Hallucination Detox: Sensitivity Dropout (SenD) for Large Language Model Training

Shahrad Mohammadzadeh[1, 4, *], **Juan David Guerra**[2, 4, *], **Marco Bonizzato**[2, 3, 4], **Reihaneh Rabbany**[1, 4, 5], **Golnoosh Farnadi**[1, 3, 4, 5]

[1]McGill University, [2]Polytechnique Montréal, [3]Université de Montréal, [4]Mila - Quebec Artificial Intelligence Institute [5]CIFAR AI Chair

shahrad.mohammadzadeh@mail.mcgill.ca, juan-2.guerra@polymtl.ca

Abstract

As large language models (LLMs) become increasingly prevalent, concerns about their reliability, particularly due to hallucinations - factually inaccurate or irrelevant outputs - have grown. Our research investigates the relationship between the uncertainty in training dynamics and the emergence of hallucinations. Using models from the Pythia suite and several hallucination detection metrics, we analyze hallucination trends and identify significant variance during training. To address this, we propose **Sensitivity Dropout (SenD)**, a novel training protocol designed to reduce hallucination variance during training by deterministically dropping embedding indices with significant variability. In addition, we develop an unsupervised hallucination detection metric, Efficient EigenScore (EES), which approximates the traditional EigenScore in 2x speed. This metric is integrated into our training protocol, allowing SenD to be both computationally scalable and effective at reducing hallucination variance. SenD improves test-time reliability of Pythia and Meta's Llama models by up to 17% and enhances factual accuracy in Wikipedia, Medical, Legal, and Coding domains without affecting downstream task performance.

1 Introduction

1.1 Motivation

As Large Language Models (LLMs) become more sophisticated and widespread across industries, concerns about their reliability and safety have grown due to misuse and user errors. One of these concerning areas discovered by the scientific community is the phenomenon of hallucinations - LLMs producing content that may not align with real-world facts, the user's input, or training data it has seen in the past (Huang et al., 2023a). In our research we target confabulations, hallucinations which occur when the LLM generates different responses given the same or similar inputs. This can be harmful when the generations alter between correct and factually incorrect responses.

Previous research has largely focused on identifying and addressing hallucinations in large language models (LLMs), but the impact of the training process on hallucinations remains underexplored (Huang et al., 2023a; Rawte et al., 2023; Ye et al., 2023; Hong et al., 2024; Xu et al., 2024; Chen et al., 2024; Li et al., 2024; Gao et al., 2024c). This paper investigates how iterative learning in LLMs causes significant variance in hallucination behavior, leading to fluctuating prediction confidence and making it difficult to identify a checkpoint where the model reliably learns facts.

To explore these hallucination trends, we analyze models ranging from 70 million to 12 billion parameters within the Pythia suite (Biderman et al., 2023), assessing them across various training checkpoints and tasks. Our goal is to validate the oscillatory behavior observed by Li et al. (2024) through evaluation metrics including HaluEval (Li et al., 2023), FactScore (Min et al., 2023), SelfCheckGPT (Manakul et al., 2023), and XSum (Narayan et al., 2018). Utilizing the reliability of internal model dynamics for quantifying hallucination likelihood, we use EigenScore (Chen et al., 2024) and Semantic Entropy (Kossen et al., 2024) to detect hallucination risk by analyzing variability in high-temperature outputs. Experiments utilize EigenScore and the HELM dataset (Su et al., 2024) to identify hallucinations during training.

We introduce **Sensitivity Dropout** (SenD), a novel training protocol that prioritizes confident learning over mere loss minimization. SenD reduces hallucination variance by selectively dropping Sensitive Embedding Indices,—those exhibiting significant fluctuations throughout training—improving model certainty and providing a reliable stopping criterion for training. To enhance effi-

[*]Equal Contribution

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 5538–5554

ciency, we propose **Efficient EigenScore** (EES), a scalable alternative to EigenScore (Chen et al., 2024) for hallucination detection, maintaining high correlation while reducing computational costs.

Our contributions to the field, emphasizing that SenD enhances the training process and **does not replace** post-hoc solutions, [1] can be summarized as follows:

1. Empirical verification of the **hallucinatory oscillation in LLM training** across various model scales and detection metrics.

