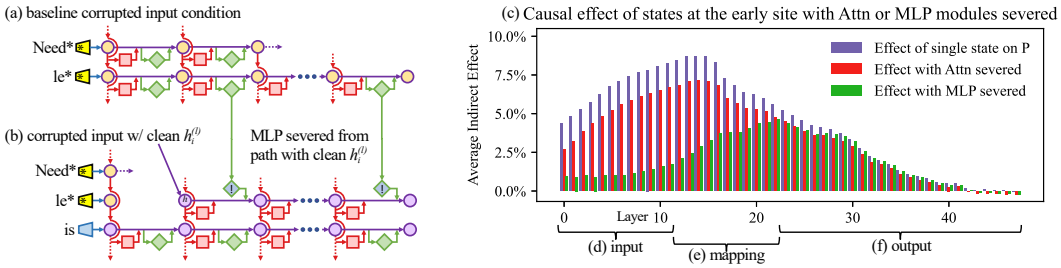

Figure 3: **Causal effects with a modified computation graph**. (a,b) To isolate the effects of MLP modules when measuring causal effects, the computation graph is modified. (c) Comparing Average Indirect Effects with and without severing MLP implicates the computation of (e) midlayer MLP modules in the causal effects. No similar gap is seen when attention is similarly severed.

2.2 Causal Tracing Results

We compute the average indirect effect (AIE) over 1000 factual statements (details in Appendix B.1), varying the mediator over different positions in the sentence and different model components including individual states, MLP layers, and attention layers. Figure 2 plots the AIE of the internal components of GPT-2 XL (1.5B parameters). The ATE of this experiment is 18.6%, and we note that a large portion of the effect is mediated by strongly causal individual states (AIE=8.7% at layer 15) at the last subject token. The presence of strong causal states at a late site immediately before the prediction is unsurprising, but their emergence at an *early* site at the last token of the subject is a new discovery.

Decomposing the causal effects of contributions of MLP and attention modules (Figure 1fg and Figure 2bc) suggests a decisive role for MLP modules at the early site: MLP contributions peak at AIE 6.6%, while attention at the last subject token is only AIE 1.6%; attention is more important at the last token of the prompt. Appendix B.2 further discusses this decomposition.

Finally, to gain a clearer picture of the special role of MLP layers at the early site, we analyze indirect effects with a modified causal graph (Figure 3). (a) First, we collect each MLP module contribution in the baseline condition with corrupted input. (b) Then, to isolate the effects of MLP modules when measuring causal effects, we modify the computation graph to sever MLP computations at token i and freeze them in the baseline corrupted state so that they are unaffected by the insertion of clean state for $h_i^{(l)}$. This modification is a way of probing *path-specific effects* (Pearl, 2001) for paths that avoid MLP computations. (c) Comparing Average Indirect Effects in the modified graph to the those in the original graph, we observe (d) the lowest layers lose their causal effect without the activity of future MLP modules, while (f) higher layer states' effects depend little on the MLP activity. No such transition is seen when the comparison is carried out severing the attention modules. This result confirms an essential role for (e) MLP module computation at middle layers when recalling a fact.

Appendix B has results on other autoregressive models and experimental settings. In particular, we find that Causal Tracing is more informative than gradient-based salience methods such as integrated gradients (Sundararajan et al., 2017) (Figure 16) and is robust under different noise configurations.

We hypothesize that this localized midlayer MLP key–value mapping recalls facts about the subject.

2.3 The Localized Factual Association Hypothesis

Based on causal traces, we posit a specific mechanism for storage of factual associations: each midlayer MLP module accepts inputs that encode a subject, then produces outputs that recall memorized properties about that subject. Middle layer MLP outputs accumulate information, then the summed information is copied to the last token by attention at high layers.

This hypothesis localizes factual association along three dimensions, placing it (i) in the MLP modules (ii) at specific middle layers (iii) and specifically at the processing of the subject's last token. It is consistent with the Geva et al. (2021) view that MLP layers store knowledge, and the Elhage et al. (2021) study showing an information-copying role for self-attention. Furthermore, informed by the Zhao et al. (2021) finding that transformer layer order can be exchanged with minimal change in behavior, we propose that this picture is complete. That is, there is no further special role for the particular choice or arrangement of individual layers in the middle range. We conjecture that any fact

