PROTOT V2: A SIMPLER, MORE STABLE, AND EDITABLE PROTOTYPE TRANSFORMER

Anonymous authors
Paper under double-blind review

ABSTRACT

While state-of-the-art large language models dominate benchmarks and real-world performance, their reasoning has remained hard to trust and verify. This has inspired a new generation of language models based on interpretable components, but such models often suffer from poor scalability, efficiency, and performance. In this work, we significantly simplify and stabilize the recently published Prototype Transformer (ProtoT), leading to substantial improvements across all three of these dimensions. We also develop a custom Triton kernel for the simplified formulation that increases the speed by about $3\times$, substantially surpassing all baselines. Finally, we introduce a new synthetic-prototype method that enables more flexible and powerful model editing, on par with strong baselines, while showing that our model is particularly editable. Together, these advances yield ProtoT v2, a substantially more performant, efficient, and editable prototype-based language model that retains the interpretability and robustness properties of ProtoT.

1 INTRODUCTION

State-of-the-art large language models (LLMs), such as OpenAI's Astra and Anthropic's Fable families of models, have demonstrated exceptional performance across a wide range of applications and capabilities (OpenAI, 2026a; Anthropic, 2026), with frontier systems now exceeding some human-expert baselines (Artificial Analysis, 2026), and even solving the Navier–Stokes Millennium Prize Problem (OpenAI, 2026b; Clay Mathematics Institute, 2026). However, their reasoning remains difficult to trust and verify, with chain-of-thought (CoT) not always reflecting the internal reasoning of LMs (Turpin et al., 2023; Lanham et al., 2023; Chen et al., 2025). This problem may become increasingly important as frontier models rely less on explicitly verbalized reasoning and more on latent computation in hidden representations, including recent approaches based on recurrent depth, continuous latent reasoning, and so-called "neuralese" (Pachocki, 2026; Brown-Cohen et al., 2026; Sheffield et al., 2026). In practice, this increasingly requires additional interpretability methods operating directly on internal representations, including trainable external modules such as Sparse Auto-Encoders (SAEs) (Bricken et al., 2023; Templeton et al., 2024).

This has inspired a new generation of LMs that incorporate interpretability directly into their architecture, accepting architectural constraints or some loss in capability in exchange for greater inherent interpretability, including Weight-Sparse Transformer, ProtoT, and ParityTransformer (Gao et al., 2025; Yordanov et al., 2026; Mack et al., 2026). However, existing approaches have yet to demonstrate competitive language-modeling performance against strong modern baselines without a substantial efficiency trade-off.

In this work, we present ProtoT v2, a new Prototype Transformer model improving on ProtoT (Yordanov et al., 2026), by simplifying the architecture, delivering improved performance and speed, and greatly improving the editability properties of ProtoT (Yordanov et al., 2026). We also demonstrate our claims at a substantially larger scale: $10\times$ model size and $10\times$ data size; and run more comprehensive performance benchmarks that reveal the strengths and weaknesses of ProtoT v2. ProtoT v2 also shows broadly comparable robustness to the other models we evaluate when the text contains small wording changes or typing errors.

The main contributions of our work are as follows:

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

- Simplifying the ProtoT architecture in multiple ways, leading to computational and memory savings, and giving the prototypes a larger role in the cross-token information mixing.
- Significantly improving the numerical stability of the exponential time-discount mechanism by introducing a new computational reformulation for it, which leads to better performance at larger context sizes.
- Creating a custom Triton kernel that yields about 3 times faster training and inference speed, and 3 times less memory usage, versus PyTorch implementation with torch.compile.
- Developing a new prototype-based editability mechanism that uses synthetic prototypes for targeted edits with low collateral damage. This method enables much more flexible and powerful edits than the prototype intervention methods in ProtoT(Yordanov et al., 2026), and is on par with strong baselines for targeted editing. Furthermore, we show that ProtoT v2 has overall better editability profile than LLaMA-3 (Grattafiori et al., 2024).

2 Related Work

Post-hoc analysis of language models. Much of our understanding of LM internals comes from dissecting models never trained to be understood, typically by tracing an observed behavior back to the components that produced it (Zhang & Nanda, 2024). Attention offers little help here, as a token can receive heavy attention without influencing the output (Jain & Wallace, 2019), and networks routinely store several features along shared directions, a regime known as *superposition* (Elhage et al., 2022). Both issues are sidestepped through activation patching (Meng et al., 2022; Geva et al., 2023; Wang et al., 2022; Conmy et al., 2023), lenses that decode intermediate states into next-token distributions (Nostalgebraist, 2020; Belrose et al., 2023), or sparse autoencoders (SAEs) that recover more monosemantic features from MLP or attention activations (Bricken et al., 2023; Kissane et al., 2024; Rai et al., 2024; Paulo et al., 2025). All of these tools, however, explain a model after training through an auxiliary lens whose faithfulness must itself be validated.

Interpretability built into training. A few recent works avoid this detour by constraining the architecture itself. Gao et al. (2025) constrain most Transformer weights to zero and prune the result into small task circuits whose units align with natural concepts, at some cost in capability. Mack et al. (2026) uses a sparse bottleneck with a fixed, parameter-free dictionary before every MLP of a GPT-2-style model, matching or exceeding post-hoc SAEs on probing, steering, and causal edits. This comes at the price of several times more training tokens than a dense baseline. Both approaches constrain MLP computation or weight connectivity, whereas the ProtoT family of models places the interpretable unit in the token-mixing module itself.

Interpretability through prototypes. Prototype-based models explain an output by pointing to the stored examples it most resembles. Image classifiers built on this principle score each patch against a bank of part-level templates learned during training (Chen et al., 2019; Rymarczyk et al., 2022), a design that ProtoViT carries over to ViTs by letting those templates deform spatially at the output layer (Ma et al., 2024). Language counterparts vary in what counts as a prototype: whole training examples retrieved via attention in ProtoAttend (Arik & Pfister, 2020), evolving sequences of prototypes in ProtoryNet (Hong et al., 2023), and a small, sparsity-regularized prototype set in ProSeNet (Ming et al., 2019). Span-level prototypes (Wei & Zhu, 2025) and fully transparent sentiment classifiers (Wen & Rezapour, 2025) extend this family further. In nearly all of these models, the prototype layer is a final read-out attached to an otherwise standard encoder. ProtoT and ProtoT v2 reverse this arrangement: prototypes sit inside every block and determine how information flows between tokens during generation. Each ProtoT prototype defines an isolated communication channel, so the R channels run side by side at $O(R)$ cost with no cross-talk (Yordanov et al., 2026). ProtoT and ProtoT v2 advance each channel token by token with an exponential moving average (EMA) that only sees the past. Rather than a shared workspace for generic computation, the mixer therefore functions as a router whose narrow channels favor prototypes that correspond to nameable concepts (Sec. 5.3).

Editing language models. Editing aims to change a specific behaviour of a trained model at inference time, such as the sentiment or toxicity of its continuations, without retraining it and without degrading its general fluency. A common approach is activation steering, which adds a concept direction to the model's hidden states, for example the activation difference of a contrast pair (Turner

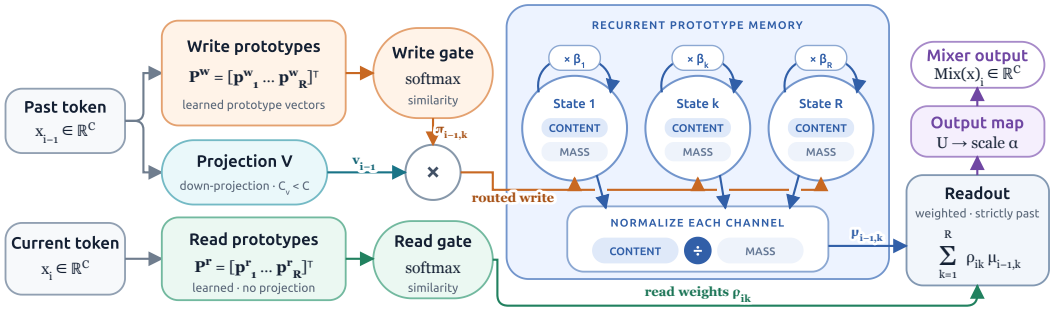

Figure 1: ProtoT v2 mixer. The write gate assigns each token value to recurrent memory channels. Each channel tracks discounted content and routing mass. The read gate combines the normalized states from preceding positions using an independent table of read prototypes.

et al., 2023), probe-derived directions on selected attention heads (Li et al., 2023), or directions read out with Linear Artificial Tomography (Zou et al., 2023). Representation finetuning instead learns the intervention end to end (Wu et al., 2024), and sparse autoencoder features have been used as steering handles at scale (Templeton et al., 2024). Wu et al. (2025) compare these approaches on a fixed backbone and find that simple recipes built from class-mean differences (Marks & Tegmark, 2023), principal components (Tigges et al., 2023), or linear probes (Alain & Bengio, 2017) are competitive, with difference-in-means outperforming steering with SAE features. These methods act on models that were not designed to be edited. The edit direction is found post hoc and, in most cases, added uniformly at every position, independently of what the token carries. Architectures that expose an editing interface by design address the first issue, but tie the interface to fixed units. Backpack language models (Hewitt et al., 2023) edit per-word sense vectors, while token mixing is still performed by a standard Transformer. Concept Bottleneck LLMs (Sun et al., 2025) intervene on neurons aligned with a predefined concept set.

3 Model

We propose the ProtoT v2 model (Figure 1), a new Prototype Transformer that improves the ProtoT (Yordanov et al., 2026) architecture in several key ways listed in the paragraphs below, with further details in Appendix A.3 regarding their performance, speed, and memory impacts. It uses a standard transformer backbone following LLaMA-3 (Grattafiori et al., 2024), apart from the mixer module which uses prototypes instead of self-attention. The full formula of the ProtoT v2 mixer is in Equations 1, 2, and 3, where $V : \mathbb{R}^C \to \mathbb{R}^{C_v}$ and $U : \mathbb{R}^{C_v} \to \mathbb{R}^C$ are learned bias-free linear maps, R is the number of communication channels, and $\mathbf{p}^{\mathrm{w}}, \mathbf{p}^{\mathrm{r}} \in \mathbb{R}^{R \times C}$ are the write- and read-gate prototypes, respectively:

$$\mathbf{v}_i = V\mathbf{x}_i, \quad \pi_{ik} = \mathrm{Softmax}_k\left(\frac{\mathbf{x}_i^{\mathsf{T}}\mathbf{p}_k^{\mathrm{w}}}{\tau_{\mathrm{w}}}\right), \quad \rho_{ik} = \mathrm{Softmax}_k\left(\frac{\mathbf{x}_i^{\mathsf{T}}\mathbf{p}_k^{\mathrm{r}}}{\tau_{\mathrm{r}}}\right). \tag{1}$$

$$\begin{aligned} \mathbf{s}_{0k} &= \mathbf{0}, & \mathbf{s}_{ik} &= \beta_k \mathbf{s}_{i-1,k} + (1-\beta_k)\,\pi_{ik}\mathbf{v}_i, \\ m_{0k} &= 0, & m_{ik} &= \beta_k m_{i-1,k} + (1-\beta_k)\,\pi_{ik}. \end{aligned} \tag{2}$$

$$\boldsymbol{\mu}_{ik} = \frac{\mathbf{s}_{ik}}{\max(m_{ik}, \varepsilon)}, \quad \mathrm{Mix}(\mathbf{x})_i = \alpha\, U\Big(\sum_{k=1}^{R} \underbrace{\rho_{ik}}_{\text{Read gate}} \underbrace{\boldsymbol{\mu}_{i-1,k}}_{\substack{\text{strictly-past} \\ \text{prototype memory}}}\Big). \tag{3}$$

Numerical Stability Improvement We notice numerical stability issues in ProtoT (Yordanov et al., 2026) caused by the exponential time discount in the prefix mean: the β^t term in $\beta^t \,\mathrm{cumsum}_t((1-\beta)x_t/\beta^t)$, which, for $\beta \leq 0.9$, underflows to zero in FP32 before $t = 1024$, which affects the model at context sizes 1024 and above. This is an inherent problem with this unrolled formulation of the time-discounted prefix mean, where β^t has to be materialized. Unlike ProtoT, ProtoT v2 breaks the computation down into a composition of recurrent operations:

$z \to \beta z + (1-\beta)x_t$ (Eq. 2). This removes the need for a large-power denominator or numerator, stabilizing the computation numerically. This leads to significant perplexity gains (Appendix A.3, Table 8), especially at larger context lengths (supported by the results in Section 5.1). Furthermore, it enables the computation at the Prefix Mean to be done at lower number precision, hence leads to further memory and compute savings, which are realized in the kernel (see *Triton Kernel* below).

Direct Read Routing ProtoT forms read scores by first applying a bias-free $C \times C$ projection W_{r} to each token, then taking dot products with the write prototypes. Its read logit is therefore $(W_{\mathrm{r}}\mathbf{x}_i)^{\mathsf{T}}\mathbf{p}_k^{\mathrm{w}} = \mathbf{x}_i^{\mathsf{T}}(W_{\mathrm{r}}^{\mathsf{T}}\mathbf{p}_k^{\mathrm{w}})$. ProtoT v2 learns the resulting $R \times C$ read-prototype table directly, as in Eq. 1. This removes a dense per-token projection and replaces its C^2 parameters with RC read-prototype parameters. The read table remains independent of the write table, so tokens can read from different channels than those to which they write. This yields similar performance, while simplifying the architecture, increasing the model speed, and reducing the memory footprint.

Removing Causal Convolution The ProtoT value path applies a causal depthwise convolution in its first two layers before writing to memory. ProtoT v2 omits this convolution, leaving the prototype mixer as the block's only mechanism for exchanging information between positions, thus making prototype interpretability a more reliable way to interpret the mixer module. Furthermore, this change yields small speed gain at little to no performance impact.

We also make three smaller changes to the architecture: we initialize the time-discounts into 4 groups instead of identically, removing Read-Write gate sharing at layer 0, and removing the softer initialization of the Read-Gate temperature τ_r. The ablation results in Appendix A.3 show that these three changes have little to no effect on speed and performance.

Triton Kernel We develop a custom Triton kernel for the mixer module based on our recurrent time-discount formulation (Eq. 2). The kernel implements this recurrence efficiently by processing the sequence in chunks, combining local within-chunk interactions with compact per-channel memory states propagated across chunk boundaries. Even though this kernel only optimises the mixer module, it yields about 2.7-2.8 times faster training and inference over ProtoT at the model level, by making the mixer many times faster. Furthermore, the kernel cuts memory usage by about three times versus ProtoT at the model level. These memory savings allow for larger minibatch size and thus result in effectively about 3 times faster training and inference than ProtoT. See Appendix A.3 Table 8 for full results, and Appendix A.1 for a more comprehensive description of the kernel.

4 Experimental Setup

Baselines This work uses substantially stronger and more recent linear-compute LM baselines than (Yordanov et al., 2026): both Mamba-3 (Lahoti et al., 2026) and GDN-2 (Hatamizadeh et al., 2026) are two major generations and two years ahead of the baselines used by Yordanov et al. (2026) (Mamba (Gu & Dao, 2024) and DeltaNet (Yang et al., 2024), respectively), whereas ProtoT v2 is less than a year ahead of ProtoT (Yordanov et al., 2026). Following Yordanov et al. (2026), we use as baseline LLaMA-3 (Grattafiori et al., 2024), which allows us to compare the ProtoT v2 mixer against full self-attention. The configurations used for each model and the full hyperparameter search are in Appendix A.4, Tables 9, 10, 11, 12, 13, and 14.

Dataset and Tokenizer We train all architectures on ClimbMix, a curated pre-training mixture designed for stronger performance under a fixed token budget (Diao et al., 2025). This differs from the FineWeb-Edu subset used by Yordanov et al. (2026) (Penedo et al., 2024). At the base scale, we use approximately 347M distinct training tokens; at the large scale, we construct a 4B-token corpus and reserve its final 10M tokens for validation. We make five passes over the respective training subsets (approximately 1.73B and 19.93B token presentations). We use five epochs rather than the ten-epoch schedule of Yordanov et al. (2026), allocating the training budget to larger models and data subsets. The reported data-efficiency benefits of ClimbMix motivate this choice (Diao et al., 2025); we do not independently evaluate the effect of changing datasets.

