

SLED: Self Logits Evolution Decoding for Improving Factuality in Large Language Models

Jianyi Zhang¹, Da-Cheng Juan², Cyrus Rashtchian², Chun-Sung Ferng², Heinrich Jiang², Yiran Chen¹

¹ Duke University, ² Google Research*
[Project Website](#)

Abstract

Large language models (LLMs) have demonstrated remarkable capabilities, but their outputs can sometimes be unreliable or factually incorrect. To address this, we introduce **Self Logits Evolution Decoding (SLED)**, a novel decoding framework that enhances the truthfulness of LLMs without relying on external knowledge bases or requiring further fine-tuning. From an optimization perspective, our SLED framework leverages the latent knowledge embedded within the LLM by contrasting the output logits from the final layer with those from early layers. It then utilizes an approximate gradient approach to enable latent knowledge to guide the self-refinement of outputs, thereby effectively improving factual accuracy. Extensive experiments have been conducted on established benchmarks across a diverse range of model families (Gemma, Qwen, Mixtral, gpt-oss) and scales (from 1B to 45B), including more advanced architectural configurations such as the mixture of experts (MoE). Our evaluation spans a wide variety of tasks and the results demonstrate that SLED consistently improves factual accuracy compared to existing decoding methods while maintaining natural language fluency and negligible latency overhead. Furthermore, it can be flexibly combined with other decoding methods to further enhance their performance.

1 Introduction

Large Language Models (LLMs) have achieved remarkable breakthroughs in recent years, demonstrating exceptional performance across various domains [1, 16, 17, 21]. However, a significant challenge associated with LLMs is their tendency to hallucinate or distort the truth, resulting in outputs that are not factual [5, 6, 31]. This issue of hallucination undermines the reliability and trustworthiness of LLMs in practical applications. A popular strategy for improving the LLM factuality involves refining the decoding process [20, 26]. Decoding focuses on how the model selects the next token during the generation process, which can significantly influence the factual accuracy of the output. The decoding methods can be cost-effective since (a) they do not rely on external knowledge and (b) no additional training is required. Furthermore, decoding methods can be synergistically combined with other techniques aimed at improving the LLM factuality, such as retrieving information from external knowledge bases [9, 10], various fine-tuning

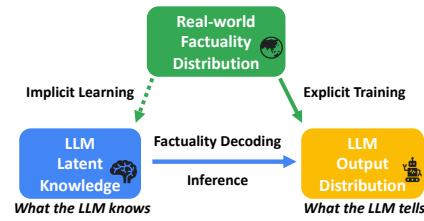


Figure 1: Factuality decoding overview.

*Due to policy restrictions on the models used, this paper has been updated. For more detailed content, please refer to the previous version.

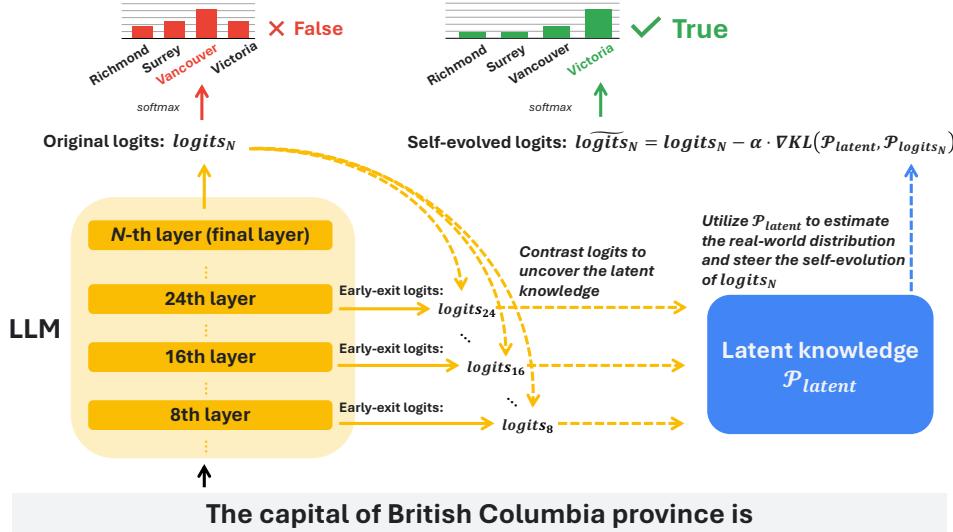


Figure 2: Illustration of our Self Logits-Evolution Decoding (SLED) workflow.

strategies for better alignment [24], or ensemble learning methods [3].

Recent studies [8, 11, 19, 25] suggest that LLMs sometimes have learned the factual content based on extensive pretraining or fine-tuning, although they fail to produce the correct answer when a user queries the model. This has inspired the development of several factuality decoding methods [2, 11, 12, 30] to reveal what the model implicitly "knows." Figure 1 summarizes the underlying mechanism of these factuality decoding methods. The LLMs' output distribution is derived by applying the softmax function to the output logits from the final layer. During the training phase, this distribution is optimized based on the real-world factuality distribution represented by the training dataset. However, during the inference phase, "what the LLM tells" might still contain factual errors, which implies a discrepancy between the output distribution and the real-world factuality distribution. While the real-world distribution remains inaccessible during the inference phase, the model's latent knowledge ("what the model knows") may have implicitly learned some factual content correctly during the training phase [8, 25]. Therefore, a key challenge for factuality decoding strategies lies in effectively harnessing the latent knowledge embedded within LLMs to refine the output distribution (logits) during inference.

