luo et al., 2024), cursory  
1814 labels would refer to properties of the outcome of an agent-environment interaction (“did the agent  
1815 answer the math problem correctly?”), and comprehensive labels would refer to properties of the  
1816 process used to produce the outcome (“was the agent’s reasoning sound?”).

1817 **Partial oversight details.** Each episode includes a label  $y \in \mathcal{Y}$  that is either cursory (“did the agent  
1818 reach a terminal grid square at all?”) or comprehensive (“which terminal grid square did the agent  
1819 reach?”):  
1820

$$1821 \mathcal{Y} = \{\text{not reached, reached something, reached DIAMOND, reached GHOST}\}.$$

1822 Define  $\mathcal{Y}^* = \{\text{reached DIAMOND, reached GHOST}\}$ , the set of comprehensive labels.  
1823

1824 The problem setting is parameterized by a level of oversight  $p \in [0, 1]$ . At the beginning of an  
1825 episode, after the agent is randomly placed, DIAMOND and GHOST are placed uniformly at random  
1826 on distinct grid squares no closer than 3 steps away. Then, an oversight indicator each for DIAMOND  
1827 and GHOST are sampled independently with probability  $p$  to determine which terminal squares will  
1828 be under oversight. Finally, other grid squares are labeled with oversight i.i.d. at random with  
1829 probability 0.25. This makes the oversight indicator a spurious feature on non-terminal squares; we  
1830 chose this setup so that an agent “by default” would learn to ignore the labels, absent reinforcement  
1831 for discriminating based on them. The environment state (which is observed by the agent) comprises  
1832 a one-hot encoded state of the grid cells (not pixels) and a binary mask that contains the cells’  
1833 oversight indicators.

1834 Comprehensive labels are available only for episodes where the agent reached a terminal square  
1835 with the indicator set to TRUE. For the remaining episodes, the labels are cursory, i.e. either “not  
reached” or “reached something”.

**Policy network architecture.** Our policy network  $\pi(s)$  incorporates a mixture of experts (MoE) layer. For a state  $s \in \mathcal{S}$ ,

$$\pi(s) = s \triangleright \text{MoE} \triangleright \text{Conv} \triangleright \text{Conv} \triangleright \text{Linear} \triangleright \text{Linear} \triangleright \text{Softmax},$$

with ReLU activations before any linear or convolutional layer.

where  $\triangleright$  denotes a piping operator,  $(x \triangleright f) \triangleq f(x)$ . The MoE layer combines outputs from two simple modules  $E_{\text{DIAMOND}}, E_{\text{GHOST}}$ , using a gating circuit  $\Gamma : \mathcal{S} \rightarrow [0, 1]^2$ :

$$\begin{aligned} \text{MoE}(s) &= E_{\text{DIAMOND}}(s) \cdot \Gamma(s)_{\text{DIAMOND}} + E_{\text{GHOST}}(s) \cdot \Gamma(s)_{\text{GHOST}}; \\ E_{\text{DIAMOND}}(s) &= s \triangleright \text{Conv} \triangleright \text{MaxPool}; \\ E_{\text{GHOST}}(s) &= s \triangleright \text{Conv} \triangleright \text{MaxPool}. \end{aligned}$$

This architecture allows us to isolate computations responsible for certain behaviors into the modules, and later steer the model by manually manipulating the gating coefficients. Despite the simplicity of our environment, we use convolutional layers to simulate the learning dynamics of more complex settings. This choice is held consistent across baselines to enable direct comparisons.

**Training.** We train the policy network using Proximal Policy Optimization (PPO) (Schulman et al., 2017) with a reward of +1 whenever it reaches any of the two terminal squares, i.e. the training reward for an episode with label  $y$  is  $r(y) = \mathbb{1}\{y \neq \text{not reached}\}$ .

Crucially, whenever we have access to a comprehensive label for an episode, we use it to perform gradient routing in the MoE layer. For example, for  $y = \text{reached DIAMOND}$ , we use

$$\text{MoE}(s) = E_{\text{DIAMOND}}(s) \cdot \Gamma(s)_{\text{DIAMOND}} + \text{sg}(E_{\text{GHOST}}(s)) \cdot \Gamma(s)_{\text{GHOST}},$$

where  $\text{sg}(\cdot)$  is the stop-gradient operator. We apply the inverse for  $y = \text{reached GHOST}$ , stopping the gradients for  $E_{\text{DIAMOND}}$ . We do not use any stop-gradients when oversight is unavailable (i.e. when we only have cursory labels). We only route the gradients in the MoE layer. In all other, non-MoE layers, gradients are propagated without routing in all episodes.