2. **Sensitivity Dropout (SenD)**, a novel training paradigm designed to reduce hallucination variance and increase model confidence during training.

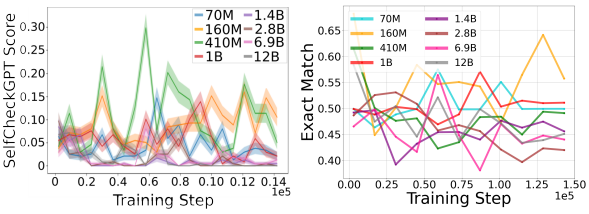
3. **Efficient EigenScore (EES)**, an efficient hallucination detection metric used to keep SenD efficient, achieving up to 2x speedup with minimal effects on accuracy.

1.2 Related Work

The majority of research on hallucinations in language models has focused on detecting and mitigating this phenomenon rather than explaining its underlying causes. Recent techniques can be categorized into two main approaches: those based on output probabilities at inference time (Manakul et al., 2023; Joshi et al., 2017; Li et al., 2023) and those that utilize internal representations or hidden layers of the model (Su et al., 2024; Chen et al., 2024; Kossen et al., 2024). While the former has shown effectiveness, the latter offers deeper insights, but often comes with computational trade-offs. Additionally, methods like Reinforcement Learning with Human Feedback (RLHF) have gained traction for enhancing model reliability (Yu et al., 2024). However, many of these post-hoc solutions enhance factual accuracy by layering algorithms atop pre-trained models, which can be inefficient. Our work addresses this gap by focusing on the internal dynamics of the model that contribute to hallucinations.

Regularization techniques have been introduced to fix the issue of variability, notably random neuron dropout, used to reduce the variance and ensure that no neuron is overpowering others (Srivastava et al., 2014; Baldi and Sadowski, 2013). Work such as that done by Santra et al. (2020); Ba and Frey (2013) aims to modify random neuron dropout to change the way neurons are dropped to a more deterministic, precise manner. This has allowed the authors to drop unimportant connections in a deep neural network to ensure that class discriminative information is propagated through the model correctly (Santra et al., 2020). Inspired by this, our aim is to target hallucinatory embedding indices in our models to ensure that information is learnt with certainty. State-of-the-art hallucination metrics, especially those based on internal model dynamics, rely on spectral analysis and embedding matrix computations. Methods like EigenScore (Chen et al., 2024) and Semantic Entropy (Kossen et al., 2024) effectively assess hallucination risk but require multiple inferences, making them computationally demanding as models scale. Tools such as the Density of States (DOS) and the kernel polynomial method (KPM) have been explored to approximate spectral properties efficiently (Huang et al., 2023b; Lin et al., 2014). Building on these advancements, our work integrates efficient spectral analysis methods into hallucination detection, demonstrated through EES and SenD.

2 Oscillatory Behaviour Validation



(a) Self-Consistency (b) QA

Figure 1: **Visualization of Oscillatory Behavior**. (a) SelfCheckGPT and (b) HaluEval EM metrics across model sizes from 70M-12B. Solid lines show average performance, shaded regions indicate standard deviation. High variance in hallucination metrics highlights the need for stabilization. For Perplexity (PPL), Rouge1 and other HaluEval metrics refer to Appendix A.2.

Transformer training checkpoints are vital for understanding learning dynamics. Our analysis shows that converged training loss does not necessarily reduce hallucinations, confirming Li et al. (2024)'s observations on LLM oscillatory hallucination behavior. We leverage Eleuther AI's Pythia and LMEval tools (Biderman et al., 2023; Gao et al., 2024a) to study 16 LLMs (ranging from 70M to 12B parameters) across 20 evenly spaced checkpoints. At each checkpoint, we evaluate the models

[1] For the code and datasets used, refer to our GitHub repository at: https://github.com/EMZEDI/SEND.

on various hallucination metrics: SelfCheckGPT for self-consistency (Manakul et al., 2023), XSum for summarization (Narayan et al., 2018), perplexity, and HaluEval for QA tasks (Li et al., 2023). As state-of-the-art (SOTA) methods, we select to present SelfCheckGPT and HaluEval QA Exact Match in Figure 1 as representatives of the set of introduced metrics as they exhibit similar behaviours to the rest of the metrics in Appendix A.2. Higher SelfCheckGPT scores indicate more self-contradiction, higher Rouge1 on XSum suggests better summary alignment, lower perplexity implies greater prediction confidence, and improved QA scores reflect better overall performance.