To address this challenge, we propose **Self Logits Evolution Decoding (SLED)**, a novel factuality decoding approach that leverages the latent knowledge within LLMs by contrasting the final layer's logits with early layers' logits. During the decoding process, as LLMs progress from early to final layers, they progressively incorporate factual information stored in each layer into the output. SLED tracks this evolution process to unearth the latent knowledge within LLMs, and enables the "self-evolution" of the output distribution further to align it more closely with real-world facts. Furthermore, our approach recognizes that the latent knowledge within LLMs, while valuable, may not always be perfect. Therefore, instead of simply replacing the original outputs with this latent knowledge, SLED integrates it into the original logits through an operation similar to "single-step gradient descent" over the output logits during the inference time. This operation minimizes the Kullback-Leibler (KL) divergence between the latent knowledge distribution and the output distribution, effectively balancing the two and mitigating potential drawbacks such as overfitting or biased outputs. Figure 2 illustrates the SLED workflow, highlighting how SLED optimizes the output logits, leading to a more factual output distribution. We evaluate SLED on various LLMs e.g., Gemma-3 [22], Qwen-3 [27], GPT-OSS [18]) and benchmarks to demonstrate its state-of-the-art performance in layer-wise contrastive decoding methods. In summary, our main contributions are:

- We propose SLED, a novel decoding method that aligns LLMs outputs with factual knowledge without requiring an external knowledge base or fine-tuning data.
- We conduct extensive experiments across a range of LLMs, with varying configurations and scales. The results demonstrate that SLED consistently improves factual accuracy on various tasks and benchmarks, including multiple-choice, open-ended generation, and chain-of-thought reasoning tasks.

Algorithm 1 Self Logits Evolution Decoding

- 1: **Initialization:** LLM with N layers, $inputs$, evolution rate α , evolution scale $k > 0$, $\eta \ll 0$, temperature parameter τ , and the one-hot vectors $\{\mathcal{P}_{e_i}\}$ defined in Section 2.3.
 - 2: Feed the $inputs$ into the LLM to obtain the logits $logits_n = (\ell_{(1,n)}, \dots, \ell_{(d,n)})$ and probabilities $\mathcal{P}_{logits_n} = (p_{(1,n)}, \dots, p_{(d,n)}) = softmax(logits_n/\tau)$ at each layer n , where $n \leq N$.
 - 3: Identify the tokens with the top- k largest values in $logits_N$ and denote their indices by I_k .
 - 4: **for** each early layer n , ($n < N$) **do**
 - 5: Compute differences for top- k logits $logits_n - logits_N$.
 - 6: Calculate $m_i^{(n)} = [\max(\text{CosSim}(logits_n - logits_N, \mathcal{P}_{logits_n} - \mathcal{P}_{e_i}), 0)]^2$, $i \in I_k$.
 - 7: **end for**
 - 8: Compute weighted average $m_i = \frac{\sum_{n=1}^N m_i^{(n)}}{\sum_{n=1}^N \sum_{j \in I_k} m_j^{(n)}}$ across different layers for each $i \in I_k$.
 - 9: **for** each i from 1 to d **do**
 - 10: Set $\tilde{\ell}_{(i,N)} = \ell_{(i,N)} - \frac{\alpha}{\tau}(p_{(i,N)} - m_i)$ **if** $i \in I_k$ **else** Set $\tilde{\ell}_{(i,N)} = \eta \ll 0$.
 - 11: **end for**
 - 12: **Output:** The self-evolved logits are $\widetilde{logits}_N = (\tilde{\ell}_{(1,N)}, \dots, \tilde{\ell}_{(i,N)}, \dots, \tilde{\ell}_{(d,N)})$.
-

- SLED can be flexibly integrated with other factuality decoding methods to enhance their effectiveness further.
- We provide a new interpretable perspective for understanding layer-wise contrastive decoding methods, paving the way for further developments in factuality decoding.

2 Self Logits Evolution Decoding

A large language model, equipped with N layers and a vocabulary $\mathcal{V} = [v_1, v_2, \dots, v_d]$, typically generates text in the next-token prediction fashion. For each given prefix, the model computes the logits at the final (N -th) layer, $logits_N \triangleq (\ell_{(1,N)}, \ell_{(2,N)}, \dots, \ell_{(d,N)})$, which are obtained by applying a linear transformation to the hidden states of the final layer, projecting the high-dimensional hidden state vectors onto the space of the vocabulary size. Subsequently, the output distribution \mathcal{P}_{logits_N} at the final (N -th) layer for the next token is derived by applying softmax function on the logits,

$$\mathcal{P}_{logits_N} \triangleq (p_{(1,N)}, \dots, p_{(d,N)}) = softmax(logits_N/\tau),$$

where τ is the temperature parameter. Therefore, for each $p_{(i,N)}$ ($1 \leq i \leq d$), we have

$$p_{(i,N)} = \exp(\ell_{(i,N)}/\tau)/S, \text{ where } S = \sum_{j=1}^d \exp(\ell_{(j,N)}/\tau).$$

Similarly, we can also derive the logits from early layers by applying the same linear transformation mentioned above to their hidden states. For any early layer n ($n < N$), we denote its logits as $logits_n \triangleq (\ell_{(1,n)}, \dots, \ell_{(d,n)})$ and the corresponding distribution as $\mathcal{P}_{logits_n} \triangleq (p_{(1,n)}, \dots, p_{(d,n)})$.