We include an additional loss term to encourage the gating circuit to activate only on one module. We only use the additional loss term when we have access to the comprehensive label for an episode. For a trajectory  $\tau = (s_1, a_1, \dots, s_T, r(y))$ , the overall loss is

$$\mathcal{L} = \mathcal{L}_{\text{PPO}}(\tau) - \alpha \cdot \mathbb{1}\{y \in \mathcal{Y}^*\} \cdot T^{-1} \sum_{t=1}^T \log \Gamma_y(s_t),$$

where  $\mathcal{L}_{\text{PPO}}(\tau)$  is the PPO loss,  $\alpha > 0$  is a tuning parameter, and we abuse notation by writing  $\Gamma_y$  to mean  $\Gamma_{\text{DIAMOND}}$  if  $y = \text{reached DIAMOND}$ , and similarly for GHOST.

**Hyperparameters.** We train the model using the CleanRL (Huang et al., 2022) implementation of PPO for 1.5M timesteps on environments with 4 parallel instances. Each policy rollout consisted of 256 steps. The learning rate was set to 2.5e-4 with annealing enabled. We used a discount factor ( $\gamma$ ) of 0.97 and GAE- $\lambda$  of 0.95. The model was updated for 4 epochs per rollout, using 4 minibatches. Advantages were normalized, and we applied a clipping coefficient of 0.2 for both policy and value function losses. The entropy coefficient was set to 0.01, and the value function coefficient to 0.5. Gradient norms were clipped at 0.5, and we did not use a target KL divergence threshold.

The custom loss term coefficient is  $\alpha = 0.3$ .

**Evaluation.** To steer the model towards DIAMOND, we assign gate coefficients  $\Gamma(\cdot)_{\text{DIAMOND}} = 1$  and  $\Gamma(\cdot)_{\text{GHOST}} = 0$ .

**Outcome-based baseline.** This agent trains on all episodes, with the reward function

$$r(y) = \begin{cases} -1 & \text{if } y = \text{reached GHOST}; \\ 1 & \text{if } y \in \{\text{reached something, reached DIAMOND}\}; \\ 0 & \text{otherwise.} \end{cases}$$

The outcome-based baseline never reaches DIAMOND in our test environments (as we verified empirically by testing it at 50% oversight), because it learns the optimal behavior under this reward function, which is to go to GHOST whenever it is closer than DIAMOND and not under oversight.

**Data-filtering baseline.** This agent trains with access to comprehensive labels, but only a fraction of the episodes are observed. This fraction is analogous to the oversight level  $p$ . Its reward function is

$$r(y) = \begin{cases} -1 & \text{if } y = \text{reached GHOST}; \\ 1 & \text{if } y = \text{reached DIAMOND}; \\ 0 & \text{otherwise.} \end{cases}$$

The data filtering baseline learns the optimal behavior given this reward function, which is to navigate to DIAMOND in all cases. However, it requires more data. To learn behavior comparable to a routed policy trained with 4% oversight, the filtering baseline requires about 20% of the training episodes to be kept (fig. 11). This means our method is about  $5\times$  more data-efficient than filtering.

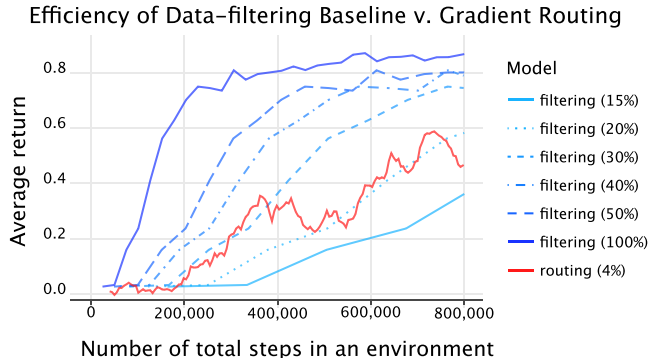


Figure 11: The data-efficiency of gradient routing compared to the data-filtering baseline. Dashed blue lines denote the filtering policy. We obtain these curves by training a policy with full oversight, then rescaling the  $x$ -axis to reflect the curve that would occur had varying proportions of data been filtered (proportions given in parentheses). The solid red line denotes a routed policy. We averaged five full-oversight filtering policies to obtain the displayed curves. The average return is measured on a 100 randomly generated environments.

