
Understanding Value Embeddings in GPT-2 Training Speedruns

Stanford CS224N Custom Project

Markus Zhang
Department of Computer Science
Stanford University
markusz@stanford.edu

Arihan Varanasi
Department of Computer Science
Stanford University
arihan@stanford.edu

Abstract

In language modeling, many new Transformer variants found at small parameter scale do not hold under scaling up, so we should first interpret their learned functions before trusting them at scale. Value Embeddings (VEs), learned embedding tables injected into the values of attention heads of Transformers, are emerging as a popular new architecture choice in GPT-2 pretraining speedruns due to strong performance, yet little is known about their function. Therefore, by pretraining a VE-model family on the 200M to 900M scale, we contribute the first mechanistic interpretation of VEs, focused on analyses of increasing fidelity: weight matrices, head-level semantic spaces and rank-1 steering via VE vectors. First, we examine VEs as matrices and that VEs in later layers consistently maintain lower condition numbers (less “stretch”) throughout training, which suggests later VE vectors are more diffuse and spread their capacity uniformly across dimensions. Next, within attention heads, we find some directions in VE map to interpretable lexical features (gender, plurality, tense) with linear word analogies matching the performance of Word2Vec, but complex or contextual features (grammar, semantics) have no linear directions. Last, we synthetically generate causal interchange interventions for lexical features, finding that its effect on log-odds ratios (LoR) scales linearly then saturates. The steering effect also decreases exponentially with prompt length, suggesting the model has learned a tradeoff between per-token and contextual information in VEs and other Transformer circuits. We conclude VEs are *per-token information caches*: static lookup tables that encode token-level linguistic features along attention heads, with linearity for lexical features as a special case that enables interpretable, rank-1 causal steering.

1 Key Information to include

- Mentor: Julie Kallini
- External Collaborators (if you have any): No
- Sharing project: No
- Late Days (if you are using any):

Team member	Late days remaining
Arihan Varanasi	1
Markus Zhang	1
Total pooled	2
Team size	2
Late days applied for the report	1

2 Introduction

Pretraining speedruns have become a source of new Transformer architectures: the modded-nanogpt project [1], an open-source competition to train GPT-2 [2] as fast as possible on a single $8\times\text{H100}$ node, has produced many architectural improvements adopted largely on empirical grounds. One such modification is the Value Embedding (VE). Inspired by Value Residual Learning [3], a VE is a learned lookup table with the same matrix shape as an embedding ($|V| \times d$), but is injected into the value of each attention head at selected Transformer blocks (Figure 1). VEs substantially increase the model’s parameter count yet incur near-zero additional inference FLOPs, since each token simply indexes a row from a matrix. Under fixed hardware and wall-clock budgets, VE models consistently achieve lower validation loss than non-VE alternatives [4, 5], and Karpathy further validated the performance of VEs in nanochat [6] at a scale of roughly $100M - 2B$ parameters. Despite recent community adoption, what VE matrices actually learn remains unstudied.

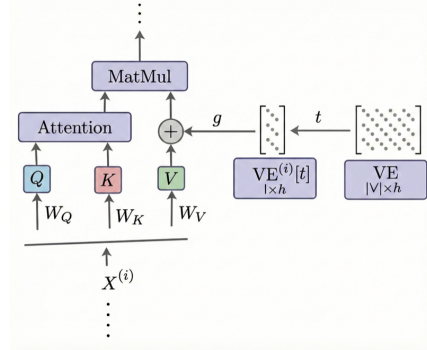


Figure 1: Attention block with VE injection via learned gate g .

Existing interpretability methods target specific transformer components such as attention patterns, residual streams, or MLP neurons, and therefore do not directly apply to Value Embeddings. We address this gap by adapting methods from three domains. From numerical linear algebra, we track the condition number of VE weight matrices over training to characterize their geometry. From classical NLP, we test whether VEs within attention heads encode linguistic features in linear directions, using word-analogy probing with Word2Vec as a baseline. From mechanistic interpretability, we apply interchange interventions and rank-1 vector steering to measure whether detected features functionally influence the model’s predictions.