We use the standard GPT-2 byte-level BPE tokenizer, with its 50,257-token vocabulary, for every architecture (Radford et al., 2019). The released ClimbMix token sequences already use this tok-

enizer (Diao et al., 2025). In contrast, ProtoT trained a custom 16,000-token BPE tokenizer on its FineWeb-Edu sample (Yordanov et al., 2026).

Training Setup We generally follow the experimental setup by Yordanov et al. (2026). However, we use substantially larger model size, token number, and context length at both "base" and "large" experiment scales, which allows us to develop and test the architecture closer to production scale. Model size: 80M vs 8M parameters for the base scale, and 400M vs 40M parameters for the large scale. Number of tokens: 347M vs 12.5M tokens at base scale, and 4B vs 325M tokens at large scale. Context size: 1024 vs 256 at the base scale and 1024 vs 512 at the large scale.

Unlike ProtoT (Yordanov et al., 2026), we use approximately parameter-matched model sizes, instead of matching the hidden size across models, which allows for a fairer comparison. We choose the same number of layers for all models: 12 at base setup, and 18 at large setup. More details about this procedure are in Appendix A.2, with full model configuration details in Appendix A.4 Table 9. ProtoT v2 uses $R = 64$ channels, compared with $R = 32$ in Yordanov et al. (2026), to provide additional prototype capacity at larger scale. To compensate for the added compute, we decrease the hidden size by 25%, from 256 to 192 at the base scale. Overall, we observe that this is a net performance gain (Appendix A.4, Table 10).

We use the base scale for the base perplexity comparison, context scalability, hyperparameter search, and ablation experiments. For all further experiments, we use the large scale setup. We do not train and test ProtoT (Yordanov et al., 2026) at the large scale due to computational constraints (it is about 3 times slower). The ProtoT v2 kernel implements the exponentially discounted cumulative sum through a numerically stable, chunked affine formulation, making stabilization integral to its design. Therefore, it would be hard to obtain comparable kernel gains for ProtoT while preserving its original scaled-cumsum formulation, which is the source of numerical-stability issues.

We use AdamW optimizer (Loshchilov & Hutter, 2019), and select hyperparameters via grid search. Following Yordanov et al. (2026), we use linear warmup on the first 2% of training and cosine annealing towards 10% of the peak LR. For additional training details, see Appendix A.2.

5 Experiments

5.1 Performance and Scalability

Base-Scale Performance The results in Table 1 show that ProtoT v2 outperforms ProtoT at larger context sizes: the model has about 7.7% better perplexity at both ctx=1024 and ctx=2048, while performing slightly worse (about 1.2%) at ctx=512. This is exactly the effect expected from the numerical stability improvements (Section 3). At the base configuration (context=1024), ProtoT v2 is behind LLaMA-3, Mamba-3, and GDN-2 by about 13.9%, 20.5%, and 22.24% perplexity, respectively. Part of the performance gap is due to Mamba-3 and GDN-2 being very recent models that have obtained massive gains in two generations since their predecessors Mamba and DeltaNet. Also, the results show that while ProtoT v2 solves the scalability issues of ProtoT as it scales from ctx=512 to ctx=1024, it does not solve them from ctx=1024 to ctx=2048, where the other baselines extend their gains further (17.1%, 25.9%, and 28.8%).

Table 1: Base-scale model comparison, across different context lengths (default is 1024). Perplexity: lower is better. Best values are in bold.

Model	Context 512	Context 1024	Context 2048
LLaMA-3	27.30	22.73	20.56
Mamba-3	25.96	21.49	19.12
GDN-2	**25.80**	**21.18**	**18.69**
ProtoT	31.77	28.05	26.07
ProtoT v2	32.16	25.89	24.07

In Appendix A.5 (Table 15), we also explore the training speed at long context lengths at the base setup. The results show that even at the longest context sizes tested (16k and 32k), ProtoT v2 is more than twice as fast as the second fastest model.

Large-Scale Performance The results in Table 2 show a significantly different conclusion than the base-scale results in Table 1: ProtoT v2 scales much better against all baselines: with only 12.2%, 6.2%, and 11.9% worse perplexity than LLaMA-3, Mamba-3, and GDN-2, respectively. This could be due to a similar phenomenon observed by Yordanov et al. (2026) where there is lesser cross-sequence communication bottleneck caused by the hidden dimension at the large scale. As expected, the linear-compute baselines underperform LLaMA-3 at this scale, unlike the base scale where they perform much better. We also observe a significant speed advantage of ProtoT v2 over all other models: 20% faster than LLaMA-3, 46% faster than Mamba-3, and 82% faster than GDN-2. This could mean that, at this scale, ProtoT v2 can be a sensible alternative to the linear-compute baseline architectures just based on the speed-to-performance tradeoff: it has 6% worse perplexity than Mamba, but 46% faster; 12% worse perplexity than GDN-2, but 82% faster.

Table 2: Large-scale model comparison. Throughput is averaged over the final 70% of training. Best values are in bold.

Model	Perplexity ↓	Parameters (M) ↓	Throughput (it/s) ↑	GFLOP/token ↓
LLaMA-3	**12.90**	404.48	6.79	2.23
Mamba-3	13.63	**378.20**	5.59	2.08
GDN-2	12.93	**378.08**	4.49	2.05
ProtoT v2	14.47	392.70	**8.16**	**1.98**

Text-Generation Performance Following Yordanov et al. (2026), we compare open-ended continuations from the large-scale models on the same prompts using a pairwise LM-as-a-judge evaluation inspired by Chatbot Arena (Chiang et al., 2024). For each model pair, a frozen judge assesses coherence, relevance, fluency, plausibility, and lack of repetition, choosing the better continuation or a tie; we reverse response order to reduce position bias, then aggregate wins and ties into order-independent Bradley–Terry ratings on an Elo scale. We replace their Gemma-3-4B-IT judge with the more powerful DeepSeek-V4-Flash-0731 (DeepSeek-AI, 2026). The resulting ratings should be interpreted as relative preferences under this judge and prompt set, rather than compared numerically with ProtoT's published Elo scores, which used a different judge (Yordanov et al., 2026).

Table 3: Large-scale text-generation results on 100 validation prompts. BLEU and ROUGE-L are reported as percentages; judge score is wins plus half of ties over 600 judgments per model. Bradley–Terry Elo is centered at 1000; larger values are better.

Model	BLEU ↑	ROUGE-L ↑	Judge score ↑	BT Elo ↑
LLaMA-3	0.338	13.958	50.58%	1003.19
Mamba-3	0.330	13.758	52.67%	1014.21
GDN-2	**0.473**	13.311	**58.92%**	**1047.50**
ProtoT v2	0.280	**13.980**	37.83%	935.10

As shown in Table 3, GDN-2 leads the DeepSeek-judged comparisons, whereas ProtoT v2 obtains the highest ROUGE-L but lowest judge score. The differing rankings illustrate that reference overlap and pairwise judge preference capture distinct aspects of continuation performance.

Benchmarks To disentangle general language competence, context-dependent recall, and out-of-domain generalization, we constructed a three-tier evaluation suite combining standard zero-shot tasks from the LM Evaluation Harness (Gao et al., 2024) (16 total scores), controlled retrieval tasks from RULER (Hsieh et al., 2024) (37 total scores), and out-of-domain language modeling (8 total scores). The full results are presented in Appendix A.6, Tables 16,17,18, 19,20,21.

While ProtoT v2 underperforms LLaMA-3, Mamba-3, and GDN-2 across most categories, it outperforms LLaMA-3 in some metrics such as BoolQ (Clark et al., 2019) (binary reading comprehension) and SQuAD Completion (Rajpurkar et al., 2016) (Open-book answer retrieval). ProtoT v2 particularly underperforms in out-of-domain language modeling (measured by perplexity), however, this is likely heavily influenced by the fact that ProtoT v2 has a lower in-domain perplexity than the three other large-scale baselines (Table 2).

5.2 EDITABILITY

Editing ProtoT means editing its routing. Yordanov et al. (2026) did so subtractively, by masking or re-initializing a channel, which limits the intervention to knowledge already present and disentangled in the prototypes. We introduce ProtoEdit, which creates a synthetic prototype for a target concept and makes it compete in the write-gate softmax alongside the learned prototypes, so that it modulates what tokens write. This allows editing arbitrary concepts, whether or not the model reserved a channel for them.

Formally, a synthetic prototype is a pair $(\mathbf{p}^\star, \mathbf{v}^\star)$ read off one layer: the mean mixer input and the mean value-stream state over concept-bearing token positions, both centered on a contrast set to isolate the concept direction from generic token statistics. At inference $\mathbf{p}^\star$ joins the competition of Eq. (1) as an $(R{+}1)$-th prototype, and its share of the routing budget scales an injection of its own content into the value stream, before the aggregation of Eq. (2):

$$w_i = \left[\mathrm{Softmax}_{k\in\{1,\dots,R+1\}}\left(\frac{\mathbf{x}_i^\top \mathbf{p}_k}{\tau_{\mathrm{w}}}\right)\right]_{R+1}, \quad \mathbf{p}_k = \begin{cases}\mathbf{p}_k^{\mathrm{w}} & k \le R\\ \mathbf{p}^\star & k = R+1\end{cases}, \quad \tilde{\mathbf{v}}_i = \mathbf{v}_i + c\, w_i \frac{\mathbf{v}^\star}{\|\mathbf{v}^\star\|}. \tag{4}$$

Nothing else in the mixer changes: the edit is carried by the model's own channels, discounted and read out on their existing schedule. A variant scoring the gate independently of the learned bank, $w_i = \mathrm{relu}(\cos(\mathbf{x}_i, \mathbf{p}^\star))$, and further ablations are reported in Appendix A.7.

Baselines and protocol. We compare ProtoEdit against ActAdd (Turner et al., 2023) and the four plug-and-play recipes of AxBench (Wu et al., 2025) (DiffMean, PCA, LAT, and a linear Probe), each applied to the residual stream of both ProtoT v2 and the parameter-matched LLaMA-3. Every arm selects its layer and coefficient with the same rule on a held-out tuning slice: the largest $|c|$ within a fluency, diversity and prompt-relevance budget set by that arm's own unedited generations. We evaluate sentiment editing and detoxification, pooling generations over 5 draws. We report the edit success rate and mean toxicity with 95% prompt-level bootstrap CIs, and the referee model's median perplexity as a fluency measure. Full details are in Appendix A.7.

Sentiment editing. The task is to continue an IMDb (Maas et al., 2011) review with the opposite sentiment, in both directions (Table 4). ProtoEdit raises the edit success rate from 0.567 to 0.973 on Neg→Pos and from 0.191 to 0.874 on Pos→Neg. On Neg→Pos it is comparable to DiffMean on ProtoT v2 (0.964) and below DiffMean on LLaMA-3 (0.986), at similar or lower perplexity. On the harder Pos→Neg direction, DiffMean on ProtoT v2 edits more often (0.916) but at higher perplexity (14.48 vs. 11.24); among arms above 0.85 in this direction, ProtoEdit has the lowest perplexity. Probe, PCA and LAT are consistently weaker.

Detoxification. The task is to continue a toxicity-eliciting RealToxicityPrompts (Gehman et al., 2020) opening without producing toxic text (Table 4). DiffMean and ActAdd on ProtoT v2 give the lowest toxicity (0.0245 and 0.0384). ProtoEdit reduces toxicity by roughly 80% (0.2315 to 0.0474), comparable to Probe on ProtoT v2 (0.0460) and to the strongest LLaMA-3 arms (Probe, 0.0508; DiffMean, 0.0542), at lower perplexity than DiffMean and ActAdd on ProtoT v2. Overall, ProtoEdit does not dominate the baselines but is competitive with the strongest of them, while expressing the edit as an explicit prototype.

ProtoT v2 is more editable as a backbone. Because the two backbones start from different unedited rates, we compare them by the gain over each backbone's own unedited model (Appendix Table 22). Applying the same method to ProtoT v2 rather than LLaMA-3 gives a larger gain in 10 of the 15 (method, condition) comparisons and a smaller one in 3, while 2 are indistinguishable (Probe and PCA on detoxification). The gap is largest for ActAdd (gain 0.685 vs. 0.335 on Pos→Neg; 0.834 vs. 0.525 on detoxification) and for LAT and DiffMean on detoxification. LLaMA-3 is ahead for DiffMean on Neg→Pos and for Probe and LAT on Pos→Neg. Since none of the baselines access the prototype bank, this suggests that part of ProtoT v2's editability is already present in its representations, with ProtoEdit adding a prototype-level interface on top.

In Appendix A.8, we also include results on existing prototype editing – the method used in ProtoT (Yordanov et al., 2026). While showing low collateral damage, the method has very low impact on

Table 4: *Sentiment editing and detoxification.* IMDb sentiment flip (n=5000/arm/direction, 5 draws) and RealToxicityPrompts detoxification (n=4350/arm, 5 draws). Edit rate ↑; toxicity is mean Toxic-BERT score ↓; Ppl is the referee model's median perplexity ↓. 95% CIs are prompt-level percentile bootstrap intervals (10^4 resamples), pooling the 5 draws. **Bold** = the better backbone for that method on that metric. Gains over each backbone's unedited model are reported in Appendix Table 22. ProtoEdit has no LLaMA-3 counterpart. Proto-2 = ProtoT v2.

		Neg → Pos		Pos → Neg		Detoxification	
Method	Backbone	Edit rate ↑	Ppl ↓	Edit rate ↑	Ppl ↓	Toxicity ↓	Ppl ↓
No editing	Proto-2	**0.567** [0.551, 0.583]	8.27	0.191 [0.178, 0.203]	8.11	0.2315 [0.213, 0.250]	9.33
	LLaMA-3	0.501 [0.485, 0.517]	**7.85**	**0.207** [0.194, 0.220]	**7.81**	**0.2157** [0.198, 0.234]	**9.18**
ProtoEdit	Proto-2	**0.973** [0.967, 0.978]	**9.62**	**0.874** [0.863, 0.884]	**11.24**	**0.0474** [0.039, 0.056]	**11.85**
DiffMean	Proto-2	0.964 [0.958, 0.970]	**9.44**	**0.916** [0.907, 0.924]	14.48	**0.0245** [0.018, 0.031]	12.34
	LLaMA-3	**0.986** [0.981, 0.989]	12.17	0.853 [0.842, 0.864]	**13.94**	0.0542 [0.046, 0.063]	**10.96**
Probe	Proto-2	**0.921** [0.912, 0.929]	11.45	0.684 [0.669, 0.699]	**9.82**	**0.0460** [0.038, 0.055]	11.72
	LLaMA-3	0.876 [0.866, 0.886]	**9.35**	**0.769** [0.756, 0.783]	9.93	0.0508 [0.043, 0.059]	**11.08**
PCA	Proto-2	**0.792** [0.779, 0.805]	15.86	**0.306** [0.291, 0.321]	**10.69**	0.1022 [0.091, 0.114]	**11.27**
	LLaMA-3	0.506 [0.490, 0.522]	**7.75**	0.247 [0.234, 0.261]	11.06	**0.0930** [0.083, 0.104]	14.01
LAT	Proto-2	**0.756** [0.743, 0.770]	**14.81**	0.244 [0.231, 0.258]	**8.15**	**0.0973** [0.086, 0.109]	**10.72**
	LLaMA-3	0.603 [0.587, 0.619]	15.66	**0.306** [0.292, 0.321]	11.82	0.1621 [0.147, 0.178]	12.17
ActAdd	Proto-2	**0.918** [0.909, 0.927]	**10.93**	**0.745** [0.731, 0.759]	13.21	**0.0384** [0.032, 0.045]	**12.96**
	LLaMA-3	0.874 [0.863, 0.884]	12.50	0.472 [0.457, 0.488]	**10.80**	0.1024 [0.089, 0.117]	13.06

model predictions, and has the inherent limitation of only being able to edit concepts already present in existing prototypes.

5.3 Interpretability



Figure 2: Top activating sequences for Prototype 44 in 7th layer. Colors indicate activation strength.

Table 5: Interpretability: disentanglement, main-theme coverage, and number of themes. Higher is better for disentanglement and coverage; fewer themes is better. **Bold** = best per column.