RQ1: How do the established iterative training processes and model complexity influence LLM hallucinations? The analysis of hallucination oscillations, as shown in Figure 1, indicates a consistent pattern across different models: oscillations persist throughout training from the initial to the final checkpoint. This finding highlights the uncertainty of halting training solely based on the convergence of training loss. For instance, in QA settings, the optimal Exact Match of the outputs with ground truths is achieved in earlier checkpoints. This observation is seen more dramatically in Figure 1a, where the size of the model has almost no effect on the performance of SelfCheckGPT. Instead, we observe oscillatory behaviour within self-consistency, implying that model size is not much effective at tackling the issue of confabulations verified by results in Appendix A.2 as well. These results suggest that optimizing solely for the loss in training is not sufficient. We also see that beyond a certain point, larger models do not significantly reduce hallucinations, indicating that scaling alone is not sufficient for building robust models. Instead, more refined approaches are needed to address the underlying variability in model behavior.

3 Internal Training Dynamics

Following our investigation of the oscillatory behaviour in training, we look into the internal states of the Pythia 1B model (Biderman et al., 2023) to see what information we are able to extract. In doing so, we establish a series of definitions and metrics in order to understand the internal processes during the training of LLMs. This information is later used in Sections 3.2 and 4 to assist us in deriving methods for improving the variance in the hallucinatory behaviour of models during training.

3.1 Sensitive Embedding Indices

To start our analysis of the internal states, we employ Su et al. (2024)'s sentence embedding extraction approach given its demonstrated success in hallucination detection. We convert the activation matrix of the model into a sentence embedding vector (Definition 3.1) which turns an $\mathbb{R}^{n,m}$ activation matrix into a sentence embedding vector a_k for input k with dimension $\mathbb{R}^n$.

Definition 3.1 (Sentence Embedding Vector)**.** The Sentence Embedding Vector is a way to convert the large $\mathbb{R}^{n,m}$ activation matrix into a smaller, easier to manage vector with dimension $\mathbb{R}^n$.

$$e_k = \frac{1}{2}((\frac{1}{m}\sum_{i=1}^{m} H^i_{N-1}) + H^m_{N-1}) \quad (1)$$

Where e_k is the activation of one input k, m is the number of tokens in the sequence, H is the token embedding activation matrix, and $N-1$ is the subtraction to get the penultimate layer index and the formula is adapted from Su et al. (2024). The penultimate layer of the LLM, being the layer closest to the output probabilities, is our primary focus for hallucination analysis due to its rich information about output certainty.

Next, we define the Net Change Formula 3.2 as a way to extract information from the model indicative of oscillatory behaviour between checkpoints from the sentence embedding vector.

Definition 3.2 (Net Change Formula)**.** Let e^t_i denote the embedding of data point x at embedding index i of the contextual embedding after checkpoint t. Then we define the net change formula as

$$\Delta e^t_i = |e^t_i - e^{t-1}_i| \quad (2)$$

Building on these definitions, we now formalize the central focus of our investigation: **Sensitive Embedding Indices (SEIs)**, which we demonstrate play a critical role in the hallucination behavior of large language models (LLMs) (see Section 3.1). Specifically, SEIs can be leveraged to refine training procedures, reducing hallucination variability during training and improving overall confidence at inference time. Conceptually, SEIs correspond to indices within the sentence embedding (Definition 3.1) that exhibit significant fluctuations across training checkpoints, a phenomenon we hypothesize to be closely linked to the oscillatory nature of hallucination performance. Identifying the most

sensitive embedding indices involves selecting the top $K\%$ of indices for a given data point's representation. In our study, we set $K = 20$.