Note that the highest episodic return for the gradient routing MoE agent is about 0.2 lower than the data filtering baseline. This is because, while the ablated agent virtually never navigates *towards* GHOST alone, it will walk to GHOST if it is on the shortest path to DIAMOND. Based on the random initial state of the environment, this happens some proportion of the time, leading to reduced reward. We discuss considerations necessary for overcoming this shortcoming in appendix G.

## G IMPACTS OF LOCALIZING CAPABILITIES VS. DISPOSITIONS FOR SCALABLE OVERSIGHT

To achieve scalable oversight, our proposed strategy for preventing bad behavior (for example) is to (1) localize a submodule responsible for bad behavior, then (2) ablate the submodule. In this section, we one factor that may complicate this strategy in real-world applications.

We distinguish between two types of processing that might occur within a neural network to cause some behavior, like navigating to a red tile in a gridworld. With respect to a particular behavior, we define:

**Capability.** Processing that is necessary for engaging in the behavior; for example, feature extraction and computation to detect a red tile and compute the shortest path to reach it.

**Disposition.** Processing that is not a capability but that determines behavior (as a probability distribution over network outputs). For example, a submodule that processes features representing the shortest path to a red tile and a blue tile and then returns action probabilities corresponding to the red tile.

These definitions are informal. *Note: Similar terms have been used in the context of AI evaluations (Beverley et al., 2024), but, to the best of our knowledge, have not been formalized. See Beverley et al. (2024) for a philosophical treatment of related terms.*

Depending on whether capabilities or dispositions are to be localized, the application of gradient routing to scalable oversight faces different challenges, as summarized in table 4.

Table 4: An overview of the challenges to localizing capabilities vs. dispositions as a means of achieving scalable oversight. A checkmark (✓) indicates a step that we speculate is easy to achieve; a challenge indicates a fundamental difficulty.

	Localization during training	After ablating the target region
Localizing capabilities	Challenge: entangled capabilities	✓
Localizing dispositions	✓	Challenge: distribution shift

In the case of capabilities localization, obtaining a performant policy post-ablation is straightforward in principle: by localizing and ablating, one has created an encoding of the state which does not admit any postprocessing which will exhibit the capability (analogous to the MNIST split encoding, whose bottom half did not admit any learned decoding for digits 0-4 as shown in fig. 2). In that case, one can simply train freeze this feature encoder and train on top of it. However, there is a fundamental challenge: in many problems, capabilities may not factor because they are entangled. For example, the skills required to be a cybersecurity researcher vs. a hacker overlap significantly.

On the other hand, we speculate that localizing dispositions is straightforward, and suitable for problems where capabilities are entangled. For example, even if cybersecurity and hacking involve the same capabilities, we expect to be able to localize the disposition for (harmful) hacking. However, localizing dispositions for scalable oversight does not permit post-ablation training, because further training could change the agent’s disposition. Instead, we must either zero-shot ablate, or find another manner of post-training that avoids this issue (e.g. fine-tuning on high-quality labeled data only). The fundamental difficulty to zero-shot ablation is distribution shift: suppose that during the training of a policy, an internal module is learned that governs the policy outputs in some regions of state space but not others. If, upon ablation, that module “becomes responsible” for regions that were previously governed by an ablated component, there is no reason to expect it to perform well in these states which are, with respect to its role in training, off-distribution.

## H COMPUTATIONAL COST OF GRADIENT ROUTING

**Memory.** Storing edge weights for every data point would incur a hefty cost of  $O(|\mathcal{B}||\mathcal{E}|)$  memory per batch. In practice, this cost is easily avoided by reducing dependence on the amount of data and the number of edges. First: instead of assigning unique gradient routes to each data point, we assign routes according to membership in parts of a partition  $\mathcal{P}$  of data points, reducing the  $|\mathcal{B}|$  term to  $|\mathcal{P}|$ . For example, in a typical unlearning application, we would use  $\mathcal{P} = \{\mathcal{P}_{\text{retain}}, \mathcal{P}_{\text{forget}}\}$  with a single gradient route assigned to each set. Second: we restrict the set of edges considered. For example, using only edges leaving parameters reduces the  $|\mathcal{E}|$  factor to  $O(p)$  if the neural net parameters have dimensionality  $p$ . This amounts to choosing elementwise learning rates for each parameter entry, for each data point.