We find that VEs function as *per-token feature caches*. As matrices, VEs in later layers consistently maintain lower condition numbers throughout training, suggesting their vectors spread capacity more uniformly across dimensions, while the earliest VE bank retains concentrated, low-rank structure. Within attention heads, we find that some directions in VE space map to interpretable lexical features (gender, plurality, tense) with linear word analogies matching the performance of Word2Vec, but complex or contextual features (grammar, semantics) have no linear directions. Causal interchange interventions confirm that lexical features are functionally used: log-odds ratios scale linearly with intervention strength then saturate, and the steering effect decreases exponentially with prompt length, suggesting the model learns a tradeoff between per-token and contextual information. We conclude that VEs are static lookup tables whose head subspaces encode token-level linguistic features, with linearity for lexical features as a special case that enables rank-1 causal steering.

3 Related Work

Value Residual Learning and modded-nanogpt. Zhou et al. [3] introduced Value Residual Learning to alleviate attention concentration in deep transformers, adding a residual connection from the input embedding into each layer’s attention value. Loosely inspired by this work, the modded-nanogpt speedrun community [1] developed Value Embeddings—independent learned matrices injected at selected layers—and found that VEs achieve state-of-the-art GPT-2 training speed [7, 4]. Karpathy’s nanochat project [5] subsequently scaled VEs to larger models. However, no prior work has studied *what* VE matrices learn or why they are effective; our work is the first interpretability analysis of this architecture.

Mechanistic interpretability. A growing body of work seeks to reverse-engineer the internal representations of neural networks. Elhage et al. [8] showed that neural networks can pack multiple features into fewer dimensions via superposition, motivating per-head analysis of learned representations. Olsson et al. [9] identified induction heads as a mechanistic basis for in-context learning in transformers. Our work applies a similar per-head analysis strategy to VE matrices, asking whether

individual 128-dimensional head subspaces encode interpretable linguistic features. This strategy is closely related to recent causal and memory-editing studies that localize and manipulate Transformer subcircuits [10, 11, 12].

Linear structure in embeddings. Mikolov et al. [13] showed that word embeddings encode semantic relationships as linear directions (e.g., $\vec{king} - \vec{man} + \vec{woman} \approx \vec{queen}$). Mu and Viswanath [14] further showed that word embedding geometry is shaped by frequency: a common mean vector and top principal components encode frequency rather than semantics. We investigate whether VE head subspaces encode *morphosyntactic* features (grammatical properties like gender, person, and tense), using Word2Vec as a baseline for comparison.

Causal evaluation and activation steering. Hu and Levy [15] introduced CausalGym, a benchmark for evaluating causal effects of model interventions on syntactic tasks using interchange interventions and log-odds ratios. Zou et al. [16] proposed representation engineering, showing that internal representations can be steered by adding direction vectors at inference time. Our steering experiments follow this paradigm but target a different locus: rather than steering hidden-state activations, we intervene directly on VE lookup vectors, exploiting their token-level independence from context.

4 Approach

4.1 Value Embedding Architecture

A Value Embedding (VE), inspired by Value Residual Learning [3], is a learned matrix $E_v \in \mathbb{R}^{|V| \times h}$, where $|V|$ is the vocabulary size and h is the hidden dimension. Given an input token t , we look up its row $E_v[t] \in \mathbb{R}^h$ and inject it into the attention value computation (the V in $\text{Attention}(Q, K, V)$) via a gated sum:

$$V' = W_v x + 3\sigma(W_g \cdot x_{1:12}) \cdot E_v[t] \quad (1)$$

where $W_v x$ is the usual value projection, σ is the sigmoid function, and the equation describes a single attention head. The gate weight $W_g \in \mathbb{R}^{12}$ reads the first 12 dimensions of the hidden state x , producing one scalar gate value per head per token. Because the gate maps only 12 input dimensions (out of 768) through a sigmoid, its output has limited capacity to vary across the vocabulary. The factor of 3 scales the sigmoid output to the range $[0, 3]$, with a neutral value of 1.5 at initialization (Figure 1).

As described in §2, we analyze nanochat d12 [5], which inserts 6 VE banks (one independent E_v matrix per layer) at every odd-numbered layer $\{1, 3, 5, 7, 9, 11\}$. We refer to these six banks as VE_0 through VE_5 , corresponding respectively to layers 1, 3, 5, 7, 9, and 11.

4.2 Analysis Framework

We investigate VEs through three experiments that form a logical progression from *observation* to *structure* to *causation*.