2.1 Logits Evolution

To improve factual accuracy, it is crucial that the correct token v_i receives a higher value of $logits_N$ to ensure a higher probability value $p_{(i,N)}$ in the output distribution \mathcal{P}_{logits_N} . From a mathematical perspective, this means aligning the model's output distribution \mathcal{P}_{logits_N} closely with the real-world factuality distribution \mathcal{P}_{real} . Specifically, we can formulate this goal as optimizing the following loss function \mathcal{L} regarding the $logits$:

$$\mathcal{L}(logits) \triangleq KL(\mathcal{P}_{real}, \mathcal{P}_{logits}), \text{ where } logits = (\ell_1, \dots, \ell_d), \mathcal{P}_{logits} = softmax(logits/\tau) \quad (1)$$

We describe the above optimization as **Logits Evolution**. Interestingly, the training of LLMs also aims at minimizing the divergence (typically the KL divergence, as the training loss function is often the cross-entropy loss) between the ground truth \mathcal{P}_{real} and the output distribution \mathcal{P}_{logits_N} . During the training phase, the logits evolution is driven externally by the real-world distribution \mathcal{P}_{real} presented in the training dataset, and the corresponding solution is $logits = logits_N$. However, \mathcal{P}_{real} is not accessible during the inference phase. To address this challenge, SLED utilizes the model's latent

knowledge to estimate \mathcal{P}_{real} and enables "self-evolution" of the logits. We denote the estimation as \mathcal{P}_{latent} and the self logits evolution can be achieved by the following gradient-descent operation:

$$\widetilde{\text{logits}}_N = \text{logits}_N - \alpha \cdot \nabla_{\text{logits}_N} KL(\mathcal{P}_{latent}, \mathcal{P}_{logits_N}). \quad (2)$$

The parameter α , termed the **Evolution Rate**, governs the magnitude of adjustments applied to logits_N in the direction of the gradient $\nabla_{\text{logits}_N} KL(\mathcal{P}_{latent}, \mathcal{P}_{logits_N})$. In the following Section 2.2 and 2.3, we discuss how we derive the \mathcal{P}_{latent} as the estimation of the real-world distribution \mathcal{P}_{real} .

2.2 Estimate \mathcal{P}_{real} by Tracking the Logits Evolution Direction throughout Layers

The core principle of our method involves leveraging the difference between each early layer's logits and the final layer's logit, $\text{logits}_n - \text{logits}_N$ to approximate the gradient of $KL(\mathcal{P}_{real}, \mathcal{P}_{logits})$ at $\text{logits} = \text{logits}_n$. Then we estimate \mathcal{P}_{real} based on this approximation.

This is inspired by a new perspective of interpreting the training phase of LLMs as the evolution of logits described in Problem 1. As mentioned above, the solution derived by the training phase is the final layer's logits $\text{logits} = \text{logits}_N$, since the final layer's logits_N directly engage with the real-world distribution \mathcal{P}_{real} through the loss function in training. This implies that we can generally consider the final logits logits_N to be a better solution than the logits from an early layer logits_n , with $KL(\mathcal{P}_{real}, \mathcal{P}_{logits_N}) < KL(\mathcal{P}_{real}, \mathcal{P}_{logits_n})$. Based on this discussion, if we contrast the final layer's logits with the early layer's logits, we can consider the direction (orientation) of $\text{logits}_n - \text{logits}_N$ can approximately align with the direction of the gradient $\nabla_{\text{logits}_n} KL(\mathcal{P}_{real}, \mathcal{P}_{logits})|_{\text{logits}=\text{logits}_n}$. To further verify this motivation, we calculate the cosine similarity between $\text{logits}_n - \text{logits}_N$ and $\nabla_{\text{logits}_n} KL(\mathcal{P}_{real}, \mathcal{P}_{logits_n})$ for thousands of tokens across different models in Figure ???. We find that the majority of these values are positive, which means that the directions of these two vectors are close.

Hence, for each early layer n , we propose to maximize the following function of cosine similarity and derive the $\mathcal{P}_{latent}^{(n)}$ to estimate the \mathcal{P}_{real} .

$$\mathcal{P}_{latent}^{(n)} = \arg \max_{\mathcal{P}} (\text{CosSim}(\text{logits}_n - \text{logits}_N, \nabla_{\text{logits}_n} KL(\mathcal{P}, \mathcal{P}_{logits_n}), 0) \quad (3)$$

2.3 Achieving the Self Logits Evolution in Three Phases

Based on the above analysis, we can introduce the procedures of SLED: First, we estimate $\mathcal{P}_{latent}^{(n)}$ for each early layer n using the gradient approximation in Section 2.2. Subsequently, we apply a weighted average on $\{\mathcal{P}_{latent}^{(n)}\}$ across all early layers $n < N$ to derive \mathcal{P}_{latent} , which serves as the final estimation of the real-world distribution. Finally, we apply \mathcal{P}_{latent} in Equation 2 to facilitate the self-evolution of logits_N , thereby derive the updated logits, $\widetilde{\text{logits}}_N$.

$$\begin{aligned} \text{logits}_n - \text{logits}_N &\xrightarrow{\text{in direction}} \nabla_{\text{logits}_n} KL(\mathcal{P}_{real}, \mathcal{P}_{logits_n}) \\ &\xrightarrow[\text{Estimate}]{\text{Phase 1}} \mathcal{P}_{latent}^{(n)} \xrightarrow[\text{Ensemble}]{\text{Phase 2}} \mathcal{P}_{latent} \xrightarrow[\text{Self-evolution in Eq 2}]{\text{Phase 3}} \widetilde{\text{logits}}_N \end{aligned}$$

Phase 1: An exhaustive search for an exact solution to the complex optimization problem (Equation 3) is computationally impractical. We can reduce the solution space by the following. Suppose the real-world factuality distribution dictates that the next word to be generated is the i -th token v_i from the vocabulary \mathcal{V} . Thus $\mathcal{P}_{real} = \mathcal{P}_{e_i}$, where \mathcal{P}_{e_i} represents a standard basis vector (one-hot vector) with the i -th component set to 1 and all other components set to 0. Then, we can simplify the aforementioned optimization problem by limiting the solution space to $\{\mathcal{P}_{e_i}\}_{i=0}^d$ and decide which token i should be selected. The corresponding gradient when $\mathcal{P} = \mathcal{P}_{e_i}$ has the following formulation.

