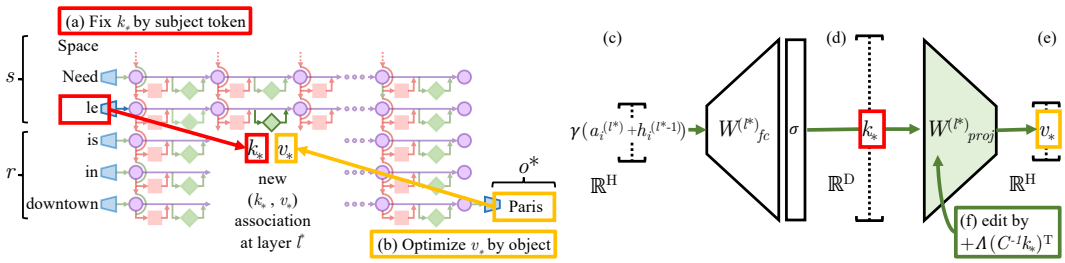

Figure 4: **Editing one MLP layer with ROME**. To associate *Space Needle* with *Paris*, the ROME method inserts a new (k_*, v_*) association into layer l^*, where (a) key k_* is determined by the subject and (b) value v_* is optimized to select the object. (c) Hidden state at layer l^* and token i is 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.