Experiment 1: Static weight analysis. The natural starting point is to examine what VE matrices look like as learned objects, without running any forward pass. We track spectral properties of each VE bank $E_v \in \mathbb{R}^{|V| \times 768}$ over the course of training, focusing on the *condition number*:

$$\kappa(E_v) = \frac{\sigma_{\max}(E_v)}{\sigma_{\min}(E_v)} \quad (2)$$

where $\sigma_{\max}$ and $\sigma_{\min}$ are the largest and smallest singular values of the matrix. Intuitively, each row of E_v is a point in 768-dimensional space, and the condition number measures how “stretched” this point cloud is. If $\kappa \approx 1$, the cloud is roughly spherical—the tokens spread out equally in every direction, with no preferred axis. If κ is large, the cloud forms an elongated ellipsoid: most of the variation among token vectors lies along a few directions while the remaining dimensions are compressed. A high condition number therefore indicates a more structured, lower-rank representation where the matrix has learned to concentrate information along specific axes. By tracking κ across VE layers and training steps, we can observe how each bank’s geometry evolves and whether different layers develop qualitatively different structure—without relying on corpus statistics or any forward pass through the model.

Experiment 2: Semantic spaces. We move beyond geometry and ask whether VE *directions* encode interpretable features. If a feature like gender is linearly encoded in a head’s subspace, then the difference vectors $\vec{v}(\text{king}) - \vec{v}(\text{queen})$ and $\vec{v}(\text{he}) - \vec{v}(\text{she})$ should all point in the same direction—the “gender direction.” We measure this alignment across many pairs.

For each feature, we synthetically generate candidate word pairs that differ along one linguistic axis using Claude Opus, manually verify each feature for correctness, and filter to pairs where both words are single GPT-2 tokens. Each VE vector is partitioned into $H = 6$ head subspaces of 128 dimensions. Given n pairs $\{(w_i^+, w_i^-)\}_{i=1}^n$, we compute difference vectors per head $d_i^{(j)} = E_v^{(j)}[w_i^+] - E_v^{(j)}[w_i^-]$ and measure the *mean pairwise cosine similarity* of these difference vectors:

$$\overline{\text{cos}}^{(j)} = \frac{2}{n(n-1)} \sum_{i < k} \frac{d_i^{(j)} \cdot d_k^{(j)}}{\|d_i^{(j)}\| \|d_k^{(j)}\|} \quad (3)$$

A value near 1 means all difference vectors point the same way (the feature has a consistent linear direction in that head); a value near 0 means the difference vectors are scattered (no shared direction).

Experiment 3: Causal steering. Finding linear structure does not prove the model *uses* those features. We close this gap using the CausalGym framework [15], which provides paired minimal sentences for controlled measurement of how interventions shift next-token predictions. We evaluate on CausalGym’s 29 syntactic tasks and 325 word pairs (across 8 features) from BATS [17] and our generated lists.

We apply two interventions. *Interchange* swaps the VE vector at a token position between paired sentences and measures the *log-odds ratio* (LOR):

$$\text{LOR} = (\log P_{\text{int}}(y_0) - \log P_{\text{int}}(y_1)) - (\log P_{\text{orig}}(y_0) - \log P_{\text{orig}}(y_1)) \quad (4)$$

where y_0, y_1 are the two target tokens and P_{int} is the distribution after interchange. LOR isolates whether VE causally shifts the model’s preference between two specific outcomes.

Finally, *steering* tests whether this causal effect is *linear* and continuously controllable. Rather than a discrete swap, we scale the VE difference vector by a continuous factor c :

$$\text{VE}_{\text{final}} = \text{VE}_{\text{base}} + c \cdot (\text{VE}_{\text{src}} - \text{VE}_{\text{base}}) \quad (5)$$

At $c = 0$ we recover the original VE; at $c = 1$ we perform a full interchange; values $c > 1$ *amplify* the feature direction beyond the natural difference between the two tokens. If VE encodes the feature as a linear direction, LOR should scale proportionally with c . We sweep $c \in \{-1, -0.5, 0, 0.25, 0.5, 0.75, 1, 1.5, 2, 3\}$, spanning three regimes—reversal ($c < 0$, pushing away from the source feature), interpolation ($0 < c < 1$), and amplification ($c > 1$)—and fit a linear regression of LOR against c ; a high R^2 confirms that the causal effect grows smoothly with intervention strength rather than saturating or behaving nonlinearly.