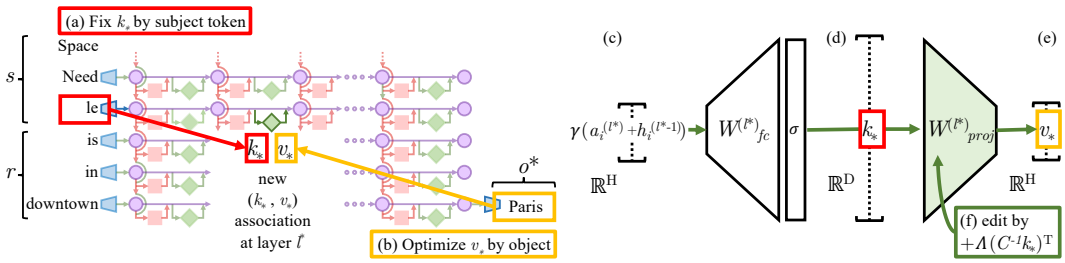

Figure 4: **Editing one MLP layer with ROME**. To associate *Space Needle* with *Paris*, the ROME method inserts a new (k_*, v_*) association into layer l^*, where (a) key k_* is determined by the subject and (b) value v_* is optimized to select the object. (c) Hidden state at layer l^* and token i is expanded to produce (d) the key vector k_* for the subject. (e) To write new value vector v_* into the layer, (f) we calculate a rank-one update $\Lambda(C^{-1}k_*)^T$ to cause $\hat{W}^{(l)}_{proj}k_* = v_*$ while minimizing interference with other memories stored in the layer.

could be equivalently stored in any one of the middle MLP layers. To test our hypothesis, we narrow our attention to a single MLP module at a mid-range layer l^*, and ask whether its weights can be explicitly modified to store an arbitrary fact.

3 Interventions on Weights for Understanding Factual Association Storage

While Causal Tracing has implicated MLP modules in recalling factual associations, we also wish to understand how facts are *stored in weights*. Geva et al. (2021) observed that MLP layers (Figure 4cde) can act as two-layer key–value memories,[6] where the neurons of the first layer $W^{(l)}_{fc}$ form a *key*, with which the second layer $W^{(l)}_{proj}$ retrieves an associated *value*. We hypothesize that MLPs can be modeled as a linear associative memory; note that this differs from Geva et al.'s per-neuron view.

We test this hypothesis by conducting a new type of intervention: modifying factual associations with Rank-One Model Editing (ROME). Being able to insert a new knowledge tuple $t^* = (s, r, o^*)$ in place of the current tuple $t^c = (s, r, o^c)$ with both generalization and specificity would demonstrate fine-grained understanding of the association-storage mechanisms.

3.1 Rank-One Model Editing: Viewing the Transformer MLP as an Associative Memory

We view $W^{(l)}_{proj}$ as a linear associative memory (Kohonen, 1972; Anderson, 1972). This perspective observes that any linear operation W can operate as a key–value store for a set of vector keys $K = [k_1 \mid k_2 \mid \dots]$ and corresponding vector values $V = [v_1 \mid v_2 \mid \dots]$, by solving $WK \approx V$, whose squared error is minimized using the Moore-Penrose pseudoinverse: $W = VK^+$. Bau et al. (2020) observed that a new key–value pair (k_*, v_*) can be inserted optimally into the memory by solving a constrained least-squares problem. In a convolutional network, Bau et al. solve this using an optimization, but in a fully-connected layer, we can derive a closed form solution:

$$\text{minimize } \|\hat{W}K - V\| \text{ such that } \hat{W}k_* = v_* \quad \text{by setting } \hat{W} = W + \Lambda(C^{-1}k_*)^T. \tag{2}$$

Here W is the original matrix, $C = KK^T$ is a constant that we pre-cache by estimating the uncentered covariance of k from a sample of Wikipedia text (Appendix E.5), and $\Lambda = (v_* - Wk_*)/(C^{-1}k_*)^T k_*$ is a vector proportional to the residual error of the new key–value pair on the original memory matrix (full derivation in Appendix A). Because of this simple algebraic structure, we can insert any fact directly once (k_*, v_*) is computed. All that remains is to choose the appropriate k_* and v_*.

Step 1: Choosing k_* to Select the Subject. Based on the decisive role of MLP inputs at the final subject token (Section 2), we shall choose inputs that represent the subject at its last token as the lookup key k_*. Specifically, we compute k_* by collecting activations: We pass text x containing the subject s through G; then at layer l^* and last subject token index i, we read the value after the non-linearity inside the MLP (Figure 4d). Because the state will vary depending on tokens that

[6] Unrelated to keys and values in self-attention.

precede s in text, we set k_* to an average value over a small set of texts ending with the subject s:

$$k_* = \frac{1}{N}\sum_{j=1}^{N} k(x_j + s), \text{ where } k(x) = \sigma\left(W_{fc}^{(l^*)}\,\gamma(a_{[x],i}^{(l^*)} + h_{[x],i}^{(l^*-1)})\right). \tag{3}$$

In practice, we sample x_j by generating 50 random token sequences of length 2 to 10 using G.