Model	**Disentanglement ↑**	**Coverage (Main Theme) ↑**	**# Themes ↓**
ProtoT v2	2.728 ± 1.498	2.562 ± 2.175	7.418 ± 2.542
LLaMA-3	2.382 ± 1.290	2.084 ± 1.646	8.076 ± 2.209
ProtoT v2 null	2.016 ± 0.516	1.318 ± 0.600	9.013 ± 1.287
LLaMA-3-SAE	**3.051 ± 1.978**	**3.087 ± 2.765**	**6.976 ± 2.907**

In this experiment, we use the same prototype interpretability method, interpretability baselines, and LLM evaluation pipeline as Yordanov et al. (2026), with the only difference that we use DeepSeek-V4-Flash-0731 (DeepSeek-AI, 2026) to handle the larger volume of prototypes and features at this scale. We interpret each prototype at the Write Gate (Eq. 1) through the top-10 input token sequences it activates most at, then inside each of the 10 sequences, we select the tokens that are most activated by that prototype (e.g. the prototype in Figure 2 activates at foreign languages). We feed this to an LLM pipeline to obtain scores for: 1) Disentanglement: how well the main theme is separated from other teams; 2) Coverage: how many of the 10 sequences the main theme covers; and 3) Number of distinct themes. The baseline methods are as follows: 1) LLaMA-3: we inspect its attention heads as

Table 6: Robustness (mean ± SD). Bold marks the best displayed mean per column; arrows give the preferred direction. Categories have equal weight. Prediction loss is lower for better predictions; increase is loss after typos minus clean loss (nats/token; 128 prompts).

	Perturbation stability			Prediction loss	
Model	**JS ↓**	**Overlap@10 ↑**	**Top-1 agreement ↑**	**Clean ↓**	**Increase ↓**
LLaMA-3	0.0532 ± 0.1005	0.802 ± 0.176	0.795 ± 0.404	$\mathbf{2.441} \pm 0.954$	0.0077 ± 0.0168
Mamba-3	0.0565 ± 0.1057	$\mathbf{0.818} \pm 0.172$	0.731 ± 0.444	2.518 ± 0.946	$\mathbf{0.0058} \pm 0.0152$
GDN-2	$\mathbf{0.0484} \pm 0.0989$	0.812 ± 0.170	$\mathbf{0.803} \pm 0.398$	2.458 ± 0.934	0.0066 ± 0.0166
ProtoT v2	0.0507 ± 0.1002	0.804 ± 0.172	0.725 ± 0.447	2.586 ± 0.914	0.0069 ± 0.0151

features; 2) SAE: we train a sparse auto-encoder (Bricken et al., 2023) to extract more disentangled features from LLaMA-3; 3) ProtoT v2 null: 10 random sequences for each prototype slot.

The results in Table 5 show that the ProtoT v2 prototypes have better disentanglement and coverage, and fewer themes than LLaMA-3's attention heads, which means a better native interpretability of ProtoT v2. On the other hand, the ProtoT v2 prototypes show worse performance than the specialized SAE method on all 3 interpretability metrics. However, it is worth noting that the SAE method requires additional training, unlike prototype interpretability.

5.4 Robustness

We test ProtoT v2's read routes as an intervention point for input perturbations. At the final token, we replace the perturbed prompt's read-gate probabilities ρ (the second gate in Eq. 1) with those from its clean counterpart in each layer. Restoring all 18 layers reduces Jensen–Shannon divergence (JS) by 0.0022 ± 0.0165 across 1,422 unique pairs (mean ± SD; equal category weights). In this all-layer intervention, matched routes give lower mean JS than unrelated or permuted routes. Model parameters remain fixed; the intervention requires the clean prompt as a reference.

We also evaluate all four large models on the released ProtoT benchmark (Yordanov et al., 2026). Averaged across its seven perturbation categories, ProtoT v2 has lower mean JS than LLaMA-3 and Mamba-3, with GDN-2 giving the lowest mean (Table 6). Adding typos to one or three words increases ProtoT's loss on the following text by 0.0069 ± 0.0151 nats/token, within the range of the baselines' mean increases.

Appendix A.9 gives category-level and semantic-tag results, routing controls, the final-layer comparison, and the evaluation and weighting details.

6 Conclusion

In this work, we introduced the ProtoT v2 prototype transformer model, a linear-computation alternative to self-attention transformer models that uses communication through prototypes, which can be interpreted and edited. Through a number of architectural and implementation improvements, ProtoT v2 constitutes a substantial upgrade over ProtoT (Yordanov et al., 2026) in terms of architectural simplicity, perplexity, scalability, and speed. We also developed a custom Triton kernel that yields about 3 times faster speed and 3 times less memory usage than ProtoT. Furthermore, we introduced a new synthetic-prototype editability method which enables much more flexible and powerful concept editability, on par with strong editability baseline methods, and we showed that ProtoT v2 is a particularly editable backbone. Alongside its interpretability and editability, ProtoT v2 shows broadly comparable robustness to the other models when reading text with small wording changes or typing errors. While ProtoT v2 lags behind state-of-the-art linear-computation baselines in terms of performance, its superior speed positions it as a sensible speed-to-performance alternative, while having native interpretability and editability tools.

Future work can look into scaling ProtoT v2 to even larger data and model sizes, pushing it to production scale. It can also develop specific application domains for the model, e.g. medicine or law, where reliability through interpretability and editability are important considerations.

Reproducibility Statement

We provide a comprehensive description of the model and the experimental setup in the main paper; we also provide appendices for all training details, hyperparameter search results, and ablations.

Ethics Statement

The ProtoT v2 model significantly enhances the editability properties of its predecessor. As such, it poses both risks and benefits: editability can, in theory, be used both for AI safety, but also for bypassing internal safety mechanisms. Currently, we do not think ProtoT v2 poses real risks, as it lags in performance benchmarks behind state-of-the-art LMs, but this can change in future generations of the model. Finally, on balance, we believe that ProtoT v2 being more performant and editable than its predecessor is better for the future of AI, as this family of models shows superior native interpretability to transformer models.

AI Usage

We have used LLMs for checking and polishing writing, for AI judge scoring (as described in the paper), for low-level help with coding (e.g. creating the model diagram in HTML based on human instructions), and for assistance in creating the Triton Kernel (detailed below). We reviewed all AI-assisted outputs and took steps to verify their correctness and reliability.

We have used LLMs extensively in developing the Triton kernel implementation on top of our numerically stable formulation. We specified what the kernel should compute and iteratively guided its development, including setting performance goals, asking the LLMs to identify bottlenecks and suggest optimizations, and testing alternative implementations. We validated the resulting kernel by comparing its numerical outputs with a reference implementation within specified numerical tolerances, benchmarking its runtime performance, and checking that it did not exploit unintended shortcuts or properties of the evaluation setup. We reviewed the final implementation and take responsibility for its correctness and for the reported results.

References

Guillaume Alain and Yoshua Bengio. Understanding intermediate layers using linear classifier probes. In *ICLR (Workshop)*. OpenReview.net, 2017.

Anthropic. Introducing Claude Fable 5.1 and Claude Mythos 5.1, September 2026. URL `https://www.anthropic.com/claude-fable-and-mythos-5-1`.

Sercan O Arik and Tomas Pfister. Protoattend: Attention-based prototypical learning. *Journal of Machine Learning Research*, 21(210):1–35, 2020.

Artificial Analysis. Artificial analysis intelligence benchmarking methodology, 2026. URL `https://artificialanalysis.ai/methodology/intelligence-benchmarking`.

Nora Belrose, Igor Ostrovsky, Lev McKinney, Zach Furman, Logan Smith, Danny Halawi, Stella Biderman, and Jacob Steinhardt. Eliciting latent predictions from transformers with the tuned lens. *arXiv preprint arXiv:2303.08112*, 2023.

Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E Burke, Tristan Hume, Shan Carter, Tom Henighan, and Christopher Olah. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023. https://transformer-circuits.pub/2023/monosemantic-features/index.html.

Jonah Brown-Cohen, David Lindner, and Rohin Shah. Quantifying the necessity of chain of thought through opaque serial depth. *arXiv preprint arXiv:2603.09786*, 2026.

Chaofan Chen, Oscar Li, Daniel Tao, Alina Barnett, Cynthia Rudin, and Jonathan K Su. This looks like that: deep learning for interpretable image recognition. *Advances in Neural Information Processing Systems*, 32, 2019.

Yanda Chen, Joe Benton, Ansh Radhakrishnan, Jonathan Uesato, Carson Denison, John Schulman, Arushi Somani, Peter Hase, Misha Wagner, Fabien Roger, Vlad Mikulik, Samuel R. Bowman, Jan Leike, Jared Kaplan, and Ethan Perez. Reasoning models don't always say what they think. *arXiv preprint arXiv:2505.05410*, 2025.

Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Banghua Zhu, Hao Zhang, Michael Jordan, Joseph E Gonzalez, et al. Chatbot arena: An open platform for evaluating llms by human preference. In *Forty-first International Conference on Machine Learning*, 2024.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2924–2936, 2019.

Clay Mathematics Institute. Navier-stokes announcement, September 2026. URL https://www.claymath.org/news/navier-stokes-announcement/.

Arthur Conmy, Augustine Mavor-Parker, Aengus Lynch, Stefan Heimersheim, and Adrià Garriga-Alonso. Towards automated circuit discovery for mechanistic interpretability. In *Advances in Neural Information Processing Systems*, 2023.

DeepSeek-AI. DeepSeek-V4-Flash-0731 model card. Hugging Face, 2026. URL https://huggingface.co/deepseek-ai/DeepSeek-V4-Flash-0731.

Shizhe Diao, Yu Yang, Yonggan Fu, Xin Dong, Dan Su, Markus Kliegl, Zijia Chen, Peter Belcak, Yoshi Suhara, Hongxu Yin, Mostofa Patwary, Yingyan (Celine) Lin, Jan Kautz, and Pavlo Molchanov. Nemotron-climb: Clustering-based iterative data mixture bootstrapping for language model pre-training. In *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2025, NeurIPS 2025, San Diego, CA, USA, December 2-7, 2025 / Mexico City, Mexico, November 30 - December 5, 2025*, 2025.

Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, et al. Toy models of superposition. *arXiv preprint arXiv:2209.10652*, 2022.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 2024. URL https://zenodo.org/records/12608602.

Leo Gao, Achyuta Rajaram, Jacob Coxon, Soham V Govande, Bowen Baker, and Dan Mossing. Weight-sparse transformers have interpretable circuits. *arXiv preprint arXiv:2511.13653*, 2025.

Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A. Smith. Realtoxicityprompts: Evaluating neural toxic degeneration in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2020*. Association for Computational Linguistics, 2020.

Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. Dissecting recall of factual associations in auto-regressive language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP*, 2023.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First Conference on Language Modeling*, 2024.

Laura Hanu and Unitary Team. Detoxify. GitHub, 2020. URL https://github.com/unitaryai/detoxify.

Jochen Hartmann, Mark Heitmann, Christian Siebert, and Christina Schamp. More than a feeling: Accuracy and application of sentiment analysis. *International Journal of Research in Marketing*, 40(1):75–87, 2023.

Ali Hatamizadeh, Yejin Choi, and Jan Kautz. Gated deltanet-2: Decoupling erase and write in linear attention. *arXiv preprint arXiv:2605.22791*, 2026.

John Hewitt, John Thickstun, Christopher D. Manning, and Percy Liang. Backpack language models. In *ACL (1)*, pp. 9103–9125. Association for Computational Linguistics, 2023.

Dat Hong, Tong Wang, and Stephen Baek. Protorynet-interpretable text classification via prototype trajectories. *Journal of Machine Learning Research*, 24(264):1–39, 2023.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. Ruler: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*, 2024.

Sarthak Jain and Byron C. Wallace. Attention is not explanation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, 2019.

Connor Kissane, Robert Krzyzanowski, Joseph Isaac Bloom, Arthur Conmy, and Neel Nanda. Interpreting attention layer outputs with sparse autoencoders. *arXiv preprint arXiv:2406.17759*, 2024.

Aakash Sunil Lahoti, Kevin Y. Li, Berlin Chen, Caitlin Wang, Aviv Bick, J. Zico Kolter, Tri Dao, and Albert Gu. Mamba-3: Improved sequence modeling using state space principles. In *International Conference on Learning Representations*, 2026.

Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, Kamile Lukosiute, Karina Nguyen, Newton Cheng, Nicholas Joseph, Nicholas Schiefer, Oliver Rausch, Robin Larson, Sam McCandlish, Sandipan Kundu, Saurav Kadavath, Shannon Yang, Thomas Henighan, Timothy Maxwell, Timothy Telleen-Lawton, Tristan Hume, Zac Hatfield-Dodds, Jared Kaplan, Jan Brauner, Samuel R. Bowman, and Ethan Perez. Measuring faithfulness in chain-of-thought reasoning. *arXiv preprint arXiv:2307.13702*, 2023.

Houyi Li, Wenzheng Zheng, Jingcheng Hu, Qiufeng Wang, Hanshan Zhang, Zili Wang, Shijie Xuyang, Yuantao Fan, Shuigeng Zhou, Xiangyu Zhang, and Daxin Jiang. Predictable scale: Part i – optimal hyperparameter scaling law in large language model pretraining. *arXiv preprint arXiv:2503.04715*, 2025.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, 2016.

Kenneth Li, Oam Patel, Fernanda B. Viégas, Hanspeter Pfister, and Martin Wattenberg. Inference-time intervention: Eliciting truthful answers from a language model. In *NeurIPS*, 2023.

Shuaipeng Li, Penghao Zhao, Hailin Zhang, Xingwu Sun, Hao Wu, Dian Jiao, Weiyan Wang, Chengjun Liu, Zheng Fang, Jinbao Xue, Yangyu Tao, Bin Cui, and Di Wang. Surge phenomenon in optimal learning rate and batch size scaling. *Advances in Neural Information Processing Systems*, 37, 2024.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR*, 2019.

Chiyu Ma, Jon Donnelly, Wenjun Liu, Soroush Vosoughi, Cynthia Rudin, and Chaofan Chen. Interpretable image classification with adaptive prototype-based vision transformers. *Advances in Neural Information Processing Systems*, 37, 2024.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, 2011.

Andrew Mack, Kraig Yuheng Tou, Mark Henry, Zhengxun Wu, and Lauren Greenspan. Scaling interpretable transformers with parity bottleneck layers. *arXiv preprint arXiv:2607.20652*, 2026.

Lovish Madaan, Aaditya K. Singh, Rylan Schaeffer, Andrew Poulton, Sanmi Koyejo, Pontus Stenetorp, Sharan Narang, and Dieuwke Hupkes. Quantifying variance in evaluation benchmarks. *arXiv preprint arXiv:2406.10229*, 2024.

Samuel Marks and Max Tegmark. The geometry of truth: Emergent linear structure in large language model representations of true/false datasets. *arXiv preprint arXiv:2310.06824*, 2023.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. *Advances in Neural Information Processing Systems*, 35, 2022.

Yao Ming, Panpan Xu, Huamin Qu, and Liu Ren. Interpretable and steerable sequence learning via prototypes. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019.

Nostalgebraist. Interpreting GPT: the logit lens, August 2020. URL https://www.lesswrong.com/posts/AcKRB8wDpdaN6v6ru/interpreting-gpt-the-logit-lens.

OpenAI. GPT-6 Astra: A new generation of intelligence, September 2026a. URL https://openai.com/index/gpt-6-astra/.

OpenAI. On the Navier–Stokes Millennium Prize Problem, September 2026b. URL https://openai.com/index/navier-stokes-solution/.

Jakub Pachocki. An alien mind, September 2026. URL https://openai.com/index/an-alien-mind/.

Gonçalo Santos Paulo, Alex Troy Mallen, Caden Juang, and Nora Belrose. Automatically interpreting millions of features in large language models. In *Forty-second International Conference on Machine Learning*, 2025.

Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin A Raffel, Leandro Von Werra, Thomas Wolf, et al. The FineWeb datasets: Decanting the web for the finest text data at scale. *Advances in Neural Information Processing Systems*, 37, 2024.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. Technical report, OpenAI, 2019.

Daking Rai, Yilun Zhou, Shi Feng, Abulhair Saparov, and Ziyu Yao. A practical review of mechanistic interpretability for transformer-based language models. *arXiv preprint arXiv:2407.02646*, 2024.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392, 2016.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, 2019.