5 Experiments

Although we trained a model family, for consistency and ease of experimentation, all analyses probe the nanochat d12 model [5]: a 12-layer transformer with 286M base parameters and 6 attention heads per layer ($h = 768$, head dimension 128), trained on FineWeb [18]—a large-scale English web corpus—for 2,205 steps on a single 8×H100 node using a custom BPE tokenizer (vocabulary size of $|\mathcal{V}|32,768$). The model inserts 6 VE matrices at alternating layers $\{1, 3, 5, 7, 9, 11\}$, each a learned matrix $E_v \in \mathbb{R}^{|\mathcal{V}| \times h}$, adding ~ 176 VE parameters on top of the 110M base. We save checkpoints at 10 evenly spaced points during training to track how VE geometry evolves. For semantic probing (Experiment 2), we construct word-pair datasets for 17 linguistic features by generating candidate pairs with Claude Opus 4.6, manually verifying each pair, and filtering to pairs where both words resolve to single BPE tokens; pair counts range from 3 (reflexive person) to 36 (progressive). Future work will increase the sample sizes of features for statistical rigor.

5.1 Static Weight Analysis

Condition number reveals layer-level structure. We compute $\kappa(E_v)$ (Equation 2) for each VE bank via full SVD at each of the 10 training checkpoints. Recall that $\kappa \approx 1$ indicates uniform use of all dimensions, while large κ signals a concentrated, lower-dimensional structure (§4.2).

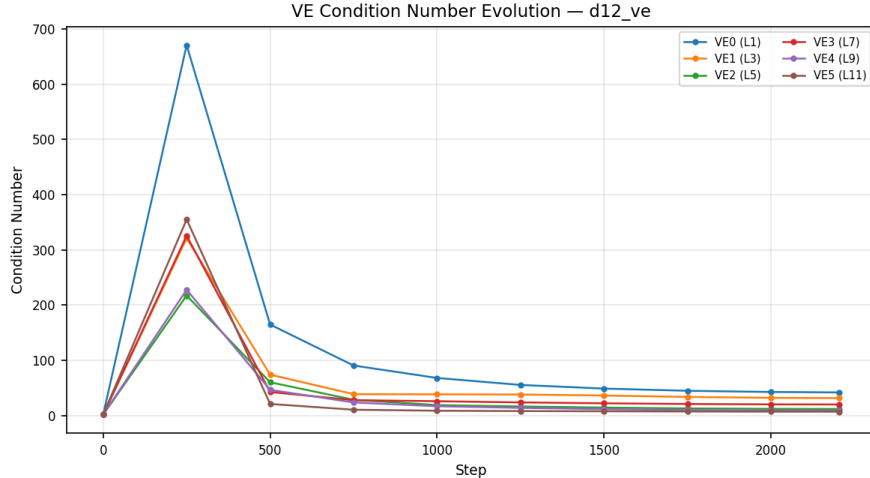


Figure 2: Condition number κ over training for each VE bank. All banks spike early then decay; VE_0 (layer 1) retains $\kappa \approx 40$ while later banks converge to $\kappa \approx 8\text{--}12$.

Training dynamics. Figure 2 reveals a striking pattern. In the first ~ 300 training steps, all VE banks spike to high condition numbers (VE_0 peaks at ~ 670), reflecting rapid learning of a few dominant directions. Over subsequent training, the condition numbers decay as the model learns to use more of the available dimensions. By step 2,205 (end of training), condition numbers have largely stabilized.

Layer differentiation. The key finding is that VE banks converge to *different* final condition numbers depending on their position in the network (Figure 2). VE_0 (layer 1, the earliest bank) stabilizes at $\kappa \approx 42$ —roughly $4\times$ higher than VE_2 through VE_5 , which converge to $\kappa \approx 8\text{--}12$. This means the earliest VE bank concentrates its information in fewer effective dimensions, suggesting a more specialized, lower-rank role. In contrast, the later banks spread their representational capacity more uniformly across the full 768 dimensions, consistent with encoding a richer, higher-dimensional set of features.

Since all six VE banks share the same architecture and initialization, this divergence must arise from the different roles that early vs. late layers play in the transformer. We investigate what those features are (§5.2) and whether they causally influence the model’s predictions (§5.3).

5.2 Semantic Spaces

The condition number analysis in §5.1 shows that later VE banks use their full dimensionality more uniformly, but does not tell us *what* those dimensions encode. We now probe VE *directions* to test whether individual attention heads encode interpretable linguistic features as linear subspaces.