Proposition 1. *The gradient of $KL(\mathcal{P}_{e_i}, \mathcal{P}_{logits})$ at $\text{logits} = \text{logits}_n$ is:*

$$\nabla_{\text{logits}_n} KL(\mathcal{P}_{e_i}, \mathcal{P}_{logits_n}) = (\mathcal{P}_{logits_n} - \mathcal{P}_{e_i})/\tau = (p_{(1,n)}, \dots, p_{(i,n)} - 1, \dots, p_{(d,n)})/\tau \quad (4)$$

We calculate the cosine similarity between the gradient $\nabla_{\text{logits}_n} KL(\mathcal{P}_{e_i}, \mathcal{P}_{logits_n})$ and the difference $\text{logits}_n - \text{logits}_N$ for each token in the vocabulary \mathcal{V} . Then we select the $\mathcal{P}_{e_{i^*}}$ of which the gradient

is closest to $\logits_n - \logits_N$ as the estimation $\mathcal{P}_{latent}^{(n)}$. Mathematically, this involves selecting i^* according to the following criterion

$$i^* = \arg \max_{1 \leq i \leq d} \bar{m}_i^{(n)}, \text{ where } \bar{m}_i^{(n)} = \max (\text{CosSim}(\logits_n - \logits_N, \mathcal{P}_{\logits_n} - \mathcal{P}_{e_i}), 0),$$

and adopting $\mathcal{P}_{latent}^{(n)} = \mathcal{P}_{e_{i^*}}$ as the "hard estimation" of \mathcal{P}_{real} . Drawing from the concept of hard and soft targets in label smoothing and knowledge distillation, we further extend it to the "soft estimation",

$$\mathcal{P}_{latent}^{(n)} = (m_1^{(n)}, \dots, m_i^{(n)}, \dots, m_d^{(n)}) / m^{(n)}, \text{ where } m_i^{(n)} = (\bar{m}_i^{(n)})^2 \text{ and } m^{(n)} = \sum_{i=1}^d m_i^{(n)}$$

We square $\{\bar{m}_i^{(n)}\}$ to moderately amplify their differences. Prior studies prove that soft targets usually offer stronger generalization capabilities, more information, and more robustness to noise than hard targets [4, 15, 23, 28, 29]. Hence, we adopt the soft estimation in lieu of the hard estimation.

Phase 2: We ensemble $\mathcal{P}_{latent}^{(n)}$ across all layers by computing a weighted average of the set $\{\mathcal{P}_{latent}^{(n)}\}$ and adopt it as the final estimation of the \mathcal{P}_{latent} :

$$\mathcal{P}_{latent} = \sum_{n=0}^N s^{(n)} \mathcal{P}_{latent}^{(n)}, \text{ where } s^{(n)} = m^{(n)} / (\sum_{n=0}^N m^{(n)})$$

This estimation suggests that the weight $s^{(n)}$ of certain layer n will be larger if the corresponding gradient approximation $\logits_n - \logits_N$ is more closely aligned with the gradients $\{\nabla_{\logits_n} KL(\mathcal{P}_{e_i}, \mathcal{P}_{\logits_n})\}$ for the tokens in the vocabulary. This in turn amplifies the influence of layer n on the final estimation, which is a desirable effect in our method. Figure 3 demonstrates that SLED can downplay incorrect tokens based on the gradient alignment. One can further validate that for each component m_i in the final estimation $\mathcal{P}_{latent} \triangleq (m_1, m_2, \dots, m_d)$, the following relationship holds: $m_i = \sum_{n=0}^N m_i^{(n)} / (\sum_{n=0}^N \sum_{j=1}^d m_j^{(n)})$. This property simplifies the description in Algorithm 1.

Phase 3: Applying \mathcal{P}_{latent} in Equation 2 enables us to derive the gradient necessary for steering the self-evolution on the final layer's logits \logits_N .

Proposition 2. *The gradient of $KL(\mathcal{P}_{latent}, \mathcal{P}_{\logits})$ at $\logits = \logits_N$ is:*

$$\nabla_{\logits_N} KL(\mathcal{P}_{latent}, \mathcal{P}_{\logits_N}) = (\mathcal{P}_{\logits_N} - \mathcal{P}_{latent}) / \tau = (p_{(1,N)} - m_1, \dots, p_{(d,N)} - m_d) / \tau$$

Then we can derive the self-evolved logits $\widetilde{\logits}_N$

$$\widetilde{\logits}_N \triangleq (\tilde{\ell}_{(1,N)}, \dots, \tilde{\ell}_{(i,N)}, \dots, \tilde{\ell}_{(d,N)}), \text{ where } \tilde{\ell}_{(i,N)} = \ell_{(i,N)} - \alpha(p_{(i,N)} - m_i) / \tau. \quad (5)$$

2.4 Computational Complexity and Design Decisions

For each layer, computing $\text{CosSim}(\logits_n - \logits_N, \mathcal{P}_{\logits_n} - \mathcal{P}_{e_i})$ for every token v_i in the vocabulary \mathcal{V} needs $\mathcal{O}(d^2)$ operations. To reduce the computational complexity, we select only a subset \mathcal{V}_{I_k} , where the token $v_i \in \mathcal{V}_{I_k}$ has the top- k highest logits in the final layer. In this scenario, we only initiate the self-evolution in Equation 2 of the logits corresponding to these top- k tokens. For the remaining tokens, which have lower probabilities, their logits are adjusted to a very lower numerical value, e.g., -1000 . This strategy significantly reduces the computational complexity, while maintaining focus on the most relevant tokens. We name the parameter k , as **Evolution Scale**, since it determines the number of top-probability tokens active for self-evolution.