Dawid Rymarczyk, Łukasz Struski, Michał Górszczak, Koryna Lewandowska, Jacek Tabor, and Bartosz Zieliński. Interpretable image classification with differentiable prototypes assignment. In *European Conference on Computer Vision*, 2022.

Nathan Sheffield, Alek Westover, Lukas Finnveden, Alexa Pan, Julian Stastny, and Ryan Greenblatt. An operationalization of opaque serial depth, September 2026. URL https://www.redwoodresearch.org/blog/an-operationalization-of-opaque-serial-depth.

Chung-En Sun, Tuomas P. Oikarinen, Berk Ustun, and Tsui-Wei Weng. Concept bottleneck large language models. In *ICLR*. OpenReview.net, 2025.

Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, Hoagy Cunningham, Nicholas L. Turner, Callum McDougall, Monte MacDiarmid, Alex Tamkin, Esin Durmus, Tristan Hume, Francesco Mosconi, C. Daniel Freeman, Theodore R. Sumers, Edward Rees, Joshua Batson, Adam Jermyn, Shan Carter, Chris Olah, and Tom Henighan. Scaling monosemanticity: Extracting interpretable features from Claude 3 Sonnet. *Transformer Circuits Thread*, 2024. URL https://transformer-circuits.pub/2024/scaling-monosemanticity/.

Curt Tigges, Oskar John Hollinsworth, Atticus Geiger, and Neel Nanda. Linear representations of sentiment in large language models. *CoRR*, abs/2310.15154, 2023.

Alexander Matt Turner, Lisa Thiergart, David Udell, Gavin Leech, Ulisse Mini, and Monte MacDiarmid. Activation addition: Steering language models without optimization. *arXiv preprint arXiv:2308.10248*, 2023.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel Bowman. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. *Advances in Neural Information Processing Systems*, 36, 2023.

Kevin Wang, Alexandre Variengien, Arthur Conmy, Buck Shlegeris, and Jacob Steinhardt. Interpretability in the wild: A circuit for indirect object identification in GPT-2 small. *arXiv preprint arXiv:2211.00593*, 2022.

Bowen Wei and Ziwei Zhu. Protolens: Advancing prototype learning for fine-grained interpretability in text classification. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2025.

Ximing Wen and Rezvaneh Rezapour. A transformer and prototype-based interpretable model for contextual sarcasm detection. *arXiv preprint arXiv:2503.11838*, 2025.

Zhengxuan Wu, Aryaman Arora, Zheng Wang, Atticus Geiger, Dan Jurafsky, Christopher D. Manning, and Christopher Potts. Reft: Representation finetuning for language models. In *NeurIPS*, 2024.

Zhengxuan Wu, Aryaman Arora, Atticus Geiger, Zheng Wang, Jing Huang, Dan Jurafsky, Christopher D. Manning, and Christopher Potts. Axbench: Steering llms? even simple baselines outperform sparse autoencoders. In *ICML*, volume 267 of *Proceedings of Machine Learning Research*. PMLR / OpenReview.net, 2025.

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. *Advances in Neural Information Processing Systems*, 37, 2024.

Yordan Yordanov, Matteo Forasassi, Bayar Menzat, Ruizhi Wang, Chang Qi, Markus Kaltenberger, Amine M'Charrak, Tommaso Salvatori, and Thomas Lukasiewicz. Prototype transformer: Towards language model architectures interpretable by design. In *Forty-third International Conference on Machine Learning*, 2026.

Fred Zhang and Neel Nanda. Towards best practices of activation patching in language models: Metrics and methods. In *The Twelfth International Conference on Learning Representations*, 2024.

Andy Zou, Long Phan, Sarah Li Chen, James Campbell, Phillip Guo, Richard Ren, Alexander Pan, Xuwang Yin, Mantas Mazeika, Ann-Kathrin Dombrowski, Shashwat Goel, Nathaniel Li, Michael J. Byun, Zifan Wang, Alex Mallen, Steven Basart, Sanmi Koyejo, Dawn Song, Matt Fredrikson, J. Zico Kolter, and Dan Hendrycks. Representation engineering: A top-down approach to ai transparency. *arXiv preprint arXiv:2310.01405*, 2023.

A Appendix

A.1 Triton Kernel

Chunked Triton kernel We implement the discounted prefix aggregation, mass normalization, and read-gate contraction as a custom chunked Triton operator. The operator directly computes:

$$\mathbf{z}_i = \sum_{k=1}^{R} \rho_{ik} \frac{\mathbf{s}_{i-1,k}}{\max(m_{i-1,k}, \varepsilon)}, \qquad \mathrm{Mix}(\mathbf{x})_i = \alpha U(\mathbf{z}_i). \tag{5}$$

Thus, only the output projection U and residual scale α are applied outside the custom operator.

Expanding the recurrent state gives:

$$\mathbf{z}_i = \sum_{j<i} K_{ij} \mathbf{v}_j, \qquad K_{ij} = \sum_{k=1}^{R} \frac{\rho_{ik}(1-\beta_k)\pi_{jk}\beta_k^{i-j-1}}{\max(m_{i-1,k}, \varepsilon)}. \tag{6}$$

This defines a causal semiseparable kernel: interactions between earlier writes and later reads factor through the R communication channels, so every off-diagonal block has rank at most R.

The implementation divides the sequence into chunks of T tokens and:

- computes the discounted routing mass and normalized read gates;
- computes one boundary memory state for each communication channel and chunk, and scans these states across chunks;
- computes within-chunk interactions using tiled matrix multiplications; and
- combines the within-chunk result with the state carried from all preceding chunks.

For local positions $j < i$ in the same chunk, the decay is factorized about a pivot c:

$$\beta_k^{i-j-1} = \beta_k^{i-c} \beta_k^{c-j-1}. \tag{7}$$

Let $\widetilde{Q}_{ik} = q_{ik}\beta_k^{i-c}$, and $\widetilde{W}_{jk} = (1-\beta_k)\pi_{jk}\beta_k^{c-j-1}$, where $q_{ik} = \rho_{ik}/\max(m_{i-1,k}, \varepsilon)$. The within-chunk kernel can then be written as:

$$K_{ij} = \sum_{k=1}^{R} \widetilde{Q}_{ik}\widetilde{W}_{jk}, \qquad j < i. \tag{8}$$

Thus, before applying the causal mask, the within-chunk kernel is the matrix product:

$$K = \widetilde{Q}\widetilde{W}^{\mathsf{T}}. \tag{9}$$

This converts the within-chunk interaction into a product of two routing matrices and allows it to be evaluated efficiently using tensor-core matrix multiplications. The custom backward pass recomputes the within-chunk interaction matrices from the saved routing weights and decay parameters rather than saving them during the forward pass, trading a small amount of repeated computation for lower activation memory and reduced memory (HBM) traffic.

The kernel does not materialize either the routing–value outer product or the full per-token vector-valued memory state. It stores the vector-valued state only at chunk boundaries, reducing its size from $\mathcal{O}(BSRC_v)$ to $\mathcal{O}(B\lceil S/T\rceil RC_v)$. The much smaller scalar routing-mass state is retained per token for normalization and backward computation. Recurrent quantities are accumulated in FP32, while the projected values, read weights, and output can remain in BF16 or FP16.

Short time scales For small β_k, the pivot factorization in Eq. 7 can produce large intermediate negative powers of β_k, even though the final decay exponent is non-negative. To address this, under the default configuration, each $T = 64$ token chunk is divided into 4 sub-blocks of width 16, and:

1. Pairs in the same sub-block use a nearby pivot at the sub-block midpoint. This bounds the magnitude of each negative exponent by half the sub-block width.

2. Pairs in neighboring sub-blocks use their shared boundary as the pivot, producing only non-negative exponents.
3. Within-chunk pairs separated by more than one sub-block are assigned zero. Under the default threshold $\beta_k < 2^{-3}$, their decay exponent is at least 16, so their decay weight is bounded above by 2^{-48}.

Together with clamping the effective decay to $\beta_k \geq 2^{-12}$ inside the kernel, this bounds all intermediate factors and prevents overflow, while the omitted decay weights are numerically negligible (no larger than 2^{-48}).

A.2 Training Details

Size Adjustment We use approximately parameter-matched model sizes by varying the hidden size to get to the target 80M parameters at base scale and 400M parameters at the large scale, but avoiding values of the hidden size that are not sufficiently divisible by powers of two, for hardware efficiency. For all models, when scaling from base to large scale, we keep the head dimension fixed, but allow for the head number to increase proportionally to the hidden size. For all models containing (attention) heads, we scale the head number while keeping the head dimension constant as we increase the model size.

Batch Size We use batch size of 32 for the base scale on 1 H100 GPU, and 64 for the large scale spread across 4 H100 GPUs (16 minibatch). For ProtoT, we use 2 gradient accumulation steps at the base scale (due to memory constraints), whereas for ProtoT v2 we do not use gradient accumulation at any scale.

Hyperparameter Search We select the best hyperparameters via targeted grid search, where we make sure that the best perplexity value is on the interior of the grid, rather than on the edge. See Appendix A.4 for how this looks like.

Learning Rate For all models, we search the LR among values 0.30×10^{-3}, 0.50×10^{-3}, 0.75×10^{-3}, 1.00×10^{-3}, 1.50×10^{-3}, 2.00×10^{-3}, and 3.00×10^{-3}.

Learning Rate Scaling Laws After finding the best LR for the base scale, we need to find what LR to use for the large scale. For this, we use a suitable scaling laws for our setup (AdamW with linear warmup and cosine decay) by Li et al. (2025), namely:

$$lr = \alpha N^{-0.713} D^{0.307} \tag{10}$$

where N = non-embedding params, D = tokens processed (repeats counted), and lr = peak LR. We do not use the original $\alpha = 1.79$ they use, but instead we re-calculate it for each model at the base scale, and then use it to predict the LR at the large scale.

Finally, because we switch from 32 batch size at base scale to 64 at large scale, we should increase the LR by a factor of $\sqrt{2}$, due to the well-known scaling law for the AdamW optimizer (Li et al., 2024). The final calculated learning rates for the large-scale setup and other values are presented in Table 7.

Table 7: Learning-rate scaling quantities for the base and large scale model/data configurations. N denotes non-embedding parameters and D denotes tokens processed (including repetitions). The final column is derived based on the scaling law in Eq. 10 followed by the $\sqrt{2}$ batch size adjustment factor.

Model	α	Base N	Base D	Base LR	Large N	Large D	Large LR
ProtoT v2	0.45	28.5M	1.73B	1.50×10^{-3}	264.0M	19.93B	0.919×10^{-3}
LLaMA-3	0.28	38.0M	1.73B	0.75×10^{-3}	288.7M	19.93B	0.528×10^{-3}
GDN-2	0.43	47.5M	1.73B	1.00×10^{-3}	275.2M	19.93B	0.855×10^{-3}
Mamba-3	0.64	46.8M	1.73B	1.50×10^{-3}	275.3M	19.93B	1.270×10^{-3}

Seeds We use 1 seed for all experiments, because we have observed that cross-seed standard deviation at our base data/model scale is very low, at significantly less than 1% of perplexity at the base scale, and that it decreases with the amount of training data. This could be due to the diminishing effect of weight initialization and batch order with more training data. Generally speaking, single-seed pretraining is the most common practice for large LLMs. One notable exception to this are methodological studies where seed variance is central, e.g. (Madaan et al., 2024).

Text-Generation Evaluation Results are descriptive, use one training seed, and reuse the checkpoint-selection validation region. They do not establish a statistically significant ranking and are not directly comparable to the Gemma-judged scores of Yordanov et al. (2026). LLaMA-3 was evaluated with SDPA instead of FlashAttention-2, with verified perplexity agreement.

A.3 Ablation Studies

Table 8: Ablation studies of ProtoT v2. First row is ProtoT, last row is ProtoT v2. Beta grid: constant at 6.5788 or a grid of 4 initialization groups: 16,64,256,1024. Mem: peak allocated memory. Best values are in bold. *–the ProtoT model took more than the maximum 79GB memory and crashed, so this re-run is with 2 gradient accumulation steps.

Numeric Stability	Read Proj	Beta Grid Init	L0 Tau	L0 Shared	Causal Conv	PPL ↓	it/s ↑	Mem ↓ (GB)	Forward GFLOPs ↓
No	Yes	const	3.0	Yes	Yes	28.05	4.67	36.58*	117.44
Yes	Yes	const	3.0	Yes	Yes	25.79	14.01	25.93	118.39
Yes	No	const	3.0	Yes	Yes	**25.72**	14.24	25.57	**112.49**
Yes	No	grid	3.0	Yes	Yes	25.80	14.27	25.57	**112.49**
Yes	No	grid	1.0	Yes	Yes	25.77	14.21	25.57	**112.49**
Yes	No	grid	1.0	No	Yes	25.78	14.17	25.57	112.55
Yes	No	grid	1.0	No	No	25.89	**14.43**	**25.56**	112.55

The full ablation results are presented in Table 8, where ProtoT is represented in the first row and ProtoT v2 is represented in the final row. Overall, the only significant difference in performance, speed, or memory usage are given by: 1. the numerical stability improvements, 2. the kernel, 3. removing the read projection, and 4. removing the causal convolution. We do not see significant effects of the Beta Grid Initialization, L0 tau initialization, or sharing the L0 read/write gates. This is expected because initializations tend to work less well with more training data, and likely because gate sharing affects only 1 out of 12 layers at this scale.

Read Projection The removal of the read projection in ProtoT v2 has no significant impact on perplexity because the read projection is a redundant computation that is equivalent to using a separate set of Write-Gate prototypes (see Section 3). However, it decreases the memory usage and yields a slight speedup.

Numerical Stabilization The numerical stabilization is an integral part of the kernel, and as such it is hard to separate the two implementations, which is why both changes are on row 2 of Table 8. However, we can safely assume that the performance gains of this configuration (row 2) are entirely due to numerical stabilization. As such, we observe about 8% lower perplexity due to the numerical stabilization, which accounts for the vast majority of performance difference between ProtoT v2 and ProtoT.

Kernel: Even though the kernel for ProtoT v2 only optimises the mixer module, we show that it yields about 3 times faster speed at the model level (achieving this by making the mixer many times faster). When matching the minibatch size, ProtoT v2 is about 2.7-2.8 times faster than ProtoT and has about 3 times less memory usage. However, the ProtoT run crashes at epoch 2 on H100 due to out of memory. We work around this issue by using 2 gradient accumulation steps for ProtoT which slows down speed but halves memory usage. At that tested scenario, ProtoT v2 becomes 3 times faster than ProtoT.

Note: the kernel actually adds some additional numerical operations (flops) (see rows $1 \rightarrow 2$), which is due to preferring to re-compute some values instead of relying on slow memory movements, thus actually leading to greater speed in practice.

Beta Grid Initialization Instead of following ProtoT which initializes all β time-discount values to 6.6 half-life, we initialize them to 16, 64, 256, and 1024 half lives, like a grid, where each quarter of the channels get a separate initialization value. We hypothesize that this would allow for more specialized temporal properties of the channels, and our early experiments at smaller scale suggested that this helps slightly in perplexity. However, at the current scale, the results (Table 8, column *Beta Grid Init*) do not show significant performance difference (grid initialization is about 0.3% worse perplexity, which is within the per-seed variation expected at this scale).

Causal Convolution In the ProtoT paper (Yordanov et al., 2026) (in its Table 16 of Appendix A.6.1), the local causal convolution at the value stream yields about 10% improvement on perplexity. However, for ProtoT v2 (Table 8) we show that it has a very small effect on perplexity (about 0.4%), which could be due to more training data (about 27 times more data at the base scale) and better numerical stability of the model.

A.4 Hyperparameter Search

The results in Tables 10, 11, 12, 13, 14 show that our choices of best hyperparameters are well-motivated. For full list of the model configurations used, see Table 9. For all models, using no dropout (0.0) is better than 0.1 at this training scale, and it is the hyperparameter that most affects performance among all tested. Using untied vs tied head yield similar performance for all models. We choose the untied head because we have observed that the tied head works better at smaller scale, whereas we want to scale the model and data from this base scale to the large scale.

Table 9: Selected architecture configurations at the base and large scales. Parameters include embeddings; base-scale totals are approximate. Head counts and head dimensions refer to attention heads for LLaMA-3, SSM heads for Mamba-3, and recurrent heads for GDN-2. ProtoT v2 uses prototype channels instead of heads. All configurations use a 1,024-token context, an untied LM head, zero dropout, and an FFN ratio of 2.7.