Data and evaluation. We evaluate 17 linguistic features, each defined by a set of manually verified word pairs where both tokens are single GPT-2 BPE tokens (see §4.1 for construction details). The features span three categories: 10 *inflectional* features that involve closed-class grammatical alternations determinable from a single token (e.g., gender: *she*→*he*; tense: *ran*→*runs*; case: *he*→*him*), 4 *reflexive* variants that combine reflexive and regular pronouns, and 3 *null* features (adjective antonyms, concrete/abstract nouns, -ly adverbs) that serve as negative controls. We compute $\overline{\cos}^{(j)}$ (Equation 3) for each feature across all 36 head slots (6 VE layers $\times$ 6 heads), using the null features to calibrate the noise floor.

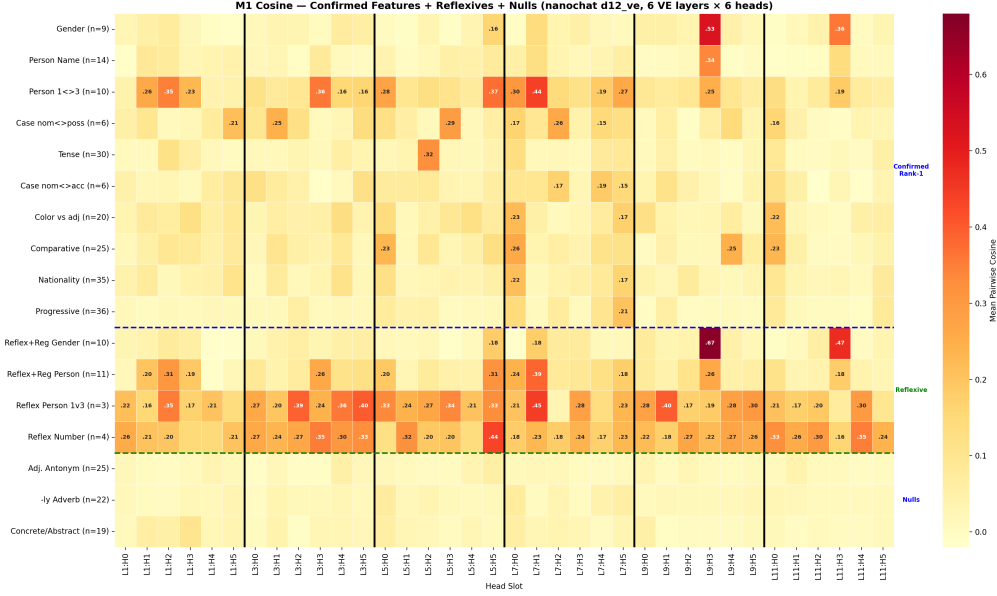


Figure 3: Mean pairwise cosine similarity $\overline{\cos}^{(j)}$ per feature per head slot. Inflectional features show strong signal in specific heads; null features remain near zero everywhere.

Head-level specialization. Figure 3 reveals that inflectional features produce cosine values 3–10× above the null baseline in specific heads, while null features remain far below the confirmation threshold across all 36 slots. Crucially, different features activate *different* heads: gender concentrates in L9:H3, person in L7:H1, tense in L5:H2, and case in late-layer heads such as L7:H2 and L7:H4. This means the 128-dimensional subspace of each head acts as an independent channel specializing for a distinct feature. Reflexive features (e.g., *herself*→*himself*) activate the same heads as their non-reflexive counterparts, confirming that the gender direction is shared across pronoun types.

Inflectional features dominate. Figure 4 summarizes the best-head cosine across all features. A clear hierarchy emerges: the strongest inflectional features (gender, person, number, and case) exceed the 0.15 confirmation threshold, with gender reaching $\overline{\cos} = 0.529$, while tense is weaker but still above the null controls at 0.104. In contrast, derivational and semantic features cluster near the noise floor and remain well below the 0.15 confirmation threshold (roughly $\overline{\cos} \approx 0.06$ –0.10).

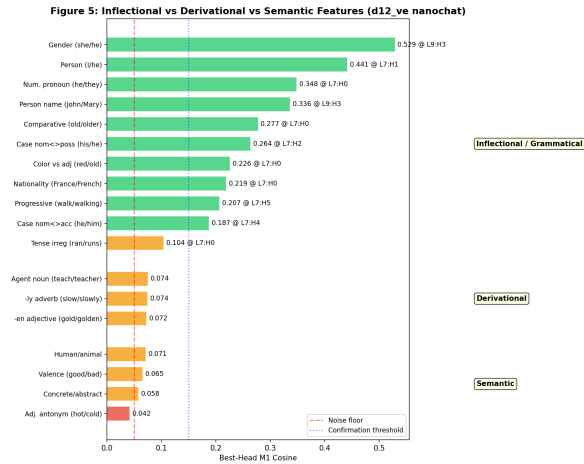


Figure 4: Best-head cosine by feature type. Inflectional features rise well above the noise floor, while derivational and semantic features do not.