Q 2.1: Why SLED contrast the final layer with all the early layers, instead of picking one premature layer to contrast based on JSD?

DoLa selects a subset of early layers to form a candidate set. Then it calculates the Jensen-Shannon Divergence (JSD) between the final layer and each layer in this set. Their strategy is to choose the layer with the highest JSD as the premature layer, and the chosen layer will be contrasted with the final layer to update probabilities. Obviously, if this strategy is reasonable, a larger candidate set should lead to a better choice of the premature layer and, consequently, enhanced overall performance. However, a paradoxical finding from their experimental results, which our tests also confirm, is that a

Question (The ground truth is \$460)

Eliza's rate per hour for the first 40 hours she works each week is \$10. She also receives an overtime pay of 1.2 times her regular hourly rate. If Eliza worked for 45 hours this week, how much are her earnings for this week?

Without SLED

Eliza's regular hourly rate is \$10. For 40 hours, her earnings are $40 \times \$10 = \400 . For 5 hours of overtime, her earnings are $5 \times \$10 = \50 . So her total earnings for the week are $\$400 + \$50 = \$450$. The answer is \$450. (Wrong)

With SLED

Eliza's regular hourly rate is \$10. For 40 hours, her earnings are $40 \times \$10 = \400 . For 5 hours of overtime, her earnings are $5 \times \$10 \times 1.2 = \60 . So her total earnings for the week are $\$400 + \$60 = \$460$. The answer is \$460. (Correct)

	t_{-2}	t_{-1}	t_0	t_1	t_2	t_3
$\mathcal{P}_{logits_{32}}$	1	0	=	\$	5	0
$\mathcal{P}_{latent}^{(31)}$	1	0	=	1	2	2
$\mathcal{P}_{latent}^{(26)}$	1	0	x	1	.	2
$\mathcal{P}_{latent}^{(21)}$	1	0	x	1	.	2
$\mathcal{P}_{latent}^{(16)}$	1	0	x	1	.	2
$\mathcal{P}_{latent}^{(11)}$	1	0	x	1	.	2
$\mathcal{P}_{latent}^{(6)}$	1	0	x	1	.	2
$\mathcal{P}_{latent}^{(1)}$	1	0	x	1	.	2

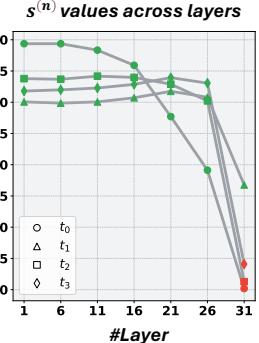


Figure 3: An example from GSM8K demonstrating SLED’s mechanism. SLED derives the estimations $\mathcal{P}_{latent}^{(n)}$ by contrasting final layer’s logits $logits_N$ with early layers’ logits $\{logits_n\}$. We list the token with the highest probability value from the $\mathcal{P}_{latent}^{(n)}$ for different early layers. As shown, SLED downplays incorrect tokens by assigning lower weights $s^{(n)}$ to the corresponding $\mathcal{P}_{latent}^{(n)}$. Conversely, if the estimation is correct, the weights are relatively larger. The parameter evolution scale is set to 2.

larger candidate set for DoLa leads to decreased performance. As shown in [2], when the candidate set for DoLa ranged from 0-32 layers, the performance was inferior compared to a smaller set of 0-16 layers. This fundamental flaw indicates that selecting a good candidate set remains a challenge when applying DoLa. In contrast, our method does not face this concern as it applies an ensemble approach to all early layers. It is also important to note that our method works well even when only contrasting the final layer with part of the early layers, as demonstrated in the results, proving the robustness of our approach.

Q 2.2: Why not use \mathcal{P}_{latent} directly as the model’s output distribution?

It is crucial to understand that \mathcal{P}_{latent} is merely an estimation of the real-world distribution based on the model’s latent knowledge instead of the exact \mathcal{P}_{real} . Consequently, relying solely on \mathcal{P}_{latent} , similar to DoLa, might lead to inaccuracies, as the latent knowledge can be imperfect. The original logits $logits_N$ are still important as they are refined directly by real-world data during training. The evolution rate α in Equation 2, serves to balance this trade-off, enabling a reciprocal enhancement between \mathcal{P}_{latent} and the original $logits_N$. More ablation studies are provided.

Q 2.2: Considering that SLED adopts $logits_n - logits_N$ as the estimation of the gradient, why not directly apply it in Equation 2?

It is important to note that while $logits_n - logits_N$ is unconstrained, the gradients estimated in Equation 2 (e.g., $p_{(1,N)} - m_1, \dots, p_{(d,N)} - m_d$) are constrained within $[-1, 1]$. Thus, direct substitution could lead to a mismatch in magnitudes and might also introduce unexpected noise. Proper normalization and subsequent aggregation of estimations from different layers are precisely what our method addresses in Section 2.2 and 2.3. Further analysis is provided.