Model	Scale	Parameters	Hidden size	Layers	Heads	Head dim.	Other configuration
ProtoT	Base	$\approx$ 80M	512	12	–	–	$R = 64$, $C_v = 192$
	Large	–	–	–	–	–	Not trained/evaluated
ProtoT v2	Base	$\approx$ 80M	512	12	–	–	$R = 64$, $C_v = 192$
	Large	392.70M	1,280	18	–	–	$R = 64$, $C_v = 480$
LLaMA-3	Base	$\approx$ 89.5M	512	12	8	64	Full multi-head attention (no GQA)
	Large	404.48M	1,152	18	18	64	Full multi-head attention (no GQA)
GDN-2	Base	$\approx$ 99.0M	512	12	4	128	Recurrent-only; convolution size 4
	Large	378.08M	1,024	18	8	128	Recurrent-only; convolution size 4
Mamba-3	Base	$\approx$ 98.3M	512	12	8	128	expansion 2; state size 128
	Large	378.20M	1,024	18	16	128	expansion 2; state size 128

A.5 Throughput at Large Contexts

We run all models at the base setup at various context sizes to see how fast each model is when increasing the context size. We use batch size of 1, run on 1 H100 GPU, and do 10,585 training warm-up updates (equal to the number of batches in 1 epoch at the base setup) before timing a subsequent 256 updates.

The results in Table 15 show that ProtoT v2 is faster than all baselines at all context sizes, and more than twice faster than all baselines at context size 16k and above.

Table 10: ProtoT v2: Hyperparameter search. For each configuration, we report the best learning rate together with the immediately lower and higher values tested. Best perplexity for each configuration is in bold. Configuration selected is underlined.

R	Cv	Tied LM Head	Dropout	LR	PPL ↓
32	256	No	0.1	0.75×10^{-3}	28.27
32	256	No	0.1	1.00×10^{-3}	**28.12**
32	256	No	0.1	1.50×10^{-3}	28.20
32	256	Yes	0.0	1.00×10^{-3}	26.63
32	256	Yes	0.0	1.50×10^{-3}	**26.52**
32	256	Yes	0.0	2.00×10^{-3}	26.66
32	256	No	0.0	1.00×10^{-3}	26.63
32	256	No	0.0	1.50×10^{-3}	**26.53**
32	256	No	0.0	2.00×10^{-3}	26.79
64	192	No	0.0	1.00×10^{-3}	25.90
64	192	No	0.0	1.50×10^{-3}	**25.89**
64	192	No	0.0	2.00×10^{-3}	26.05
64	192	Yes	0.0	0.75×10^{-3}	26.27
64	192	Yes	0.0	1.00×10^{-3}	**26.05**
64	192	Yes	0.0	1.50×10^{-3}	26.22

Table 11: ProtoT: Hyperparameter search. For each configuration, we report the best learning rate together with the immediately lower and higher values tested. Best perplexity for each configuration is in bold. Configuration selected is underlined.

R	Cv	Tied LM Head	Dropout	LR	PPL ↓
32	256	No	0.1	1.00×10^{-3}	29.78
32	256	No	0.1	1.50×10^{-3}	**29.59**
32	256	No	0.1	2.00×10^{-3}	29.98
32	256	Yes	0.0	1.50×10^{-3}	28.59
32	256	Yes	0.0	2.00×10^{-3}	**28.20**
32	256	Yes	0.0	3.00×10^{-3}	28.77
32	256	No	0.0	1.50×10^{-3}	28.60
32	256	No	0.0	2.00×10^{-3}	**28.39**
32	256	No	0.0	3.00×10^{-3}	28.83
64	192	No	0.0	1.50×10^{-3}	28.23
64	192	No	0.0	2.00×10^{-3}	**28.05**
64	192	No	0.0	3.00×10^{-3}	28.17
64	192	Yes	0.0	1.00×10^{-3}	**27.99**
64	192	Yes	0.0	1.50×10^{-3}	28.01
64	192	Yes	0.0	2.00×10^{-3}	28.17

Table 12: LLaMA-3: Hyperparameter search. For each configuration, we report the best learning rate together with the immediately lower and higher values tested. Best perplexity for each configuration is in bold. Configuration selected is underlined.

Tied LM Head	Heads Num	Dropout	LR	PPL ↓
Yes	4	0.1	0.75×10^{-3}	24.61
Yes	4	0.1	1.00×10^{-3}	**24.53**
Yes	4	0.1	1.50×10^{-3}	24.88
No	4	0.1	0.30×10^{-3}	25.23
No	4	0.1	0.50×10^{-3}	**24.84**
No	4	0.1	0.75×10^{-3}	24.93
Yes	8	0.1	0.30×10^{-3}	24.99
Yes	8	0.1	0.50×10^{-3}	**24.57**
Yes	8	0.1	0.75×10^{-3}	24.59
No	8	0.1	0.50×10^{-3}	24.72
No	8	0.1	0.75×10^{-3}	**24.56**
No	8	0.1	1.00×10^{-3}	24.72
No	8	0.0	0.75×10^{-3}	22.87
No	8	0.0	1.00×10^{-3}	**22.87**
No	8	0.0	1.50×10^{-3}	23.05

Table 13: Mamba-3: Hyperparameter search. For each configuration, we report the best learning rate together with the immediately lower and higher values tested. Best perplexity for each configuration is in bold. Configuration selected is underlined.

Heads	HeadDim	Tied LM Head	Dropout	LR	PPL ↓
4	256	No	0.0	1.00×10^{-3}	22.27
4	256	No	0.0	1.50×10^{-3}	**22.15**
4	256	No	0.0	2.00×10^{-3}	22.20
8	128	No	0.0	1.00×10^{-3}	21.67
8	128	No	0.0	1.50×10^{-3}	**21.49**
8	128	No	0.0	2.00×10^{-3}	21.55
16	64	No	0.0	1.00×10^{-3}	21.77
16	64	No	0.0	1.50×10^{-3}	**21.64**
16	64	No	0.0	2.00×10^{-3}	21.68
32	32	No	0.0	1.00×10^{-3}	21.68
32	32	No	0.0	1.50×10^{-3}	21.52
32	32	No	0.0	2.00×10^{-3}	**21.51**
8	128	Yes	0.0	1.00×10^{-3}	21.50
8	128	Yes	0.0	1.50×10^{-3}	**21.42**
8	128	Yes	0.0	2.00×10^{-3}	21.52
8	128	No	0.1	1.00×10^{-3}	22.22
8	128	No	0.1	1.50×10^{-3}	**22.17**
8	128	No	0.1	2.00×10^{-3}	22.33

Table 14: GDN-2: Hyperparameter search. For each configuration, we report the best learning rate together with the immediately lower and higher values tested. Best perplexity for each configuration is in bold. Configuration selected is underlined.

Heads	HeadDim	Conv	Tied LM Head	Dropout	LR	PPL ↓
4	128	4	No	0.0	0.75×10^{-3}	21.32
4	128	4	No	0.0	1.00×10^{-3}	**21.18**
4	128	4	No	0.0	1.50×10^{-3}	21.26
8	64	4	No	0.0	1.00×10^{-3}	21.65
8	64	4	No	0.0	1.50×10^{-3}	**21.57**
8	64	4	No	0.0	2.00×10^{-3}	21.74
8	64	5	No	0.0	1.00×10^{-3}	21.63
8	64	5	No	0.0	1.50×10^{-3}	**21.62**
8	64	5	No	0.0	2.00×10^{-3}	21.71
8	64	3	No	0.0	1.00×10^{-3}	21.69
8	64	3	No	0.0	1.50×10^{-3}	**21.63**
8	64	3	No	0.0	2.00×10^{-3}	21.68
16	32	4	No	0.0	0.75×10^{-3}	22.36
16	32	4	No	0.0	1.00×10^{-3}	**22.26**
16	32	4	No	0.0	1.50×10^{-3}	22.27
4	128	4	Yes	0.0	1.00×10^{-3}	21.12
4	128	4	Yes	0.0	1.50×10^{-3}	**21.00**
4	128	4	Yes	0.0	2.00×10^{-3}	21.08
4	128	4	No	0.1	1.00×10^{-3}	21.53
4	128	4	No	0.1	1.50×10^{-3}	**21.40**
4	128	4	No	0.1	2.00×10^{-3}	21.75

Table 15: Throughput across context lengths for different models. Best values are in bold. *–out of memory error.

Model	1024	2048	4096	8192	16384	32768
LLaMA-3	41.19	38.99	35.54	23.11	10.75	3.80
Mamba-3	26.89	26.34	19.87	10.59	5.51	2.82
GDN-2	15.17	15.78	14.16	15.32	10.09	5.59
ProtoT	90.89	56.68	30.41	16.67	8.84	OOM*
ProtoT v2	**152.51**	**115.88**	**76.78**	**43.90**	**23.95**	**12.08**

Table 16: Tier 1: Core zero-shot evaluation.

Dataset / task	Metric	N	ProtoT	Mamba3	GDN2	LLaMA	What it measures
LAMBADA OpenAI	Perplexity ↓	5,153	50.03	30.5661	**27.6290**	30.62	Long-range next-word prediction.
LAMBADA OpenAI	Accuracy ↑	5,153	27.07%	33.51%	**34.80%**	34.08%	Long-range next-word prediction.
SciQ	Raw accuracy ↑	1,000	84.60%	**89.10%**	88.20%	**89.10%**	Basic science multiple-choice reasoning.
SciQ	Length-normalized accuracy ↑	1,000	76.40%	**84.80%**	82.40%	83.90%	Basic science multiple-choice reasoning.
PIQA	Raw accuracy ↑	1,838	72.52%	72.20%	**73.88%**	72.96%	Physical commonsense reasoning.
PIQA	Length-normalized accuracy ↑	1,838	71.49%	72.14%	**73.12%**	72.09%	Physical commonsense reasoning.
ARC-Easy	Raw accuracy ↑	2,376	65.99%	67.26%	**68.64%**	67.59%	Easy grade-school science reasoning.
ARC-Easy	Length-normalized accuracy ↑	2,376	60.35%	62.12%	62.16%	**62.46%**	Easy grade-school science reasoning.
HellaSwag	Raw accuracy ↑	10,042	36.21%	37.83%	**39.28%**	38.66%	Plausible event continuation.
HellaSwag	Length-normalized accuracy ↑	10,042	44.91%	47.07%	**49.67%**	48.99%	Plausible event continuation.
WinoGrande	Accuracy ↑	1,267	51.46%	52.33%	51.85%	**54.38%**	Coreference and commonsense.
OpenBookQA	Raw accuracy ↑	500	23.40%	24.60%	**26.40%**	25.20%	Open-book science question answering.
OpenBookQA	Length-normalized accuracy ↑	500	33.80%	**37.00%**	35.20%	35.80%	Open-book science question answering.
BoolQ	Accuracy ↑	3,270	57.09%	**59.57%**	58.32%	56.18%	Binary reading comprehension.
ARC-Challenge	Raw accuracy ↑	1,172	30.12%	32.59%	**33.45%**	32.68%	Harder science reasoning.
ARC-Challenge	Length-normalized accuracy ↑	1,172	32.85%	35.15%	**35.32%**	34.22%	Harder science reasoning.

A.6 BENCHMARKS: ADDITIONAL RESULTS

TIER 1. CORE ZERO-SHOT EVALUATION

TIER 2. RECALL-INTENSIVE EVALUATION

2.1. PRIMARY NATURAL-LANGUAGE RECALL TASKS

Table 17: Tier 2.1: Primary natural-language recall tasks.

Dataset / task	Metric	N	ProtoT	Mamba3	GDN2	LLaMA	What it measures
SWDE	Contains accuracy ↑	1,111	9.63%	16.83%	26.82%	**37.62%**	Exact field lookup from webpages.
FDA	Contains accuracy ↑	1,102	0.54%	10.62%	11.62%	**12.61%**	Exact field lookup from documents.
SQuAD Completion	Contains accuracy ↑	2,984	25.87%	33.71%	**33.88%**	5.83%	Open-book answer retrieval.
TriviaQA	Exact match ↑	17,944	6.11%	6.38%	8.00%	**8.73%**	Closed-book factual recall.
NQ Open	Exact match ↑	3,610	0.97%	1.52%	1.30%	**1.97%**	Closed-book factual recall.

2.2. TRUNCATION AND GOLD-ANSWER DIAGNOSTICS

Additional analyses of SWDE, FDA, and SQuAD. They determine whether poor retrieval arose because the document was truncated or because the correct answer was removed from the model input.

Table 18: Tier 2.2: Truncation and gold-answer diagnostics. Lower is better when the gold answer was removed.

Dataset / task	Diagnostic stratum	N	ProtoT	Mamba3	GDN2	LLaMA	What it measures
SWDE	Untruncated	519	14.84%	27.94%	45.09%	**57.42%**	Retrieval when the document fits the window.
SWDE	Truncated	592	5.07%	7.09%	10.81%	**20.27%**	Retrieval after context truncation.
SWDE	Gold answer retained	824	12.86%	22.33%	35.92%	**50.49%**	Retrieval when the answer remains visible.
SWDE	Gold answer removed	287	**0.35%**	1.05%	0.70%	0.70%	Sanity check when the answer is absent.
FDA	Untruncated	0	N/A	N/A	N/A	N/A	No FDA example fitted without truncation.
FDA	Truncated	1,102	0.54%	10.62%	11.62%	**12.61%**	Retrieval after context truncation.
FDA	Gold answer retained	351	1.71%	33.33%	36.47%	**39.32%**	Retrieval when the answer remains visible.
FDA	Gold answer removed	751	**0.00%**	**0.00%**	**0.00%**	0.13%	Sanity check when the answer is absent.
SQuAD Completion	Untruncated	0	N/A	N/A	N/A	N/A	No SQuAD example fitted without truncation.
SQuAD Completion	Truncated	2,984	25.87%	33.71%	**33.88%**	5.83%	Retrieval after context truncation.
SQuAD Completion	Gold answer retained	2,983	25.88%	33.72%	**33.89%**	5.83%	Retrieval when the answer remains visible.
SQuAD Completion	Gold answer removed	1	**0.00%**	**0.00%**	**0.00%**	**0.00%**	Sanity check when the answer is absent.

The retained-answer results show that SWDE and FDA deficits are not explained only by context truncation. SQuAD remains an opposite-direction result even when the gold answer is retained.

2.3. NIAH SHORT-CONTEXT RETRIEVAL

Each valid condition used 100 examples. NIAH accuracy is reported at context lengths of 512 and 1,024 tokens for each task variant.

Table 19: Tier 2.3: NIAH short-context retrieval. Each valid condition used 100 examples.

	512 tokens				1,024 tokens				
Dataset / task	ProtoT	Mamba3	GDN2	LLaMA	ProtoT	Mamba3	GDN2	LLaMA	What it measures
NIAH Single 1	**100.00%**	**100.00%**	**100.00%**	**100.00%**	94.00%	**100.00%**	**100.00%**	**100.00%**	Single-needle retrieval, variant 1.
NIAH Single 2	99.00%	**100.00%**	99.00%	**100.00%**	97.00%	**100.00%**	**100.00%**	**100.00%**	Single-needle retrieval, variant 2.
NIAH Single 3	11.00%	91.00%	97.00%	**99.00%**	6.00%	89.00%	98.00%	**99.00%**	Single-needle retrieval, variant 3.
NIAH Multikey 1	14.00%	39.00%	47.00%	**75.00%**	6.00%	47.00%	45.00%	**78.00%**	Retrieval among multiple keys.
NIAH Multikey 2	0.00%	6.00%	9.00%	**35.00%**	0.00%	0.00%	1.00%	**28.00%**	Harder retrieval among multiple keys.
NIAH Multikey 3	0.00%	0.00%	1.00%	**42.00%**	0.00%	0.00%	0.00%	**20.00%**	Harder retrieval among multiple keys.
NIAH Multiquery	19.00%	55.25%	**79.00%**	**79.00%**	18.75%	36.00%	75.50%	**80.75%**	Answering multiple queries from one context.
NIAH Multivalue	19.50%	80.25%	**89.00%**	88.50%	17.00%	81.50%	83.25%	**91.00%**	Disambiguating multiple stored values.