This pattern is consistent with VE being a context-free lookup: it naturally favors token-intrinsic regularities such as inflectional alternations (*he/she*, *run/runs*) over broader derivational or semantic relations. Notably, the best VE head achieves about $1.37\times$ the directional consistency of Word2Vec on gender (0.529 vs. 0.387), despite VEs being trained for language modeling rather than explicit analogy tasks.

5.3 Steering

Having found interpretable linear directions in VE head subspaces (§5.2), we now test whether these features *causally* influence the model’s predictions. We apply continuous steering (Equation 5) and measure log-odds ratios (Equation 4), finding that VE encodes features as linear, reversible causal pathways.

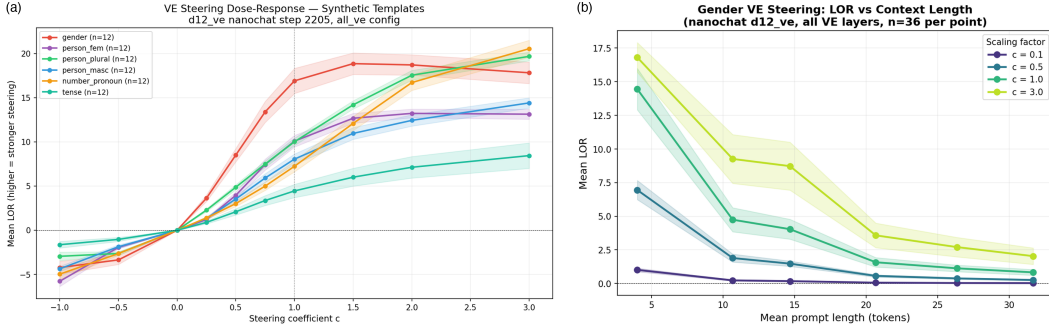


Figure 5: **(a)** Log-odds ratio vs. steering coefficient c for six synthetic features (all VE layers). All features produce approximately linear LOR through the origin; negative c reverses the effect. **(b)** LOR decay with context length for gender interchange ($n = 36$ per length bin). LOR drops $17\times$ from 4 to 31 tokens (Spearman $\rho = -0.83$, $p < 10^{-56}$).

LOR scales linearly with intervention strength. Figure 5a shows that all six synthetic features—gender, person_fem, person_plural, person_masc, number_pronoun, and tense—produce approximately linear LOR vs. c through the origin across the sweep described in §4.2. For gender, a linear fit yields $R^2 = 0.989$ with $\text{LOR} \approx 17$ at $c = 1$ —the steepest slope among all features. Negative c reverses the effect, pushing predictions *away* from the source feature. This linearity and reversibility confirm that VE encodes features as additive directions rather than nonlinear switches.

Steering effect decays with context length. We test how far the VE signal travels through the sequence by embedding 12 gendered noun pairs in sentences of varying length (3 filler chains $\times$ 6 length levels, $n = 36$ per bin) and measuring LOR at the final token. Figure 5b shows that at $c = 1$, LOR drops from $+14.4$ (95% CI: [12.9, 16.0]) at 4 tokens to $+0.8$ (95% CI: [0.6, 1.0]) at 31 tokens—a $17\times$ attenuation. Higher c shifts curves upward without changing the decay shape. This is consistent with attention dilution: as more tokens intervene between the steered noun and the prediction point, the final position attends less to the noun’s VE contribution.