Definition 3.3 (Sensitive Embedding Indices - SEIs)**.** Indices of the contextual embedding for data point x which exhibit the highest net change across the last C checkpoints of training, indicating overall high variability during this period. This is calculated by

$$V_i = Var(e_i) \sum_{t=T-C+1}^{T} \Delta e_i^t \tag{3}$$

where V_i is the total variability during the last C checkpoints and the most sensitive embedding indices are

$$\mathbf{s} = \arg \max_{1 \le i \le N} \{V_i \mid V_i \ge \text{percentile}(V, 100 - k)\} \tag{4}$$

where N is the embedding vector size and k is the desired percentile threshold.

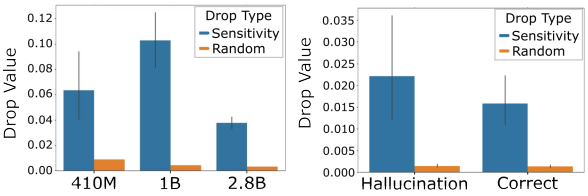
The aforementioned definition of Sensitive Embedding Indices (SEIs) is subsequently applied to LLM hallucinations through an analysis of EigenScores. Chen et al. (2024) introduce a novel metric for detecting confabulations, a specific subclass of hallucinations. Their approach computes an EigenScore (Definition 3.4) by leveraging determinant calculations derived from multiple LLM outputs generated under a high-temperature setting (*temperature* = 0.5), thereby encouraging greater output diversity. They hypothesize that when an LLM hallucinates, the resulting text exhibits increased semantic variability, leading to an elevated EigenScore. Notably, this method achieves SOTA performance while remaining unsupervised, as it relies solely on the model's learned representations. In the following sections, we examine the correlation between EigenScores at various training checkpoints and the most sensitive embedding indices associated with the corresponding data points.

Definition 3.4 (EigenScore)**.** The **EigenScore** of data point x indicates the degree of hallucination on input x by the average logarithm of the eigenvalues on the covariance matrix of the multiple output generations (typically 10 in our experiments).

$$ES = \mathbb{E}(Y \mid x, \theta) = \frac{1}{K} \sum_{i=1}^{K} \log(\lambda_i) \tag{5}$$

where $\lambda = \{\lambda_1, \ldots, \lambda_K\}$ denotes the eigenvalues of the regularized covariance matrix $\Sigma + \alpha \cdot \mathbb{I}$. we advise referring to Chen et al. (2024) for a more detailed analysis of this formula.

RQ2: What is the impact of Sensitive Embedding Index on EigenScores and hallucination in LLMs? To assess the correlation between SEIs and other indices in the embedding matrix of 10 generated outputs at a specific checkpoint, we conduct experiments aimed to determine if the presence of SEIs indicates higher uncertainty and a greater likelihood of hallucinations.



(a) Model Size (b) Output Type

Figure 2: **Comparison of sensitive embedding index dropout** on inference of Eleuther AI's Pythia modeld with random embedding index dropout. The Y axis, Drop Value, denotes the decrease in EigenScore, ie less confabulations, following the dropout method used. (a) SEI dropout results are consistent across model sizes. (b) Hallucinatory outputs show a larger EigenScore drop than correct ones. SEI dropout significantly reduces EigenScore compared to random dropout in both (a) and (b)

To evaluate the effect of SEIs on hallucination, we conduct experiments using the HELM dataset (Su et al., 2024), which is comprised of model-generated outputs from over 50,000 Wikipedia articles. This dataset was selected due to Wikipedia's role as a primary factual source.

To quantify the influence of SEIs, we extend the EigenScore method by applying it to sentence embeddings extracted from the penultimate layer of EleutherAI's Pythia 1B model (Biderman et al., 2023). Our analysis focuses on checkpoints between 133,000 and 143,000 training steps, where embeddings exhibit greater stability and the model demonstrates a higher level of language understanding compared to earlier training phases.

We implement SEI dropout by removing the top 10% of SEIs at each checkpoint and compare it to a baseline where 10% of embedding indices are randomly dropped. Furthermore, we examine the impact of SEI dropout on hallucination-prone versus non-hallucination-prone inputs to assess whether SEIs play a critical role in hallucination without adversely affecting factually correct outputs.

