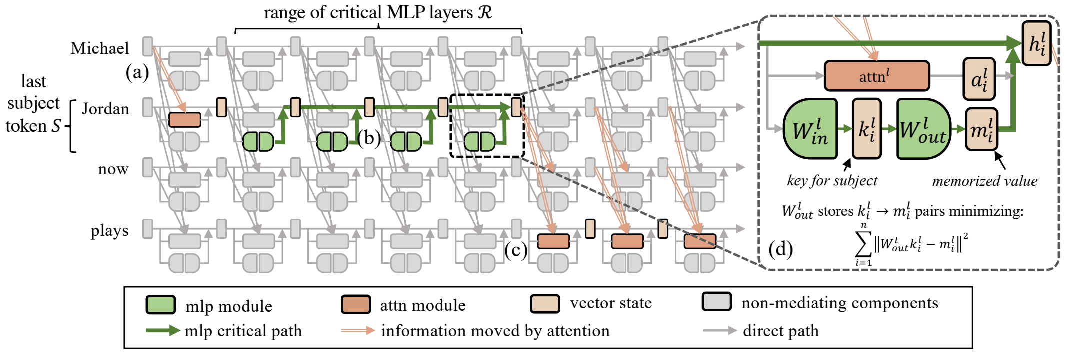

Figure 2: **MEMIT modifies transformer parameters on the critical path of MLP-mediated factual recall.** We edit stored associations based on observed patterns of causal mediation: (a) first, the early-layer attention modules gather subject names into vector representations at the last subject token S. (b) Then MLPs at layers $l \in \mathcal{R}$ read these encodings and add memories to the residual stream. (c) Those hidden states are read by attention to produce the output. (d) MEMIT edits memories by storing vector associations in the critical MLPs.

confirm that GPT-J has a concentration of mediating states h_i^l; moreover, they highlight a mediating causal role for a range of MLP modules, which can be seen as a large gap between the effect of single states (purple bars in Figure 3) and the effects with MLP severed (green bars); this gap diminishes after layer 8. Unlike Meng et al. (2022) who use this test to identify a single edit layer, we select the whole range of critical MLP layers $l \in \mathcal{R}$. For GPT-J, we have $\mathcal{R} = \{3, 4, 5, 6, 7, 8\}$.

Given that a *range* of MLPs play a joint mediating role in recalling facts, we ask: what is the role of *one* MLP in storing a memory? Each token state in a transformer is part of the residual stream that all attention and MLP modules read from and write to (Elhage et al., 2021). Unrolling Eqn. 2 for $h_i^L = h_{[S]}^L(p_i)$:

$$h_i^L = h_i^0 + \sum_{l=1}^{L} a_i^l + \sum_{l=1}^{L} m_i^l. \tag{6}$$

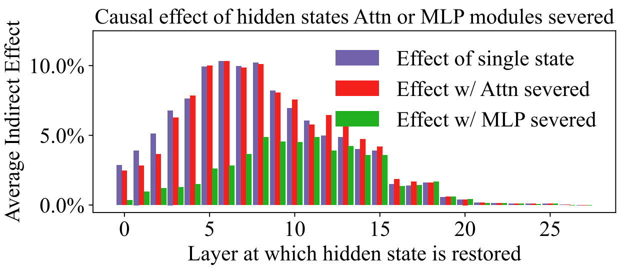

Figure 3: A critical mediating role for mid-layer MLPs.

Eqn. 6 highlights that each individual MLP contributes by *adding* to the memory at h_i^L (Figure 2b), which is later read by last-token attention modules (Figure 2c). Therefore, when writing new memories into G, we can spread the desired changes across all the critical layers m_i^l for $l \in \mathcal{R}$.

4.2 Batch update for a single linear associative memory

In each individual layer l, we wish to store a large batch of $u \gg 1$ memories. This section derives an optimal single-layer update that minimizes the squared error of memorized associations, assuming that the layer contains previously-stored memories that should be preserved. We denote $W_0 \triangleq W_{out}^l$ (Eqn. 4, Figure 2) and analyze it as a linear associative memory (Kohonen, 1972; Anderson, 1972) that associates a set of input keys $k_i \triangleq k_i^l$ (encoding subjects) to corresponding memory values $m_i \triangleq m_i^l$ (encoding memorized properties) with minimal squared error:

$$W_0 \triangleq \underset{\hat{W}}{\operatorname{argmin}} \sum_{i=1}^{n} \left\| \hat{W} k_i - m_i \right\|^2. \tag{7}$$

If we stack keys and memories as matrices $K_0 = [k_1 \mid k_2 \mid \cdots \mid k_n]$ and $M_0 = [m_1 \mid m_2 \mid \cdots \mid m_n]$, then Eqn. 7 can be optimized by solving the normal equation (Strang, 1993, Chapter 4):

$$W_0 K_0 K_0^T = M_0 K_0^T. \tag{8}$$

Suppose that pre-training sets a transformer MLP's weights to the optimal solution W_0 as defined in Eqn. 8. Our goal is to update W_0 with some small change Δ that produces a new matrix W_1 with