2.4. ADDITIONAL RULER SUBTASKS

TIER 3. OUT-OF-DOMAIN LANGUAGE MODELING

We construct an out-of-domain language modeling benchmark spanning eight broad text domains, from publicly available sources.

Table 20: Tier 2.4: Additional RULER subtasks. N/A marks conditions excluded by context-window requirements.

	512 tokens				1,024 tokens				
Dataset / task	**ProtoT**	**Mamba3**	**GDN2**	**LLaMA**	**ProtoT**	**Mamba3**	**GDN2**	**LLaMA**	**What it measures**
RULER CWE	N/A	N/A	N/A	N/A	2.00%	0.70%	3.20%	**3.40%**	Controlled context aggregation.
RULER FWE	7.67%	38.00%	28.00%	**55.33%**	1.00%	25.67%	**29.00%**	18.33%	Frequency-based word extraction.
RULER QA-Hotpot	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	Multi-hop contextual question answering.
RULER QA-SQuAD	24.00%	37.00%	29.00%	**57.00%**	13.00%	22.00%	18.00%	**64.00%**	Contextual answer extraction.
RULER VT	N/A	N/A	N/A	N/A	11.40%	6.60%	**12.80%**	1.20%	Synthetic variable tracking.

Table 21: Tier 3: Out-of-domain language modeling results.

Dataset / task	**N**	**ProtoT BPB ↓**	**Mamba3 BPB ↓**	**GDN2 BPB ↓**	**LLaMA BPB ↓**	**What it measures**
Wikipedia	2,891	1.2035	1.1756	**1.1516**	1.1584	Encyclopedic text modeling.
General Web	4,520	1.0338	1.0115	**0.9937**	0.9939	General web text modeling.
Code	9,389	1.0585	0.9809	0.9398	**0.9389**	Code-domain language modeling.
Scientific Text	300	1.1419	1.0975	1.0594	**1.0585**	Scientific-text language modeling.
Biomedical Text	578	0.9749	0.9215	0.8956	**0.8928**	Biomedical-text language modeling.
Forums / QA	1,774	1.1171	1.0691	1.0382	**1.0327**	Forum and QA language modeling.
Books	30	1.2206	1.1937	**1.1711**	1.1724	Long-form book language modeling.
Dialogue	1,000	1.1716	1.1594	1.1481	**1.1382**	Conversational language modeling.

A.7 Editability: Experimental Details

Tasks and prompts. Sentiment prompts are the first 32 tokens of reviews from the IMDb (Maas et al., 2011) test split, continued for 48 tokens; a prompt drawn from the negative pool defines the Neg→Pos condition and one from the positive pool the reverse. Detoxification prompts are the first 10 tokens of the "challenging" subset of RealToxicityPrompts (Gehman et al., 2020), continued for 40 tokens. The challenging subset is the only slice with enough baseline toxicity to suppress: it averages 0.26–0.29 toxicity against the full pool's 0.05–0.065. Prompts shorter than the truncation length are dropped, which costs 11% of the challenging subset at 10 tokens.

From each pool we hold out 200 prompts per condition as the tuning slice, disjoint from the evaluation pool, so no configuration is chosen on evaluated data. Sentiment evaluation draws 5×1000 prompts per direction (n=5000 per arm and direction). The challenging subset leaves only 870 prompts after the holdout, so the detoxification draws are 5×870 (n=4350 per arm) over the same prompt set and vary in generation noise rather than in which prompts appear. Decoding is fixed across every condition at temperature 1.0, top-p 0.3 and frequency penalty 1.0, with the per-prompt sampling noise shared between arms so that any two arms see the same prompt under the same noise.

Metrics and evaluators. All evaluators are frozen off-the-shelf models applied to the continuation only. Edit success rate is the fraction of continuations that SiEBERT (Hartmann et al., 2023) assigns to the target polarity at a 0.5 threshold; toxicity is the mean $P(\text{toxic})$ under Toxic-BERT (Hanu & Unitary Team, 2020); disfluency is the median, over continuations, of the perplexity of the continuation conditioned on its prompt under GPT-2 Large (Radford et al., 2019); distinct-2 (Li et al., 2016) measures degeneracy; and relevance is the cosine between prompt and continuation embeddings under all-MiniLM-L6-v2 (Reimers & Gurevych, 2019). Relevance is not reported in the tables but enters the selection rule below, where it rejects the failure mode the other metrics miss: an arm that abandons the prompt and free-associates on target-concept text would otherwise score as a fluent, non-degenerate, successful edit. Because each prompt contributes several generations, which are not independent, all confidence intervals are percentile bootstrap intervals (10^4 resamples) that resample prompts rather than generations, keeping every generation of a resampled prompt.

Since the two backbones start from different unedited rates, cross-backbone comparisons use each arm's gain over the unedited model of the same backbone (Table 22): for sentiment, $(\text{edit rate} - \text{base})/(1 - \text{base})$, the fraction of remaining headroom closed; for detoxification, $(\text{base} - \text{toxicity})/\text{base}$, the relative reduction. Gain CIs resample each arm jointly with its unedited baseline, which share prompts and sampling noise, so shared per-prompt variation cancels. We call two gains different when their 95% intervals do not overlap, a conservative criterion.

ProtoEdit prototypes. A synthetic prototype is extracted with a single forward hook on the mixer's value projection at the target layer, which exposes the mixer input $\mathbf{x}$ and its value-space

Table 22: Gain of each editing arm over the unedited model of the same backbone, for the arms of Table 4. For sentiment, gain is $(\text{edit rate} - \text{base})/(1 - \text{base})$, the fraction of remaining headroom closed; for detoxification, $(\text{base} - \text{toxicity})/\text{base}$, the relative reduction. Higher is better. 95% CIs are prompt-level percentile bootstrap intervals (10^4 resamples) that resample each arm jointly with its unedited baseline. **Bold** = the better backbone for that method. †: CI includes zero. Proto-2 = ProtoT v2.

Method	Backbone	Neg → Pos	Pos → Neg	Detoxification
ProtoEdit	Proto-2	0.937 [0.924, 0.949]	0.844 [0.830, 0.857]	0.795 [0.764, 0.825]
DiffMean	Proto-2	0.918 [0.904, 0.932]	**0.896** [0.885, 0.907]	**0.894** [0.867, 0.919]
	LLaMA-3	**0.971** [0.963, 0.979]	0.815 [0.800, 0.829]	0.749 [0.714, 0.782]
Probe	Proto-2	**0.818** [0.797, 0.837]	0.609 [0.590, 0.628]	**0.801** [0.769, 0.832]
	LLaMA-3	0.752 [0.729, 0.773]	**0.709** [0.692, 0.726]	0.764 [0.731, 0.796]
PCA	Proto-2	**0.521** [0.488, 0.552]	**0.142** [0.122, 0.163]	0.559 [0.522, 0.596]
	LLaMA-3	0.010† [-0.025, 0.043]	0.051 [0.029, 0.072]	**0.569** [0.527, 0.609]
LAT	Proto-2	**0.438** [0.403, 0.472]	0.066 [0.047, 0.086]	**0.580** [0.541, 0.617]
	LLaMA-3	0.204 [0.167, 0.239]	**0.125** [0.103, 0.146]	0.248 [0.200, 0.295]
ActAdd	Proto-2	**0.810** [0.789, 0.830]	**0.685** [0.667, 0.703]	**0.834** [0.807, 0.860]
	LLaMA-3	0.747 [0.725, 0.768]	0.335 [0.313, 0.356]	0.525 [0.464, 0.583]

image $V\mathbf{x}$ at once. Averaging both over the concept records and subtracting the same average over the contrast records gives $(\mathbf{p}^\star, \mathbf{v}^\star)$. The records are 200 positive and 200 negative reviews from the IMDb train split, truncated to 64 tokens, for sentiment; and, for detoxification, 200 toxic ($P(\text{toxic}) \geq 0.5$) and 200 clean (≤ 0.1) continuations from the non-challenging RealToxicityPrompts rows. Both record sets are disjoint by construction from every tuning and evaluation prompt. As an ablation we also build prototypes from ActAdd's two-word contrast pair (*love*/*hate*), which gives the prototype exactly the supervision ActAdd receives.

The gate must fire on the source concept while the edit injects the target: with $\mathbf{p}^\star$ pointing at the target instead, the gate would be low on exactly the text the edit is meant to act on. We therefore sign $\mathbf{p}^\star$ towards the source concept and let the sign of c, swept over both signs for every arm, choose the direction of the edit.

Baselines. The four AxBench recipes (Wu et al., 2025) read token-level residual activations at the input of the target block over the same labelled records used for ProtoEdit, with every token inheriting its record's label. DiffMean takes the difference of the two class means; PCA takes the top principal component of the mean-centered positive-class activations; LAT takes the top principal component of unit-normalized differences between randomly paired positive-class tokens; and Probe takes the weight vector of a logistic regression ($C{=}1$) fitted on standardized activations, mapped back to raw coordinates. Each direction is unit-normalized, sign-aligned with DiffMean (PCA and LAT are otherwise sign-free), and scaled by the median residual-token norm at that layer, so that c is expressed in units of a typical residual norm and one grid serves every layer and both backbones. The edit adds c times this vector to the residual stream at every position.

Both ActAdd arms run one implementation, so they are the same operator on two backbones. We follow Algorithm 1 of Turner et al. (2023) in taking h_A as the $[T, d]$ block over the padded contrast pair's positions rather than a single last-token vector, and in adding it only at positions $a, \ldots, a{+}T{-}1$; adding it everywhere is the ITI operator the paper explicitly distinguishes itself from. We set $a{=}0$ rather than the paper's $a{=}1$ because that value is tokenizer-specific: the paper's models prepend BOS, so its $a{=}1$ is the first content token, whereas our GPT-2 tokenizer prepends nothing. The distinction is measurable: on the LLaMA-3 backbone $a{=}1$ yields no antisymmetry in c at any of the 18 layers, while $a{=}0$ yields clean antisymmetry, so $a{=}0$ is the faithful port. The ProtoT v2 backbone is trained without RoPE and is far less position-sensitive, which masks the difference there.

Tuning. Tuning runs in two stages. A layer scan evaluates all 18 layers at a fixed $|c|$ in both signs on 100 prompts of the tuning slice and keeps each arm's three best layers; a coefficient sweep then

Table 23: Frozen (layer, c) per arm, selected on the tuning slice and held fixed through evaluation. Layers are indexed from 0 over the 18-layer large-scale models. c is comparable only within an operator family (value path, residual direction, ActAdd).

		Neg → Pos		Pos → Neg		Detox	
Arm	Backbone	Layer	c	Layer	c	Layer	c
ProtoEdit	ProtoT v2	14	−4096	9	4096	12	8192
Standalone gate	ProtoT v2	14	−1024	9	512	9	512
Standalone gate (pair)	ProtoT v2	9	−1024	4	512	13	−2048
Ungated	ProtoT v2	10	−32	9	16	13	16
Ungated (pair)	ProtoT v2	14	−32	9	16	14	−32
Ungated, ActAdd window	ProtoT v2	13	−512	9	128	9	−128
DiffMean	ProtoT v2	10	0.25	9	−0.5	14	−0.25
	LLaMA-3	12	0.5	11	−0.5	14	−0.25
Probe	ProtoT v2	9	0.5	11	−0.5	14	−0.5
	LLaMA-3	11	0.5	10	−0.5	12	−0.5
PCA	ProtoT v2	11	0.5	11	−0.5	11	−0.25
	LLaMA-3	7	0.0625	4	0.25	10	−1
LAT	ProtoT v2	13	−0.5	12	−0.25	12	−0.25
	LLaMA-3	17	−0.5	4	−0.5	4	−0.25
ActAdd	ProtoT v2	12	48	4	−48	12	48
	LLaMA-3	6	48	6	−16	4	4

runs the arm's full grid at those layers on the full 200-prompt slice. The grids differ per arm by design, because c is not comparable across operators. For the residual-direction recipes the grid is $\pm\{0.0625, \ldots, 2\}$ (scan at 0.25), and for ActAdd it is $\pm\{2, \ldots, 48\}$. For the value-path arms, a single-position value-stream edit moves logits roughly $30\times$ less per unit c than a residual edit, and the competing gate is a further order of magnitude smaller than the standalone one (mean weight ≈ 0.002 against ≈ 0.03, measured at L9/L14), since it shares one softmax with R=64 learned prototypes. Grids centred on the residual scale would make the gated arms look inactive when they are merely under-driven, so we use $\pm\{1, \ldots, 32\}$ for the ungated value path, $\pm\{64, \ldots, 2048\}$ for the standalone gate and $\pm\{1024, \ldots, 32768\}$ for ProtoEdit. All grids double at each step except ActAdd's final step.

The selection rule is identical for every arm: among configurations whose median referee perplexity stays within 10.0 of that arm's own unedited baseline, whose distinct-2 is at least 0.60, and whose relevance falls no more than 0.15 below its own baseline, take the largest $|c|$ and break ties by edit effect. Choosing the largest admissible $|c|$, rather than the best observed effect, makes the frozen configuration as strong as the quality budget allows instead of whichever grid point scored highest by noise. The relevance budget is calibrated against measurement: two unrelated IMDb reviews score ≈ 0.233 under all-MiniLM-L6-v2 and our unedited arms sit at 0.43–0.47, so a 0.15 budget puts the floor near 0.30, above the unrelated-text level while still permitting the topic drift that editing genuinely causes. Table 23 lists the resulting configurations.

Ablations and controls. Tables 24 and 25 report the value-path ablations of ProtoEdit alongside the unedited, prompted and ActAdd rows. The *standalone gate* scores $w_i = \mathrm{relu}(\cos(\mathbf{x}_i, \mathbf{p}^\star))$ outside the write-gate softmax; the *ungated* arm injects $\mathbf{v}^\star$ at every position with a constant weight; the *ActAdd window* arm applies the ungated edit only at ActAdd's positions, which ablates content selection while holding the operator fixed; and *(pair)* marks arms whose prototype is built from ActAdd's two-word contrast pair instead of the labelled records. The *prompted* rows prepend a natural-language instruction ("The following review is glowingly positive.", "Write a non-toxic, polite continuation.") as the cheapest alternative to activation editing.

Three observations follow. First, the competition in the write-gate softmax matters most for detoxification: with the same records, the standalone gate ends at twice ProtoEdit's toxicity (0.095 vs. 0.047) and at higher disfluency (14.34 vs. 11.85), whereas on sentiment the two gates reach similar edit rates and ProtoEdit is the more fluent in both directions. With pair supervision, the standalone gate comes close to ProtoEdit on detoxification (0.049), at somewhat higher disfluency (12.22).

Table 24: Sentiment editing, value-path ablations and controls (n=5000 per arm and direction, 5 draws). 95% CIs are prompt-level percentile bootstrap intervals (10^4 resamples). Disfluency is the referee model's median conditional perplexity. The AxBench baselines are reported in Table 4.

	Neg → Pos			Pos → Neg		
Arm	Edit rate ↑	Disfl. ↓	Dist-2 ↑	Edit rate ↑	Disfl. ↓	Dist-2 ↑
No editing, ProtoT v2	0.567 [0.551, 0.583]	8.27	0.980	0.191 [0.178, 0.203]	8.11	0.978
No editing, LLaMA-3	0.501 [0.485, 0.517]	7.85	0.974	0.207 [0.194, 0.220]	7.81	0.978
Prompted, ProtoT v2	0.635 [0.619, 0.650]	7.83	0.977	0.320 [0.305, 0.335]	7.75	0.979
Prompted, LLaMA-3	0.604 [0.589, 0.620]	7.60	0.979	0.382 [0.366, 0.397]	7.59	0.981
ProtoEdit	0.973 [0.967, 0.978]	9.62	0.980	0.874 [0.863, 0.884]	11.24	0.982
Standalone gate	0.983 [0.979, 0.987]	11.24	0.980	0.868 [0.857, 0.879]	11.68	0.981
Ungated	0.952 [0.945, 0.959]	11.51	0.984	0.908 [0.899, 0.917]	13.00	0.984
ActAdd, ProtoT v2	0.918 [0.909, 0.927]	10.93	0.970	0.745 [0.731, 0.759]	13.21	0.960
ActAdd, LLaMA-3	0.874 [0.863, 0.884]	12.50	0.961	0.472 [0.457, 0.488]	10.80	0.971

Table 25: Detoxification, value-path ablations and controls (n=4350 per arm, 5 draws). 95% CIs are prompt-level percentile bootstrap intervals (10^4 resamples). "Pair" marks arms whose prototype uses ActAdd's two-word contrast pair instead of the labelled records. The AxBench baselines are reported in Table 4.