3 Experiments

As a novel layer-wise contrastive decoding approach, we first benchmark SLED against the state-of-the-art approach DoLa [2] across a diverse range of model families (Gemma, Qwen, Mixtral, gpt-oss) and model scales (from 2B to 45B), including the more advanced mixture of experts (MoE) architecture, as detailed in Section 3.2. The results showcase notable factuality improvements across a variety of tasks, including multi-choice, open-generation, and adaptations to chain-of-thought reasoning tasks.

3.1 Experimental Setup

Benchmarks We compare our method with baselines on several multiple-choice. For multiple-choice question tasks, we use the TruthfulQA [13] and FACTOR (Wiki) [14] datasets to assess the LLMs’ factuality in short-answer/long-paragraph scenario, respectively.

Table 1: Comparison on Gemma-3 model family. The best results are in bold for each dataset/metric. SLED outperforms DoLa and vanilla greedy decoding.

Model	FACTOR	TruthfulQA			Model	FACTOR	TruthfulQA		
		MC1	MC2	MC3			MC1	MC2	MC3
Gemma-3-1B-PT	47.83	32.28	63.22	30.98	Gemma-3-12B-PT	66.67	35.82	63.70	34.83
+DoLa	12.63	35.44	68.90	35.29		7.28	33.54	66.62	34.25
+SLED (ours)	63.29	35.57	71.79	40.69		74.25	37.85	67.05	34.59
Gemma-3-1B-IT	37.17	33.29	61.65	30.74	Gemma-3-12B-IT	59.12	39.49	63.62	37.89
+DoLa	16.50	36.46	68.86	35.69		11.06	35.19	67.53	35.04
+SLED (ours)	55.04	36.71	71.79	42.00		70.81	46.33	71.15	40.90
Gemma-3-4B-PT	58.78	33.80	64.12	32.64	Gemma-3-27B-PT	72.04	36.20	61.39	34.95
+DoLa	9.25	34.68	68.21	34.90		55.88	31.65	65.03	32.72
+SLED (ours)	69.94	35.19	69.25	38.96		78.12	38.48	67.40	35.65
Gemma-3-4B-IT	50.17	36.96	58.42	34.13	Gemma-3-27B-IT	66.00	41.14	64.06	38.71
+DoLa	10.79	36.08	68.59	35.58		47.43	33.16	65.93	33.63
+SLED (ours)	66.15	45.19	66.15	38.06		73.75	47.47	73.58	43.53

Models & Baselines We evaluate the performance of SLED on eight Gemma-3 models [22] (<{1B,4B,12B,27B}-PT, {1B,4B,12B,27B}-IT), two Gemma-1 models (2B,7B), two MoE models (Mixtral-8×7B, Mixtral-8×7B-IT) [7], one Qwen-3 Model and one OpenAI gpt-oss-20b model. We adopt the following baselines: 1) standard decoding (greedy decoding or sampling depending on the tasks), 2) DoLa [2].

Metrics We adopt the factual accuracy evaluation implemented in [2] for multiple-choice tasks.

3.2 Evaluation on a Broad Range of LLM Benchmarks

The objective of these tasks is to employ decoding methods that enable LLMs to assign higher probabilities to correct completions/answers over incorrect alternatives. We demonstrate the effectiveness of SLED for both Short-Answer Factuality on the TruthfulQA and Long-Paragraph Factuality on the FACTOR dataset. For both DoLa and our SLED, we contrast the results from the final layer against all preceding layers. We randomly sample approximately 5% of the data for validation regarding parameter selection. The results, as shown in Table 1, indicate that SLED achieves superior outcomes in almost all metrics across 8 Gemma-3 models. Notably, SLED achieves better performance under the MC1/MC3 metrics on TruthfulQA, which are more sensitive to fluctuations and pose a greater challenge. For long sentences in FACTOR, our method shows improvements over baselines by 5-13%. These results not only underscore the benefits of our method for factuality but also demonstrate its robustness across different lengths of text.

3.3 Evaluation Across Diverse LLM Configurations

As discussed above and shown in Table 1, our method, SLED, demonstrates strong generalization capabilities across the Gemma-3 model family, proving robust from 1B to 27B model sizes. In Table 2, we further showcase SLED’s impressive performance on the other family models, such as Mixtral, Qwen-3, gpt-oss models, in terms of long paragraph factuality and short answer factuality. Interestingly, SLED is also applicable to the increasingly popular Mixture of Experts (MoE) architectures. These results confirm the exceptional adaptability of our method across various LLM configurations.

4 Conclusion

We introduced Self Logits Evolution Decoding (SLED), which is a new method to improve accuracy and factuality without requiring external knowledge (e.g., RAG) or fine-tuning (e.g., SFT). The key idea is to optimize the output logits based on the LLMs’ latent knowledge to improve factuality during inference. On several datasets, SLED achieved the SOTA results, improving over the vanilla decoding and DoLa. SLED does not increase the inference time significantly, and it can be combined with other factuality decoding methods. For future work, it would be interesting to combine SLED with supervised fine-tuning methods, e.g., to adapt to other domains.

Table 2: Using SLED with other LLM families also improves the factuality.