Given that a reduction in the EigenScore metric serves as a proxy for decreased hallucination likelihood, we adopt this metric as the primary evaluation measure in our study. Through a comparative analysis of baseline random embedding index dropout and SEI dropout, we demonstrate that SEI dropout significantly lowers the EigenScore across model sizes, thereby reducing confabulation probability (Figure 2a). Notably, while this reduction is most pronounced in hallucinatory outputs, we observe a decrease for correctly answered queries (Figure 2b), suggesting that our approach effectively modulates uncertainty without adversely impacting factual responses. Furthermore, our findings indicate that the internal states of the model play an important role in mitigating the generation of confabulated text across various model sizes.

3.2 Efficient EigenScore Approximation

Algorithm 1 Efficient EigenScore Algorithm

Require: Embedding matrix $E \in \mathbb{R}^{d_{\text{model}} \times K}$, number of Chebyshev terms M, number of stochastic trace estimation samples N_z
Ensure: Approximated EigenScore *EES*
Standardize and Scale the Embedding Matrix E**:**
$E_{\text{mean}} = \frac{1}{K}\sum_{i=1}^{K} E[:,i]$
$E_{\text{std}} = \sqrt{\frac{1}{K}\sum_{i=1}^{K}(E[:,i] - E_{\text{mean}})^2}$
$E_{\text{normalized}} = \frac{E - E_{\text{mean}}}{E_{\text{std}}}$ ▷ Normalize E with mean and standard deviation
$\sigma_{\max} = \text{Power Method}(E_{\text{normalized}})$ ▷ Compute largest singular value using the power method
$E_{\text{normalized}} \leftarrow \frac{E_{\text{normalized}}}{\sigma_{\max}}$ ▷ Scale E by $\sigma_{\max}$
Initialize:
$d_m = 0 \quad \forall m \in \{0, 1, \ldots, M\}$ ▷ Initialize d_m coefficients
$c_m = 0 \quad \forall m \in \{0, 1, \ldots, M\}$ ▷ Initialize c_m coefficients
Compute DOS coefficients d_m**:**
for $m = 0$ to M **do**
    Sample $z_j \sim \mathcal{N}(0, I)$ ▷ Sample random vectors for stochastic trace estimation
    Compute Chebyshev polynomial using the recurrence relation
end for
Compute Chebyshev coefficients c_m**:**
for $m = 0$ to M **do**
    $c_m \leftarrow \int_0^1 \log(\lambda) T_m^*(\lambda)\, d\lambda$ ▷ Using Equation 27 and Gaussian Quadrature for approximation
end for
Compute EigenScore:
EES $\leftarrow \frac{1}{K}\sum_{m=0}^{M} d_m c_m$ ▷ Approximate EigenScore using DOS coefficients
return *EES* ▷ Return the approximated EigenScore

If n denotes the hidden layer size, and computing the EigenScore for a single inference requires an eigen-decomposition with complexity on the order of $O(n^3)$. For T inferences, the overall computational cost scales as $O(T \cdot n^3)$, which quickly becomes prohibitive as both n and T increase. To address the computational complexity of EigenScore calculations, particularly as LLM hidden layer sizes increase, we develop an approximation method. This approximation, detailed in Algorithm 1, leverages the properties of Spectral Density or Density of States (DOS) to estimate EigenScore without explicitly constructing the covariance matrix. While this approximation provides a general overview of EigenScore trends, it is important to note that the output scales differ: EigenScore ranges from $[0, \infty)$, whereas the approximation, referred to as **Efficient EigenScore (EES)**, outputs values between $[-1, 1]$. Since the spectrum of the matrix is altered to make EES computable and operates on its own scale, EES can be seen as a standalone metric for hallucination detection.

The computation of the Efficient EigenScore (EES) is based on two fundamental concepts: Chebyshev Polynomials and Density of States (DOS). A detailed introduction to these concepts is provided in Appendix sections B.1 and B.2. Below, we outline a brief sketch of the derivation of EES. Since Chen et al. (2024) uses the covariance matrix of the embedding matrix of 10 sequences generated by the model in their methods, we represent it with H and use it in our derivation.