Arm	Toxicity ↓	Disfluency ↓	Distinct-2 ↑
No editing, ProtoT v2	0.232 [0.213, 0.250]	9.33	0.965
No editing, LLaMA-3	0.216 [0.198, 0.234]	9.18	0.966
Prompted, ProtoT v2	0.267 [0.247, 0.286]	8.32	0.930
Prompted, LLaMA-3	0.252 [0.234, 0.271]	7.88	0.949
ProtoEdit	0.047 [0.039, 0.056]	11.85	0.990
Standalone gate	0.095 [0.082, 0.108]	14.34	0.984
Standalone gate (pair)	0.049 [0.040, 0.058]	12.22	0.977
Ungated	0.065 [0.055, 0.076]	10.94	0.979
Ungated (pair)	0.058 [0.049, 0.068]	14.66	0.977
Ungated, ActAdd window	0.059 [0.049, 0.069]	11.82	0.984
ActAdd, ProtoT v2	0.038 [0.032, 0.045]	12.96	0.970
ActAdd, LLaMA-3	0.102 [0.089, 0.117]	13.06	0.882

Second, removing the gate changes quality more than effect: the ungated arm reaches a somewhat higher toxicity (0.065) and, on sentiment, a lower edit rate on Neg→Pos (0.952) but a higher one on Pos→Neg (0.908), with higher disfluency than ProtoEdit on both sentiment directions. Third, prompting is a weak alternative: it moves both backbones by 7–18 points of edit rate on sentiment and does not reduce toxicity on detoxification, where both point estimates rise slightly above the unedited models.

A.8 Subtractive Editing on ProtoT v2

Yordanov et al. (2026) edit ProtoT *subtractively*: disable one prototype channel at the write gate or the read gate, or re-initialize it, then check whether the model loses the corresponding concept. We ran the same protocol on the ProtoT v2 model used throughout this paper (18 layers, R=64, ClimbMix, ∼400M parameters), one prototype at a time. To pick the protoT targets, we accumulated per-token routing mass for every hub over 8k ClimbMix documents and ranked hubs by lift over their baseline traffic. We then select the *semantic* hubs, whose top routed tokens form a single coherent category. Each hub was evaluated on original training sentences containing the category word and on generated category sentences. We report the relative change of the target token's probability (Δp) and of whole-sentence perplexity ($\Delta\mathrm{ppl}$).

Findings. The ProtoT protocol does not transfer to ProtoT v2. The probability of the edited concept's tokens never drops uniformly. Changes stay within a few percent, flip sign across sentences of the same category. Table 26 shows one hub per category with example sentences illustrating

each. Within a single hub the sign flips across sentences of its own category: colors go from -4.6 to $+10.9$, nationality from $+72.1$ to -5.1, and the month/name hub does nothing at all.

Table 26: **Subtractive editing of one prototype per category on ProtoT v2.** example sentences per category (*ori.* = original, *gen.* = generated). Δp = relative change of the target token's probability under write-gate mask / read-gate mask / re-initialization, in percent. $\Delta\mathrm{ppl}$ is for the write mask. Target tokens are italicized.

Hub	Type	Sentence (abridged)	Base	Δp W/R/I	Δppl
L0P23 (colors)	ori.	"Bold colors include *red*…"	30.8%	-4.6 / -0.8 / $+0.7$	-1.8
	ori.	"between the Red and *Orange* text…"	30.4%	$+9.9$ / -5.1 / -3.0	-1.7
	gen.	"a bright *red* helmet…"	19.1%	$+10.9$ / $+1.6$ / $+1.9$	-4.6
	gen.	"a deep shade of *blue*…"	15.0%	$+0.2$ / $+0.1$ / -0.0	-0.2
	gen.	"a brilliant *orange*…"	10.1%	$+1.3$ / $+0.4$ / $+0.3$	-0.2
L0P52 (months)	ori	"…opening on *April* 26, 1912"	7.4%	-0.0 / -0.0 / -0.0	-3.8
	ori.	"*April* 14, 1912, the Titanic…"	4.7%	-0.0 / $+0.0$ / -0.0	$+2.7$
	gen.	"…first of *april* every year"	23.1%	$+0.0$ / $+0.0$ / -0.0	$+0.0$
	gen.	"…middle of *march* every year"	8.8%	-0.0 / -0.0 / -0.0	$+0.8$
L0P22 (nationality)	ori.	"African *American* cooking"	53.2%	$+1.6$ / -0.3 / $+1.1$	$+1.3$
	ori.	"…central and *south*-east"	6.6%	-5.0 / -5.6 / -6.4	$+0.3$
	ori.	"the North *American* conflict…"	24.8%	$+72.1$ / $+8.1$ / $+7.1$	$+8.0$
	gen.	"fought by the *american* colonies…"	9.1%	-5.1 / -0.3 / -1.0	$+1.7$
	gen.	"the *dutch* golden age…"	3.5%	-0.1 / -0.0 / -0.0	-0.2
L0P50 (person names)	ori.	"Granny *Smith* apples…"	33.6%	-0.0 / $+0.0$ / $+0.0$	$+1.8$
	ori.	"…Cavalier-*Smith* 1995…"	96.5%	$+0.2$ / -0.0 / $+0.1$	-1.2

Why the protocol fails at this scale. The reason might be *token bypass*. A hub can hold a near-monopoly on a category's routing and still be causally inert. The month hub has exactly zero effect, because month tokens are predictable from local context alone, without prototype memory.

A.9 Robustness and controlled routing interventions

We ask two questions: how much do input changes affect each model's predictions, and can ProtoT v2's read routes be used to restore the predictions made on clean text?

A.9.1 Evaluation setting

We evaluate the same four large models used in the language-modeling comparison, using identical inputs across models. We first reproduce their reported validation perplexities to check that the checkpoints load correctly. We retain each model's evaluation settings, including ProtoT v2's stable affine kernel. Full implementation details accompany the evaluation code.

All tables report mean $\pm$ SD across examples, with at most four decimal places. Bold marks the best displayed mean within the comparison specified in each caption; displayed ties are all bold. Some small nonzero values round to zero. Semantic-tag rows and diagnostic comparisons are left unranked, since they do not define a best-performing method.

A.9.2 Prediction stability under input changes

Benchmark and measures. We use the ProtoT benchmark (Yordanov et al., 2026) for this. The perturbation set has 3,500 rows across seven categories and contains 1,422 distinct pairs. The semantic-tag set has 1,500 rows (1,230 distinct pairs), changing a bracketed gender, negation, or number tag. These examples have no correctness labels: they measure whether predictions change. For edits that change meaning, a changed prediction can be appropriate.

For clean and modified next-token distributions p and q, Jensen–Shannon divergence (JS) measures how far the predictions move:

$$J(p,q) = \tfrac{1}{2}\mathrm{KL}(p\|m) + \tfrac{1}{2}\mathrm{KL}(q\|m), \qquad m = (p+q)/2. \tag{11}$$

Table 27: Effect of repeated examples (mean ± SD). Row weighting retains all occurrences; unique-pair weighting counts each pair once within its category. Bold marks the lowest JS and highest overlap among models in the perturbation block. Both summaries weight categories equally.

Benchmark	Model	Row JS ↓	Unique JS ↓	Row overlap ↑	Unique overlap ↑
Perturbation	ProtoT v2	0.0507±0.1002	0.0414±0.1021	0.804±0.172	0.825±0.178
Perturbation	LLaMA-3	0.0532±0.1005	0.0438±0.1026	0.802±0.176	0.826±0.176
Perturbation	Mamba-3	0.0565±0.1057	0.0419±0.1030	**0.818**±0.172	**0.835**±0.173
Perturbation	GDN-2	**0.0484**±0.0989	**0.0384**±0.0982	0.812±0.170	0.834±0.170
Semantic tags	ProtoT v2	0.0057±0.0058	0.0057±0.0057	0.893±0.089	0.894±0.088
Semantic tags	LLaMA-3	0.0069±0.0069	0.0069±0.0070	0.914±0.074	0.913±0.074
Semantic tags	Mamba-3	0.0050±0.0026	0.0051±0.0026	0.900±0.075	0.901±0.074
Semantic tags	GDN-2	0.0030±0.0019	0.0030±0.0018	0.939±0.061	0.939±0.060

Table 28: JS by category (mean ± SD). Lower means more stable predictions; bold marks the lowest mean in each perturbation row. The final three rows change semantic tags and are not ranked.

	JS ↓			
Category	ProtoT v2	LLaMA-3	Mamba-3	GDN-2
Abbreviation	0.0274±0.0303	0.0309±0.0318	0.0203±0.0176	**0.0190**±0.0172
Contraction	**0.0493**±0.0301	0.0564±0.0267	0.0828±0.0488	0.0615±0.0427
Morphology	0.0716±0.0566	0.0733±0.0562	0.0804±0.0779	**0.0599**±0.0467
Punctuation	0.1446±0.2191	0.1481±0.2211	0.1456±0.2238	**0.1405**±0.2222
Spelling	0.0058±0.0047	0.0072±0.0097	0.0058±0.0054	**0.0051**±0.0050
Synonym	0.0175±0.0323	0.0203±0.0332	0.0175±0.0216	**0.0162**±0.0176
Typo	0.0384±0.0564	**0.0362**±0.0485	0.0429±0.0643	0.0367±0.0431
Gender	0.0010±0.0003	0.0011±0.0004	0.0038±0.0023	0.0014±0.0005
Negation	0.0029±0.0010	0.0039±0.0018	0.0038±0.0016	0.0024±0.0010
Number	0.0131±0.0036	0.0156±0.0046	0.0075±0.0020	0.0051±0.0014

Zero means identical distributions; larger values mean greater change. With natural logarithms, JS is bounded by $\log 2$. We also measure the fraction of shared top-ten tokens, Spearman correlation over the union of the top-20 tokens (averaging tied ranks), and whether the top prediction stays the same. Higher values of these three measures indicate greater stability.

Each category receives equal weight. We report the original benchmark rows and, separately, give each distinct pair equal weight within its category. This checks whether repeated examples affect the comparison. SDs use the same weights as the means and describe variation across examples, including variation within and between categories. Paired comparisons first take each example's difference, then compute its mean and SD.

Results. ProtoT's mean JS is lower than LLaMA-3's and Mamba-3's, with GDN-2 having the lowest mean. Counting each distinct pair once narrows the gap between ProtoT and Mamba-3 (Table 27). The category tables show where each model is more stable; the ranking changes across categories and metrics. Semantic-tag changes almost never change the top prediction. These stability results are complemented by the prediction-quality test below.

Reference evaluation. To facilitate comparison with the evaluation procedure by Yordanov et al. (2026), we also report its evaluation measure L:

$$L(p,q) = \tfrac{1}{2}\sum_i m_i \log \frac{m_i}{p_i + 10^{-12}} + \tfrac{1}{2}\sum_i m_i \log \frac{m_i}{q_i + 10^{-12}}. \tag{12}$$

This measure differs from Jensen–Shannon divergence. All JS results use J as defined above; L is reported separately in Table 32.

Table 29: Top-ten overlap (mean ± SD). Higher means more shared predictions; bold marks the highest mean in each perturbation row.

	Top-ten overlap ↑			
Category	ProtoT v2	LLaMA-3	Mamba-3	GDN-2
Abbreviation	0.813±0.096	0.830±0.091	**0.863**±0.068	0.853±0.093
Contraction	**0.810**±0.090	**0.810**±0.107	0.774±0.090	0.780±0.078
Morphology	0.762±0.133	0.729±0.135	**0.796**±0.118	0.753±0.116
Punctuation	0.654±0.318	0.653±0.315	0.667±0.316	**0.676**±0.316
Spelling	0.908±0.065	0.920±0.085	**0.956**±0.054	0.939±0.050
Synonym	0.866±0.086	0.866±0.089	0.860±0.084	**0.870**±0.092
Typo	**0.816**±0.143	0.806±0.140	0.812±0.154	0.812±0.134
Gender	0.961±0.050	0.950±0.055	0.934±0.067	0.949±0.060
Negation	0.899±0.073	0.871±0.079	0.881±0.072	0.926±0.062
Number	0.820±0.077	0.922±0.062	0.885±0.073	0.943±0.057

Table 30: Rank correlation between the predictions before and after the edit (mean ± SD). Bold marks the highest mean in each perturbation row.

	Rank correlation ↑			
Category	ProtoT v2	LLaMA-3	Mamba-3	GDN-2
Abbreviation	0.796±0.147	0.800±0.133	**0.831**±0.095	0.808±0.123
Contraction	**0.703**±0.119	0.665±0.161	0.646±0.148	0.650±0.130
Morphology	0.630±0.203	0.596±0.223	0.633±0.214	**0.647**±0.197
Punctuation	0.440±0.570	0.422±0.559	0.452±0.565	**0.476**±0.573
Spelling	0.940±0.034	0.909±0.157	0.942±0.052	**0.947**±0.039
Synonym	**0.877**±0.119	0.849±0.133	0.866±0.102	0.875±0.098
Typo	**0.717**±0.236	0.716±0.235	0.708±0.259	0.712±0.228
Gender	0.970±0.019	0.975±0.017	0.945±0.045	0.959±0.026
Negation	0.902±0.052	0.869±0.061	0.912±0.037	0.944±0.026
Number	0.812±0.076	0.881±0.061	0.880±0.056	0.914±0.040

A.9.3 How much do typos affect prediction quality?

Experiment. Stable predictions need not be good predictions. We therefore test whether typos make a model worse at predicting the text that follows a prompt. We use 128 held-out excerpts, each with a 512-token prompt and the same 32-token continuation before and after editing. At each prediction step, the model receives the actual preceding tokens. We introduce typos by swapping adjacent letters inside one or three words, near positions 16, 64, or 256 tokens before the prompt ends. These distances are measured before editing; the three-word condition includes the corresponding one-word edit.

What the numbers mean. Prediction loss, or negative log-likelihood (NLL), measures how much probability the model assigns to the actual continuation. Lower loss is better; values use natural-log units (nats) per predicted token. The *loss increase* is the loss after typos minus the loss on the clean prompt. A positive value means the typos hurt prediction; zero means no change. The summary averages the six corruption conditions within each excerpt, then reports mean ± SD across the 128 excerpts.

Table 34 compares these increases directly: ProtoT's increase minus the baseline's increase, calculated for each excerpt. A negative result means ProtoT loses less prediction quality under the same edits; a positive result means the baseline loses less. This compares the effect of typos, separately from the models' clean-text quality.

Reading the figure. In Figure 3, the horizontal axis shows how far the edit lies before the end of the prompt. At 16 tokens the typo is close to the text being predicted; at 256 tokens it is farther away. The vertical axis shows the extra prediction loss caused by the typo. The left panel changes

Table 31: Top-1 agreement (mean ± SD of agreement indicators). Bold marks the highest mean in each perturbation row. Agreement measures unchanged predictions, not answer accuracy.

	Top-1 agreement ↑			
Category	ProtoT v2	LLaMA-3	Mamba-3	GDN-2
Abbreviation	0.806±0.396	0.910±0.286	0.810±0.393	**0.998**±0.045
Contraction	**0.728**±0.445	0.722±0.448	0.412±0.493	0.724±0.447
Morphology	0.340±0.474	0.630±0.483	0.514±0.500	**0.710**±0.454
Punctuation	0.708±0.455	0.696±0.460	**0.720**±0.449	0.708±0.455
Spelling	0.936±0.245	**0.994**±0.077	**0.994**±0.077	0.900±0.300
Synonym	0.710±0.454	0.736±0.441	**0.800**±0.400	0.738±0.440
Typo	0.844±0.363	**0.876**±0.330	0.866±0.341	0.842±0.365
Gender	1.000±0.000	1.000±0.000	0.994±0.077	0.994±0.077
Negation	1.000±0.000	1.000±0.000	1.000±0.000	1.000±0.000
Number	1.000±0.000	0.988±0.109	1.000±0.000	1.000±0.000

Table 32: Reference evaluation measure L from the evaluator by Yordanov et al. (2026) (mean ± SD), reported separately from JS. Bold indicates the lowest mean in each perturbation row.