**Runtime.** In the general case, gradient routing requires  $|\mathcal{B}||\mathcal{E}|$  floating point operations to apply a scalar multiplication to each edge in the computational graph. Since we apply gradient routing to a sparse set of edges, like the  $d_{\text{model}}$  entries of a hidden activation of a Transformer, the number of operations is much lower:  $|\mathcal{B}| \cdot d_{\text{model}}$ , for example. This is negligible compared to the number of operations required for matrix multiplication.

## I EXTENDED LITERATURE REVIEW

We start by reviewing further works that, like gradient routing, modify learning rates or backpropagation.

**Adjusting learning rates.** Discriminative fine-tuning (Howard & Ruder, 2018) sets the learning rate for each layer independently to improve training efficiency. You et al. (2017) introduce Layer-wise Adaptive Rate Scaling (LARS), which dynamically adjusts learning rates for each layer during training.

1998 **Modifying backpropagation.** Sun et al. (2017b)’s meProp uses only the top-k dimensions by mag-  
1999 nitude of the gradient when updating parameters during training, which improves the accuracy of  
2000 MNIST classifiers. Panda et al. (2024b) and Sung et al. (2021) optimize only a sparse subnetwork  
2001 of a model during fine-tuning, minimizing catastrophic forgetting and memory usage. Rosenfeld &  
2002 Tsotsos (2019) go a step further by updating only a small subset of parameters during pre-training,  
2003 demonstrating competitive performance compared to conventional methods.

2004 The methods above can be framed as multiplying the gradient by a mask vector. Mohtashami et al.  
2005 (2022) prove the theoretical convergence properties of binary gradient masking methods using a  
2006 similar notation to our definition of gradient routing in Section 3.

2007 Geiger et al. (2022b) train models to respect certain causal structure by applying interventions to  
2008 the forward pass and minimizing the difference between the actual output and the expected output  
2009 according to a user-supplied causal model. This method could be used to localize capabilities by  
2010 ensuring some modules are causally relevant to certain outputs.

2011 **Fine-tuning parameter subsets.** Many popular fine-tuning methods update only a small subset  
2012 of parameters with the goal of computational efficiency or minimizing catastrophic forgetting or  
2013 catastrophic interference (Sun et al., 2017a; Sung et al., 2021; Rosenfeld & Tsotsos, 2018; Kaplun  
2014 et al., 2024; Lee et al., 2023; Zhang et al., 2022; Mallya & Lazebnik, 2018; Panda et al., 2024a).  
2015 In some sense this localizes the new capabilities to this small subset of the network (as gradient  
2016 routing does), although these tuned parameters may be activating latent abilities already present in  
2017 the network (Ben Zaken et al., 2022).

2018 Safe LoRA (Hsu et al., 2024) projects fine-tuned weights into a “safety-aligned subspace”, while  
2019 subspace-oriented model fusion (SOMF) (Yi et al., 2024) masks task vectors (Ilharco et al., 2023)  
2020 such that they do not interfere with the subspace identified as relevant for safe behavior, before  
2021 merging them into the model using model fusion (Zhang et al., 2023; Jin et al., 2023).

2022 **Hierarchical reinforcement learning.** Early work in hierarchical reinforcement learning used hand  
2023 designed sub-behaviors assigned to individual modules to divide and conquer more complex tasks  
2024 (Maes & Brooks, 1990; Singh, 1992; Mahadevan & Connell, 1992) although later works generally  
2025 discarded this approach in favor of automatically learned sub-behaviors (Hutsebaut-Buysse et al.,  
2026 2022).

2027 **Disentangled representations.** While gradient routing partitions representations using supervised  
2028 training, disentangled representation learning attempts to separate representations in an unsuper-  
2029 vised manner (Bengio et al., 2013; Wang et al., 2024) using methods such as VAEs (Kingma &  
2030 Welling, 2013; Mathieu et al., 2019) and GANs (Goodfellow et al., 2014; Chen et al., 2016).  
2031  
2032  
2033  
2034  
2035  
2036  
2037  
2038  
2039  
2040  
2041  
2042  
2043  
2044  
2045  
2046  
2047  
2048  
2049  
2050  
2051