Step 2: Choosing v_* to Recall the Fact. Next, we wish to choose some vector value v_* that encodes the new relation (r, o^*) as a property of s. We set $v_* = \operatorname{argmin}_z \mathcal{L}(z)$, where the objective $\mathcal{L}(z)$ is:

$$\frac{1}{N}\sum_{j=1}^{N} \underbrace{-\log \mathbb{P}_{G(m_i^{(l^*)}:=z)}\left[o^* \mid x_j + p\right]}_{\text{(a) Maximizing } o^* \text{ probability}} + \underbrace{D_{\mathrm{KL}}\left(\mathbb{P}_{G(m_{i'}^{(l^*)}:=z)}\left[x \mid p'\right] \middle\| \mathbb{P}_G\left[x \mid p'\right]\right)}_{\text{(b) Controlling essence drift}}. \tag{4}$$

The first term (Eqn. 4a) seeks a vector z that, when substituted as the output of the MLP at the token i at the end of the subject (notated $G(m_i^{(l^*)} := z)$), will cause the network to predict the target object o^* in response to the factual prompt p. The second term (Eqn. 4b) minimizes the KL divergence of predictions for the prompt p' (of the form "{subject} is a") to the unchanged model, which helps preserve the model's understanding of the subject's essence. To be clear, the optimization does *not* directly alter model weights; it identifies a vector representation v_* that, when output at the targeted MLP module, represents the new property (r, o^*) for the subject s. Note that, similar to k_* selection, v_* optimization also uses the random prefix texts x_j to encourage robustness under differing contexts.

Step 3: Inserting the Fact. Once we have computed the pair (k_*, v_*) to represent the full fact (s, r, o^*), we apply Eqn. 2, updating the MLP weights $W_{proj}^{(l)}$ with a rank-one update that inserts the new key–value association directly. For full implementation details, see Appendix E.5.

3.2 Evaluating ROME: Zero-Shot Relation Extraction (zsRE)

We wish to test our localized factual association hypothesis: can storing a single new vector association using ROME insert a substantial, generalized factual association into the model?

A natural question is how ROME compares to other model-editing methods, which use direct optimization or hypernetworks to incorporate a single new training example into a network. For baselines, we examine Fine-Tuning **(FT)**, which applies Adam with early stopping at one layer to minimize $-\log \mathbb{P}\left[o^* \mid x\right]$. Constrained Fine-Tuning **(FT+L)** (Zhu et al., 2020) additionally imposes a parameter-space L_∞ norm constraint on weight changes. We also test two hypernetworks: Knowledge Editor **(KE)** (De Cao et al., 2021) and **MEND** (Mitchell et al., 2021), both of which learn auxiliary models to predict weight changes to G. Further details are described in Appendix E.

Table 1: zsRE Editing Results on GPT-2 XL.

Editor	Efficacy ↑	Paraphrase ↑	Specificity ↑
GPT-2 XL	22.2 (±0.5)	21.3 (±0.5)	24.2 (±0.5)
FT	99.6 (±0.1)	82.1 (±0.6)	23.2 (±0.5)
FT+L	92.3 (±0.4)	**47.2 (±0.7)**	23.4 (±0.5)
KE	65.5 (±0.6)	61.4 (±0.6)	24.9 (±0.5)
KE-zsRE	92.4 (±0.3)	90.0 (±0.3)	23.8 (±0.5)
MEND	75.9 (±0.5)	65.3 (±0.6)	24.1 (±0.5)
MEND-zsRE	99.4 (±0.1)	**99.3 (±0.1)**	24.1 (±0.5)
ROME	**99.8 (±0.0)**	88.1 (±0.5)	**24.2 (±0.5)**

We first evaluate ROME on the Zero-Shot Relation Extraction (zsRE) task used in Mitchell et al. (2021) and De Cao et al. (2021). Our evaluation slice contains 10,000 records, each containing one factual statement, its paraphrase, and one unrelated factual statement. "Efficacy" and "Paraphrase" measure post-edit accuracy $\mathbb{I}\left[o^* = \operatorname{argmax}_o \mathbb{P}_{G'}\left[o\right]\right]$ of the statement and its paraphrase, respectively, while "Specificity" measures the edited model's accuracy on an unrelated fact. Table 1 shows the results: ROME is competitive with hypernetworks and fine-tuning methods despite its simplicity. We find that it is not hard for ROME to insert an association that can be regurgitated by the model. Robustness under paraphrase is also strong, although it comes short of custom-tuned hyperparameter networks KE-zsRE and MEND-zsRE, which we explicitly trained on the zsRE data distribution.[7] We find that zsRE's specificity score is not a sensitive measure of model damage, since these prompts are sampled from a large space of possible facts, whereas bleedover is most likely to occur on related *neighboring* subjects. Appendix C has additional experimental details.

[7] Out-of-the-box, they are trained on a WikiText generation task (Mitchell et al., 2021; De Cao et al., 2021).