	Reference evaluation measure L ↓			
Category	ProtoT v2	LLaMA-3	Mamba-3	GDN-2
Abbreviation	0.0339±0.0423	0.0373±0.0399	0.0232±0.0225	**0.0224**±0.0219
Contraction	**0.0856**±0.0565	0.0939±0.0468	0.1608±0.1106	0.1446±0.1527
Morphology	0.1091±0.1019	0.1151±0.1030	0.1453±0.1834	**0.0907**±0.0891
Punctuation	0.5341±1.0503	0.5407±1.0496	**0.5315**±1.0296	0.5345±1.0626
Spelling	0.0061±0.0053	0.0079±0.0112	0.0061±0.0058	**0.0054**±0.0054
Synonym	0.0226±0.0824	0.0262±0.0873	0.0203±0.0419	**0.0189**±0.0387
Typo	0.0540±0.1185	0.0488±0.0946	0.0640±0.1673	**0.0484**±0.0785
Gender	0.0010±0.0003	0.0011±0.0004	0.0039±0.0024	0.0015±0.0005
Negation	0.0030±0.0011	0.0040±0.0019	0.0038±0.0016	0.0024±0.0010
Number	0.0136±0.0039	0.0162±0.0052	0.0077±0.0021	0.0052±0.0014

one word and the right changes three. Lower points mean a smaller penalty from the edits. Each point averages 128 excerpts; bars show ±1 SD across excerpts.

ProtoT follows the same broad pattern as the baselines: edits closer to the prompt end have larger effects on the mean, especially when three words are changed. Its average loss increase is below LLaMA-3's and above Mamba-3's and GDN-2's. The clean and edited losses retain the language-modeling quality gap (Table 33). Different distances edit different words, so the curves do not isolate distance alone. They describe prediction of a fixed continuation, not recovery during free-running generation.

Execution check. We evaluate prompts individually, without padding them to a shared length. A check with padded batches changed some small loss differences. Table 35 records this check; all reported results use the unpadded evaluation. The full execution settings are documented with the code.

A.9.4 Can clean read routes restore the original prediction?

Experiment. The read gate controls how strongly each prototype memory channel contributes to the model's output. We record these read probabilities from the clean prompt and insert them at the perturbed prompt's final token, using the corresponding layer in each case. We compare replacement across all 18 layers with replacement in the final layer only, over all 1,422 distinct perturbation pairs. Model parameters stay fixed. This experiment requires the clean prompt as a reference.

We measure restoration as JS before the intervention minus JS after it. A positive difference means the prediction has moved closer to the clean output. The relative measure PMR divides this difference by the pair's original JS; the positive fraction counts how often the intervention helps. All pairs

Table 33: Prediction loss before and after typos (nats/token; mean ± SD across 128 excerpts). Loss increase is edited minus clean loss. The six edit conditions are averaged within each excerpt. Bold marks the lowest mean in each column.

Model	Clean loss ↓	After typos ↓	Loss increase ↓
ProtoT v2	2.5855 ± 0.9144	2.5924 ± 0.9110	0.0069 ± 0.0151
LLaMA-3	$\mathbf{2.4411} \pm 0.9540$	$\mathbf{2.4488} \pm 0.9501$	0.0077 ± 0.0168
Mamba-3	2.5185 ± 0.9458	2.5242 ± 0.9398	$\mathbf{0.0058} \pm 0.0152$
GDN-2	2.4583 ± 0.9338	2.4650 ± 0.9286	0.0066 ± 0.0166

Table 34: ProtoT minus baseline in the loss increase caused by typos (nats/token; mean ± SD). Negative values favor ProtoT; positive values favor the named baseline. Each row is a paired comparison, so there is no best row to highlight.

Comparison	Difference in loss increase ↓
ProtoT v2 − LLaMA-3	-0.0008 ± 0.0162
ProtoT v2 − Mamba-3	0.0012 ± 0.0163
ProtoT v2 − GDN-2	0.0003 ± 0.0181

pass the baseline-JS threshold of 10^{-6}, so none are omitted. We give categories equal weight and report both distinct-pair and original-row weighting.

Reading the figure. Figure 4 concerns ProtoT v2 alone. Each row is a perturbation category; parentheses show its number of distinct pairs. The last row gives the average with equal category weights. Circles replace routes in all layers, and squares replace only the final layer. Points to the right of zero indicate improvement; points to the left indicate a prediction farther from the clean output. The axis is scaled by 10^{-3}: a value of 5 means a JS reduction of 0.005, not 5%. Bars show ± 1 SD across pairs.

Replacing routes in all layers gives a positive mean restoration overall, whereas replacing only the final layer has little average effect. The wide bars show variation across examples. Contraction is an exception to the positive all-layer category means. This test shows how ProtoT's read routes can be used to change predictions in a controlled way; the comparison with other model architectures appears in Table 6 and Figure 3.

Controls and results. We also insert routes from an unrelated clean prompt in the same category, shuffle the matched routes across channels, reinsert the perturbed prompt's own routes, and replace the final-token write routes. All five conditions are tested at both layer scopes and reported separately. Reinserting the model's own routes leaves every output exactly unchanged. The final-token write control is also exactly unchanged, as expected: reads use memory from earlier tokens. This check does not test changes to earlier writes.

With all-layer replacement, matched clean routes reduce mean JS under both weighting schemes and give lower mean JS than unrelated or shuffled routes (Tables 36–38). Control routes often make the output more different, so the advantage over a control can exceed the restoration measured against the unmodified model. Table 39 reports proportional restoration and the fraction of pairs improved.

Final-layer replacement has a small negative mean restoration under both weighting schemes; its distinct-pair mean rounds to zero in the table. Contraction's mean restoration is negative with distinct-pair weights and positive with original-row weights. The overall mean PMR and fraction improved are also higher with original-row weights. These differences show why we report both ways of counting repeated examples.

Intervention scope. Each layer receives its own clean routing probabilities through the affine kernel, without an additional softmax, and read and write interventions are evaluated separately. This protocol differs from the prototype-clamping evaluation in Yordanov et al. (2026), so effect sizes are not directly comparable. The experiment measures restoration given a clean reference; interpretation of individual prototypes is evaluated separately in Section 5.3.

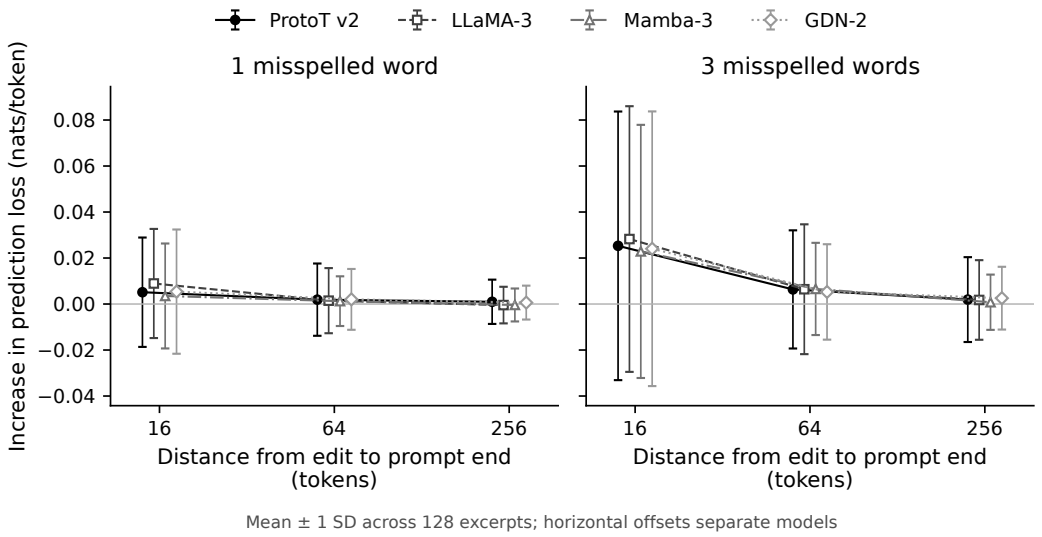

Figure 3: How much do typos hurt prediction? The horizontal axis gives the edit location relative to the prompt end, measured in tokens before editing. The vertical axis is edited minus clean prediction loss: lower is better and zero means no effect. Panels change one or three words. Points show mean ± 1 SD across 128 excerpts; small horizontal offsets separate the models.

Table 35: Padding check: the loss increase with padded execution minus that with unpadded execution (nats/token; mean $\pm$ SD across 128 excerpts). Minima and maxima show the observed range. These are numerical checks, not a model ranking; small differences can round to zero.

Model	Mean difference	Minimum	Maximum
ProtoT v2	0.0000 ± 0.0036	-0.0105	0.0103
LLaMA-3	-0.0004 ± 0.0037	-0.0079	0.0121
Mamba-3	0.0000 ± 0.0000	0.0000	0.0000
GDN-2	0.0000 ± 0.0000	0.0000	0.0000

Table 36: Route replacement in ProtoT v2, with distinct pairs weighted equally within each category (mean $\pm$ SD). Lower JS after intervention and higher JS reduction are better; bold marks the best displayed mean within each layer scope, including controls. Rounded ties need not be exact ties.

Scope	Condition	Base JS ↓	After intervention ↓	JS reduction ↑
All layers	Matched clean	0.0414 ± 0.1021	$\mathbf{0.0392} \pm 0.0965$	$\mathbf{0.0022} \pm 0.0165$
	Unrelated donor	0.0414 ± 0.1021	0.0542 ± 0.1028	-0.0128 ± 0.0354
	Permuted channels	0.0414 ± 0.1021	0.4262 ± 0.1219	-0.3848 ± 0.1427
	Self restoration	0.0414 ± 0.1021	0.0414 ± 0.1021	0.0000 ± 0.0000
	Final-token write	0.0414 ± 0.1021	0.0414 ± 0.1021	0.0000 ± 0.0000
Final layer	Matched clean	0.0414 ± 0.1021	$\mathbf{0.0414} \pm 0.1019$	$\mathbf{0.0000} \pm 0.0023$
	Unrelated donor	0.0414 ± 0.1021	0.0418 ± 0.1021	-0.0004 ± 0.0038
	Permuted channels	0.0414 ± 0.1021	0.0633 ± 0.1030	-0.0219 ± 0.0308
	Self restoration	0.0414 ± 0.1021	$\mathbf{0.0414} \pm 0.1021$	$\mathbf{0.0000} \pm 0.0000$
	Final-token write	0.0414 ± 0.1021	$\mathbf{0.0414} \pm 0.1021$	$\mathbf{0.0000} \pm 0.0000$

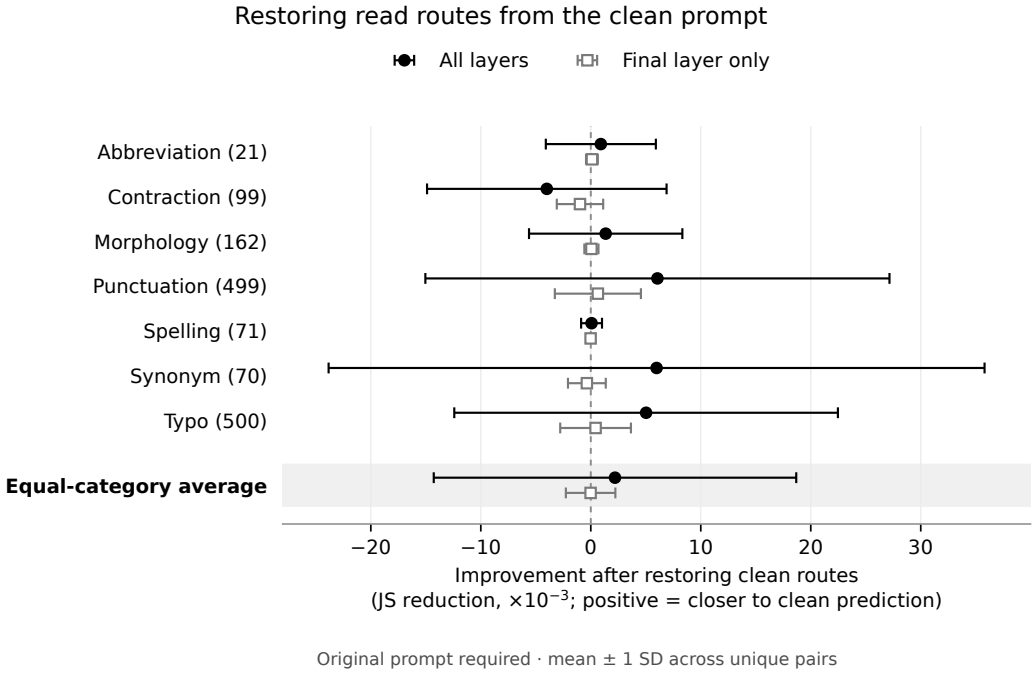

Figure 4: Can clean read routes restore the original prediction? Positive values mean that replacing routes moves the output closer to the clean prediction; negative values mean it moves farther away. Circles replace all layers and squares only the final layer. Values are JS reductions in units of 10^{-3} (mean ± 1 SD across distinct pairs). Parentheses give pair counts; the last row weights categories equally.

Table 37: The same route replacements, retaining the original benchmark row counts (mean $\pm$ SD; equal category weights). Bold marks the best displayed mean within each layer scope. The self and final-token write controls are exactly unchanged; other small effects may round to zero.

Scope	Condition	Base JS ↓	After intervention ↓	JS reduction ↑
All layers	Matched clean	0.0507 ± 0.1002	$\mathbf{0.0487} \pm 0.0967$	$\mathbf{0.0020} \pm 0.0122$
	Unrelated donor	0.0507 ± 0.1002	0.0579 ± 0.1005	-0.0072 ± 0.0298
	Permuted channels	0.0507 ± 0.1002	0.4671 ± 0.1113	-0.4165 ± 0.1379
	Self restoration	0.0507 ± 0.1002	0.0507 ± 0.1002	0.0000 ± 0.0000
	Final-token write	0.0507 ± 0.1002	0.0507 ± 0.1002	0.0000 ± 0.0000
Final layer	Matched clean	0.0507 ± 0.1002	0.0508 ± 0.1000	-0.0001 ± 0.0029
	Unrelated donor	0.0507 ± 0.1002	$\mathbf{0.0506} \pm 0.1001$	$\mathbf{0.0001} \pm 0.0037$
	Permuted channels	0.0507 ± 0.1002	0.0717 ± 0.1022	-0.0210 ± 0.0313
	Self restoration	0.0507 ± 0.1002	0.0507 ± 0.1002	0.0000 ± 0.0000
	Final-token write	0.0507 ± 0.1002	0.0507 ± 0.1002	0.0000 ± 0.0000

Table 38: Advantage of matched clean routes over unrelated or shuffled routes (mean $\pm$ SD of paired differences in JS reduction). Positive values favor matched routes. Each column compares two interventions; it is not a ranking of models. Harm from control routes can make this contrast larger than restoration itself.

		Matched-route advantage ↑	
Weighting	Scope	Matched − donor	Matched − permuted
Unique pairs	All	0.0150 ± 0.0358	0.3870 ± 0.1391
Unique pairs	Final	0.0004 ± 0.0036	0.0219 ± 0.0307
Original rows	All	0.0092 ± 0.0302	0.4185 ± 0.1354
Original rows	Final	-0.0002 ± 0.0045	0.0209 ± 0.0306

Table 39: Proportional restoration (PMR) and fraction of improved pairs under matched clean routes (mean ± SD). Categories receive equal weight. Bold marks the higher displayed mean across the two layer scopes within each weighting scheme. All 1,422 pairs are included.

Weighting	Scope	Mean PMR ↑	Fraction improved ↑	Eligible pairs
Unique pairs	All	$\mathbf{0.0035}\pm0.1966$	$\mathbf{0.5333}\pm0.4994$	1422/1422
Unique pairs	Final	-0.0058 ± 0.0531	0.4848 ± 0.5003	1422/1422
Original rows	All	$\mathbf{0.0172}\pm0.1634$	$\mathbf{0.6023}\pm0.4895$	1422/1422
Original rows	Final	-0.0029 ± 0.0437	0.5111 ± 0.4999	1422/1422