VE steering transfers to generation. The experiments above measure causal effects on fixed prompts. To test whether VE features extend to open-ended text, we apply a complementary intervention on a larger model: we add the contrastive direction $\delta = E_v[\textit{she}] - E_v[\textit{he}]$ to every row of the VE matrix, scaled by c , and generate autoregressively from d20_ve (897M parameters, 10 VE banks). Table 1 shows that the unsteered baseline produces roughly balanced output (44% feminine). Steering at $c = 0.5$ shifts the model to 100% feminine or 0% feminine depending

Condition	# fem	# masc
Baseline	7	9
$\rightarrow\textit{fem}$, $c=0.5$, all VE	13	0
$\rightarrow\textit{masc}$, $c=0.5$, all VE	0	20
$\rightarrow\textit{fem}$, $c=1.0$, mid-4 VE	13	2

Table 1: Gendered tokens in autoregressive generation (d20_ve, 897M; 6 prompts $\times$ 2 seeds, 50 tokens each).

on direction—symmetric and bidirectional. A mid-layer subset (layers 7/9/11/13) achieves 87% at $c = 1.0$ while maintaining higher text coherence than all-layer steering at the same scale, consistent with early layers serving a gain-control role (§5.1) rather than encoding feature directions.

Qualitative examples in Appendix A (Tables 2–4) confirm that steering produces consistent gendered pronouns while largely preserving topic and fluency.

6 Analysis

Our three experiments reinforce a coherent picture of VEs as *per-token feature caches*. A consistent finding across all three is that early and late VE layers play different roles: VE_0 maintains $\sim 4\times$ higher condition number (Figure 2), shows no strong feature signal (Figure 3), and mid-layer steering outperforms all-layer steering (Table 1). This convergence from independent methods—spectral, probing, and causal—strengthens our confidence in the early-layer gain-control vs. late-layer feature-encoding distinction.

Ves succeed at encoding closed-class inflectional features determinable from a single token (Figure 4), but fail on semantic and derivational features that require context ($\overline{\cos} \approx 0.04\text{--}0.07$, indistinguishable from null controls). The steering effect also decays $17\times$ with context length (Figure 5b), confirming that VE influence is strongest for nearby predictions. Qualitatively, steered generations (Table 2, Appendix A) show consistent pronoun shifts, but also reveal that VE directions influence correlated lexical choices beyond just grammatical agreement—the masculine-steered scientist output introduces “Edwin Hubble,” suggesting VE features capture training-data correlations, not purely grammatical structure.

7 Conclusion

We presented the first interpretability analysis of Value Embeddings, characterizing them as *per-token feature caches*: static lookup tables whose head subspaces specialize for distinct inflectional features and whose causal influence scales linearly with intervention strength. The consistent early-layer gain-control vs. late-layer feature-encoding distinction, supported by three independent methods, suggests that VE architecture could be exploited for targeted, per-feature steering of model behavior.

Limitations. Our word-pair datasets are modest (3–36 pairs per feature), limiting statistical power for low- n features. We analyze one model family on one dataset, so we cannot claim universality. The generation experiments use a larger model (d20_ve, 897M) than the probing experiments (d12, 286M), making cross-experiment comparisons imprecise. Our feature inventory is narrow—VEs may encode structure (e.g., syntactic category) that our probes were not designed to detect. Finally, we show VEs are *sufficient* for these features but not that they are *necessary*: the model may encode them redundantly in other parameters.

Future work. Natural extensions include testing whether VE representations are consistent across training distributions, extending the feature inventory beyond inflectional morphology, and investigating whether VE’s head-level organization can be exploited for more targeted model editing than full hidden-state steering [16].

Team contributions (Required for multi-person team)

Arihan Varanasi led the semantic-spaces analysis, including feature selection, evaluation design, and interpretation of the head-level probing results, and also contributed to the steering experiments and their analysis. Markus Zhang led the overall project direction and the remaining experimental components, including the static weight analysis, and contributed to the steering experiments, writing, and revision of the report.