Model	FACTOR	TruthfulQA			Model	FACTOR	TruthfulQA		
		MC1	MC2	MC3			MC1	MC2	MC3
Gemma-2B	50.87	23.38	37.16	17.42	Mixtral-8×7B	71.41	35.13	49.98	34.17
+DoLa	32.93	26.07	48.97	26.55		58.28	32.44	35.91	33.68
+SLED (ours)	57.05	25.21	50.20	26.94		74.92	35.86	57.26	32.96
Gemma-7B	60.42	31.58	47.63	22.75	Mixtral-8×7B-IT	70.51	37.94	62.51	35.25
+DoLa	36.07	25.21	43.14	26.13		56.15	32.19	39.17	33.76
+SLED (ours)	65.56	32.31	49.88	25.22		75.55	41.73	68.52	37.70
gpt-oss-20b	41.12	34.43	67.24	34.41	Qwen-3-14B-Base	57.69	38.10	68.65	36.16
+DoLa	43.59	28.99	61.72	30.33		58.42	34.43	65.09	33.17
+SLED (ours)	55.31	36.71	67.69	35.26		64.09	40.00	68.99	36.23

Acknowledgment

This work was done when Jianyi Zhang was an intern at Google Research. In addition, Jianyi Zhang and Yiran Chen disclose the support from grants NSF CNS-2112562 and ARO W911NF-23-2-0224. We thank area chair and reviewers for their valuable comments.

References

- [1] Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*, 2023.
- [2] Yung-Sung Chuang, Yujia Xie, Hongyin Luo, Yoon Kim, James R. Glass, and Pengcheng He. Dola: Decoding by contrasting layers improves factuality in large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=Th6NyL07na>.
- [3] Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate, 2024. URL <https://openreview.net/forum?id=QAwaalJNCK>.
- [4] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015.
- [5] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qian-glong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, et al. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *arXiv preprint arXiv:2311.05232*, 2023.
- [6] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38, 2023.
- [7] Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mixtral of experts, 2024. URL <https://arxiv.org/abs/2401.04088>.
- [8] Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, et al. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022.

- [9] Deren Lei, Yaxi Li, Mengya Hu, Mingyu Wang, Vincent Yun, Emily Ching, Eslam Kamal, et al. Chain of natural language inference for reducing large language model ungrounded hallucinations. *arXiv preprint arXiv:2310.03951*, 2023.
- [10] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.
- [11] Kenneth Li, Oam Patel, Fernanda Viégas, Hanspeter Pfister, and Martin Wattenberg. Inference-time intervention: Eliciting truthful answers from a language model. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 41451–41530. Curran Associates, Inc., 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/81b8390039b7302c909cb769f8b6cd93-Paper-Conference.pdf.
- [12] Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, Tatsunori Hashimoto, Luke Zettlemoyer, and Mike Lewis. Contrastive decoding: Open-ended text generation as optimization. *arXiv preprint arXiv:2210.15097*, 2022.
- [13] Stephanie Lin, Jacob Hilton, and Owain Evans. TruthfulQA: Measuring how models mimic human falsehoods. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3214–3252, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.229. URL <https://aclanthology.org/2022.acl-long.229>.
- [14] Dor Muhlgay, Ori Ram, Inbal Magar, Yoav Levine, Nir Ratner, Yonatan Belinkov, Omri Abend, Kevin Leyton-Brown, Amnon Shashua, and Yoav Shoham. Generating benchmarks for factuality evaluation of language models. *arXiv preprint arXiv:2307.06908*, 2023.
- [15] Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. When does label smoothing help? *Advances in neural information processing systems*, 32, 2019.
- [16] OpenAI. Introducing chatgpt. <https://openai.com/blog/chatgpt/>, November 2022.
- [17] OpenAI. GPT-4 Technical Report. *arXiv e-prints*, art. arXiv:2303.08774, March 2023. doi: 10.48550/arXiv.2303.08774.
- [18] OpenAI. gpt-oss-120b & gpt-oss-20b model card. Technical report, OpenAI, August 2025.
- [19] William Saunders, Catherine Yeh, Jeff Wu, Steven Bills, Long Ouyang, Jonathan Ward, and Jan Leike. Self-critiquing models for assisting human evaluators. *arXiv preprint arXiv:2206.05802*, 2022.
- [20] Chufan Shi, Haoran Yang, Deng Cai, Zhisong Zhang, Yifan Wang, Yujiu Yang, and Wai Lam. A thorough examination of decoding methods in the era of llms. *arXiv preprint arXiv:2402.06925*, 2024.
- [21] Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricu, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.
- [22] Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ramé, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, Gaël Liu, Francesco Visin, Kathleen Kenealy, Lucas Beyer, Xiaohai Zhai, Anton Tsitsulin, Robert Busa-Fekete, Alex Feng, Noveen Sachdeva, Benjamin Coleman, Yi Gao, Basil Mustafa, Iain Barr, Emilio Parisotto, David Tian, Matan Eyal, Colin Cherry, Jan-Thorsten Peter, Danila Sinopalnikov, Surya Bhupatiraju, Rishabh Agarwal, Mehran Kazemi, Dan Malkin, Ravin Kumar, David Vilar, Idan Brusilovsky, Jiaming Luo, Andreas Steiner, Abe Friesen, Abhanshu Sharma, Abheesh Sharma, Adi Mayrav Gilady, Adrian Goedeckemeyer, Alaa Saade, Alex Feng, Alexander Kolesnikov, Alexei Bendebury,

