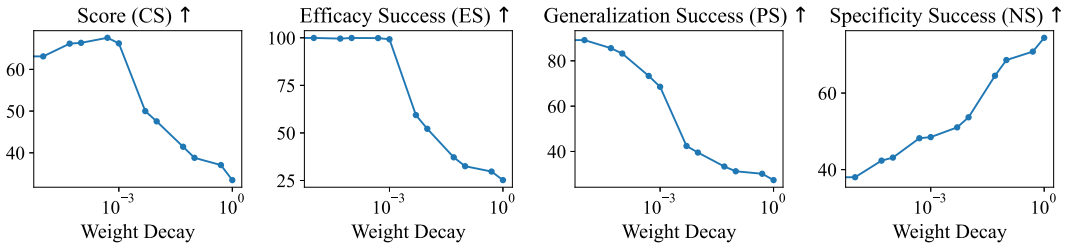

Figure 9: **Optimizing fine-tuning weight decay on 10,000 edits**. We find an evident tradeoff between generalization and specificity, opting for the value with the highest Score.

Note that we choose not to complicate the analysis by tuning FT-W on more than one layer. Table 2 demonstrates that FT-W, with just one layer, already gets near-perfect efficacy at the cost of low specificity, which indicates sufficient edit capacity.

B.2 Model Editing Networks with Gradient Decomposition (MEND)

MEND makes concurrent edits by accumulating gradients from all edit examples, then passing them through the hypernetwork together. We use the GPT-J MEND hypernetwork trained by Meng et al. (2022). During inference, learning rate scale is set to the default value of 1.0. MEND is by far the fastest method, taking 98.25 seconds to execute $10{,}000$ updates on GPT-J.

B.3 Rank-One Model Editing (ROME)

The default ROME hyperparameters are available in their open source code: GPT-J updates are executed at layer 5, where optimization proceeds for 20 steps with a weight decay of 0.5, KL factor of 0.0625, and learning rate of 5×10^{-1}. ROME uses prefix sampling, resulting in 10 prefixes of length 5 and 10 prefixes of length 10. Covariance statistics are collected in `fp32` on Wikitext using a sample size of 100,000. See Meng et al. (2022) for more details.

ROME takes $44{,}248.26\,\text{sec} \approx 12.29\,\text{hr}$ for $10{,}000$ edits on GPT-J, which works out to approximately 4 seconds per edit.

B.4 Mass-Editing Memory in a Transformer (MEMIT)

On GPT-J, we choose $\mathcal{R} = \{3, 4, 5, 6, 7, 8\}$ and set λ, the covariance adjustment factor, to $15{,}000$. Similar to ROME, covariance statistics are collected using 100,000 samples of Wikitext in `fp32`. δ_i optimization proceeds for 25 steps with a learning rate of 5×10^{-1}. In practice, we clamp the L_2 norm of δ_i such that it is less than $\frac{3}{4}$ of the original hidden state norm, $\|h_i^L\|$. On GPT-NeoX, we select $\mathcal{R} = \{6, 7, 8, 9, 10\}$ and set $\lambda = 20{,}000$. Covariance statistics are collected over 50,000 samples of Wikitext in `fp16` but stored in `fp32`. Optimization for δ_i proceeds for 20 steps using a learning rate of 5×10^{-1} while clamping $\|h_i^L\|$ to $\frac{3}{10}\|h_i^L\|$.

In MEMIT, we have the luxury of being able to pre-compute and cache z_i values, since they are inserted in parallel. If all such vectors are already computed, MEMIT takes $3{,}226.35\,\text{sec} \approx 0.90\,\text{hr}$ for $10{,}000$ updates on GPT-J, where the most computationally expensive step is inverting a large square matrix (Eqn. 14). Computing each z_i vector is slightly less expensive than computing a ROME update; to get all 10,000 z_i vectors, we need $23{,}546.65\,\text{sec} \approx 6.54\,\text{hr}$. This optimization is currently done in series, but it is actually "embarrassingly parallel," as we can greatly reduce computation time by batching the gradient descent steps. Note that this speed-up does not apply to ROME, since each update must be done iteratively.