References

- [1] modded-nanogpt contributors. modded-nanogpt: GPT-2 training speedrun. <https://github.com/KellerJordan/modded-nanogpt>, 2024.
- [2] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.
- [3] Zhanchao Zhou, Tianyi Wu, Zhiyun Jiang, Fares Obeid, and Zhenzhong Lan. Value residual learning for alleviating attention concentration in transformers. In *International Conference on Learning Representations*, 2025.
- [4] Markus Zhang. New record: Untie value embeddings (-0.4s, -25 steps). GitHub Pull Request #209, <https://github.com/KellerJordan/modded-nanogpt/pull/209>, January 2026. Merged into KellerJordan/modded-nanogpt on Jan 29, 2026.
- [5] Andrej Karpathy. Beating GPT-2 for <\$100: the nanochat journey. GitHub Discussion #481, <https://github.com/karpathy/nanochat/discussions/481>, January 2026. Nanochat repository announcement, Jan 31, 2026.
- [6] Andrej Karpathy. nanochat: The best ChatGPT that \$100 can buy. <https://github.com/karpathy/nanochat>, 2025.
- [7] Markus Zhang. New WR: Tuned value embeddings (-0.5s). GitHub Pull Request #218, <https://github.com/KellerJordan/modded-nanogpt/pull/218>, February 2026. Merged into KellerJordan/modded-nanogpt on Feb 16, 2026.
- [8] Nelson Elhage, Tristan Hume, Catherine Olsson, Nicholas Schiefer, Tom Henighan, Shauna Kravec, Zac Hatfield-Dodds, Robert Lasenby, Dawn Drain, Carol Chen, Roger Grosse, Sam McCandlish, Jared Kaplan, Dario Amodei, Martin Wattenberg, and Christopher Olah. Toy models of superposition. *Transformer Circuits Thread*, 2022.
- [9] Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. In-context learning and induction heads. *Transformer Circuits Thread*, 2022.
- [10] Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in gpt. *arXiv preprint arXiv:2002.10689*, 2020.
- [11] Kevin Meng, Arnab Sen Sharma, Alex Andonian, Yonatan Belinkov, and David Bau. Mass-editing memory in a transformer. *arXiv preprint arXiv:2202.05262*, 2022.
- [12] Owen Dugan, Roberto Garcia, Ronny Junkins, Jerry Liu, Dylan Zinsley, Sabri Eyuboglu, Atri Rudra, and Chris Ré. Constructing efficient fact-storing mlps for transformers. *arXiv preprint arXiv:2512.00207*, 2025.
- [13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.
- [14] Jiaqi Mu, Suma Bhat, and Pramod Viswanath. All-but-the-top: Simple and effective postprocessing for word representations. In *International Conference on Learning Representations*, 2018.
- [15] Jennifer Hu and Roger Levy. Prompting is not a substitute for probability measurements in large language models. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2023.
- [16] Andy Zou, Long Phan, Sarah 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 Koyber, 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.

- [17] Anna Gladkova, Aleksandr Drozd, and Satoshi Matsuoka. Analogy-based detection of morphological and semantic relations with word embeddings: what works and what doesn't. In *Proceedings of the NAACL Student Research Workshop*, pages 8–15, 2016.
- [18] Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Thomas Wolf, and Leandro von Werra Joshi. Fineweb: decanting the web for the finest text data at scale. *arXiv preprint arXiv:2406.17557*, 2024.

A Additional Generation Examples

Tables 2–4 show VE-steered generation samples from d20_ve at $c = 0.5$ with all VE layers. In each case, the unsteered baseline uses gender-neutral language (“their,” “they”) while steered variants produce consistent gendered pronouns.

Baseline	→fem ($c=0.5$)	→masc ($c=0.5$)
<i>“The scientist published a groundbreaking paper and then”</i>		
released the publicly available findings of their research, which has been accepted by numerous renowned peer review journals...	defended her work in an online journal, explained to reporters that “the key question we’re asking is who gets the message first when she ’s working on climate science?”	released the high-profile image of his close companion and collaborator, Edwin Hubble... gave credit for this discovery to so many others before him .

Table 2: VE-steered generation: Scientist prompt (d20_ve, $c = 0.5$, all VE layers, seed 42). Bold pronouns show gender shift.

Baseline	→fem ($c=0.5$)	→masc ($c=0.5$)
<i>“The doctor walked into the room and told the patient that”</i>		
their blood pressure was too high. The patient explained to the doctor that it seemed as though they had a headache...	their blood pressure was ‘average’ and, as the doctor made brief notes in her head about it...	their medication was working, but the patient didn’t feel anything. He did not know what to do...

Table 3: VE-steered generation: Doctor prompt (d20_ve, $c = 0.5$, all VE layers, seed 42).

Baseline	→fem ($c=0.5$)	→masc ($c=0.5$)
<i>“In a small coastal town, the local baker was famous for”</i>		
making a special treat every day—one she proudly labeled as her “favorite flavor.” With the help of the sweetest man in town...	making incredibly light and airy cakes using almond flour, which she would use as a neutral base for the layers of toppings...	making a special loaf every day at his local bakery. The owner, Mr. Bert Herman, owned the place for many years...

Table 4: VE-steered generation: Baker prompt (d20_ve, $c = 0.5$, all VE layers, seed 43). The baker baseline already uses feminine pronouns; steering reinforces or reverses this default.