Alvin Abdagic, Amit Vadi, András György, André Susano Pinto, Anil Das, Ankur Bapna, Antoine Miech, Antoine Yang, Antonia Paterson, Ashish Shenoy, Ayan Chakrabarti, Bilal Piot, Bo Wu, Bobak Shahriari, Bryce Petrin, Charlie Chen, Charline Le Lan, Christopher A. Choquette-Choo, CJ Carey, Cormac Brick, Daniel Deutsch, Danielle Eisenbud, Dee Cattle, Derek Cheng, Dimitris Paparas, Divyashree Shivakumar Sreepathihalli, Doug Reid, Dustin Tran, Dustin Zelle, Eric Noland, Erwin Huizenga, Eugene Kharitonov, Frederick Liu, Gagik Amirkhanyan, Glenn Cameron, Hadi Hashemi, Hanna Klimczak-Plucińska, Harman Singh, Harsh Mehta, Harshal Tushar Lehri, Hussein Hazimeh, Ian Ballantyne, Idan Szpektor, Ivan Nardini, Jean Pouget-Abadie, Jetha Chan, Joe Stanton, John Wieting, Jonathan Lai, Jordi Orbay, Joseph Fernandez, Josh Newlan, Ju yeong Ji, Jyotinder Singh, Kat Black, Kathy Yu, Kevin Hui, Kiran Vodrahalli, Klaus Greff, Linhai Qiu, Marcella Valentine, Marina Coelho, Marvin Ritter, Matt Hoffman, Matthew Watson, Mayank Chaturvedi, Michael Moynihan, Min Ma, Nabila Babar, Natasha Noy, Nathan Byrd, Nick Roy, Nikola Momchev, Nilay Chauhan, Noveen Sachdeva, Oskar Bunyan, Pankil Botarda, Paul Caron, Paul Kishan Rubenstein, Phil Culliton, Philipp Schmid, Pier Giuseppe Sessa, Pingmei Xu, Piotr Stanczyk, Pouya Tafti, Rakesh Shivanna, Renjie Wu, Renke Pan, Reza Rokni, Rob Willoughby, Rohith Vallu, Ryan Mullins, Sammy Jerome, Sara Smoot, Sertan Girgin, Shariq Iqbal, Shashir Reddy, Shruti Sheth, Siim Pöder, Sijal Bhatnagar, Sindhu Raghuram Panyam, Sivan Eiger, Susan Zhang, Tianqi Liu, Trevor Yacovone, Tyler Liechty, Uday Kalra, Utku Evci, Vedant Misra, Vincent Roseberry, Vlad Feinberg, Vlad Kolesnikov, Woohyun Han, Woosuk Kwon, Xi Chen, Yinlam Chow, Yuvein Zhu, Zichuan Wei, Zoltan Egyed, Victor Cotrata, Minh Giang, Phoebe Kirk, Anand Rao, Kat Black, Nabila Babar, Jessica Lo, Erica Moreira, Luiz Gustavo Martins, Omar Sanseviero, Lucas Gonzalez, Zach Gleicher, Tris Warkentin, Vahab Mirrokni, Evan Senter, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, Yossi Matias, D. Sculley, Slav Petrov, Noah Fiedel, Noam Shazeer, Oriol Vinyals, Jeff Dean, Demis Hassabis, Koray Kavukcuoglu, Clement Farabet, Elena Buchatskaya, Jean-Baptiste Alayrac, Rohan Anil, Dmitry, Lepikhin, Sebastian Borgeaud, Olivier Bachem, Armand Joulin, Alek Andreev, Cassidy Hardin, Robert Dadashi, and Léonard Hussenot. Gemma 3 technical report, 2025. URL <https://arxiv.org/abs/2503.19786>.

- [23] Christian Thiel. Classification on soft labels is robust against label noise. In Ignac Lovrek, Robert J. Howlett, and Lakhmi C. Jain, editors, *Knowledge-Based Intelligent Information and Engineering Systems*, pages 65–73, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. ISBN 978-3-540-85563-7.
- [24] Katherine Tian, Eric Mitchell, Huaxiu Yao, Christopher D Manning, and Chelsea Finn. Fine-tuning language models for factuality. In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=WPZ2yPag4K>.
- [25] Chenguang Wang, Xiao Liu, and Dawn Song. Language models are open knowledge graphs. *arXiv preprint arXiv:2010.11967*, 2020.
- [26] Sean Welleck, Amanda Bertsch, Matthew Finlayson, Hailey Schoelkopf, Alex Xie, Graham Neubig, Ilia Kulikov, and Zaid Harchaoui. From decoding to meta-generation: Inference-time algorithms for large language models. *arXiv preprint arXiv:2406.16838*, 2024.
- [27] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengan Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL <https://arxiv.org/abs/2505.09388>.
- [28] Chang-Bin Zhang, Peng-Tao Jiang, Qibin Hou, Yunchao Wei, Qi Han, Zhen Li, and Ming-Ming Cheng. Delving deep into label smoothing. *IEEE Transactions on Image Processing*, 30: 5984–5996, 2021.
- [29] Jianyi Zhang, Aashiq Muhamed, Aditya Anantharaman, Guoyin Wang, Changyou Chen, Kai Zhong, Qingjun Cui, Yi Xu, Belinda Zeng, Trishul Chilimbi, and Yiran Chen. ReAugKD:

- Retrieval-augmented knowledge distillation for pre-trained language models. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1128–1136, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-short.97. URL <https://aclanthology.org/2023.acl-short.97>.
- [30] Yue Zhang, Leyang Cui, Wei Bi, and Shuming Shi. Alleviating hallucinations of large language models through induced hallucinations. *arXiv preprint arXiv:2312.15710*, 2023.
 - [31] Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lema Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, et al. Siren’s song in the ai ocean: a survey on hallucination in large language models. *arXiv preprint arXiv:2309.01219*, 2023.