Lemma 1. *Let $f = \log$. Then, for a covariance matrix H with eigenvalues λ_i, we have*

$$\textit{trace}(\log(H)) = \sum_{i=1}^{N} \log(\lambda_i), \tag{6}$$

where λ_i are the eigenvalues of H.

Proposition 1. *Using the property of the density of states (DOS), we have:*

$$\int \log(\lambda)\, \mu(\lambda)\, d\lambda = \log\left(\prod_{i=1}^{N} \lambda_i\right), \tag{7}$$

which follows from Lemma 1 since $\sum_{i=1}^{N} \log(\lambda_i) = \log\left(\prod_{i=1}^{N} \lambda_i\right)$.

Note that from Proposition 1, the integral is equal to $N.EigenScore(H)$ or in our application, given C the integral equals $K.EigenScore(C)$, K being the number of model generations.

Our objective is to simplify the integral and approximate its value, avoiding the direct computation of the covariance matrix. This approach is

intended to mitigate the computational complexity and associated costs of explicitly handling the covariance matrix. Further utilizing Chebyshev Polynomials, DOS, and KPM (as introduced in Appendix B.2), we can simplify the integral mentioned in Equation 7 to $\sum_{m=0}^{M} d_m c_m$, where d_m term in DOS is approximated using Stochastic Trace Estimation and c_m m'th Chebyshev Polynomial coefficient. Appendices B.3 and B.4 provide the derivation of this equation. Note that the simplified integral is ultimately used to approximate the EigenScore of the matrix which is ultimately equivalent to $\frac{1}{K}\sum_{m=0}^{M} d_m c_m$. Performance of EES approximation is closely correlated with that of the original EigenScore which can be seen in Figure 10.

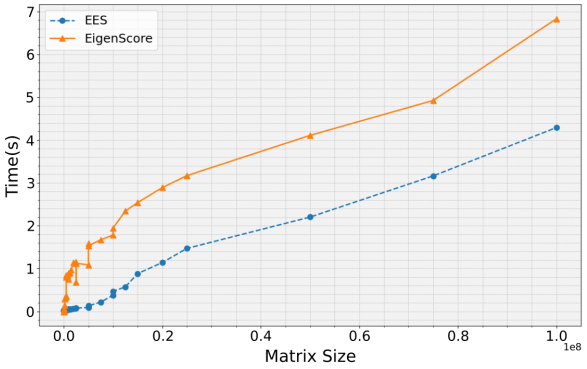


Figure 3: **Efficient EigenScore approximation scaling**. Computation time comparison between EigenScore and EES (moments = 20). The x-axis represents matrix size (rows × columns), and the y-axis shows computation time. As matrix size increases, EES consistently reduces computation time, making it a practical choice.

RQ3: How does EES scale compared to regular EigenScore? The efficiency of EES is compared to that of the regular EigenScore calculation with respect to scaling matrix sizes. These tests are imperative to the application of our training protocol on increasing LLM sizes in Section 4 due to larger matrix sizes to decompose for the EigenScore calculation. We conduct a grid search over two important parameters: Matrix size (Figure 3) and Moments used for EES calculation (Figure 9). The difference between EES time in comparison to EigenScore when increasing the number of columns and rows is visualized in Figure 3 using a moments value of 20. It is evident that EES provides a significant computational advantage when increasing the number of columns or rows. Remarkably, at matrix size $\mathbb{R}^{1e8}$, EES nearly halves the computation time of regular EigenScore calculation at around 4 seconds whereas EigenScore takes approximately 7 seconds to calculate. We can then deduce that given a good enough approximation, EES provides a significant reduction in computational complexity as model and matrix size increase.

4 Sensitivity Dropout (SenD)

Building on the findings from Section 3.1, and aiming to reduce hallucination variance during LLM training, this section introduces SenD, an efficient framework for training LLMs. SenD integrates the EES method discussed in Section 3.2 to enhance computational efficiency while addressing variance in SEI behavior. By identifying SEIs, which contribute to the oscillatory behavior of hallucinations during training, SenD deterministically drops these indices based on a small subset of the training data. This approach ensures an increase in the model's response certainty by the end of training as explained in Algorithm 2.

Algorithm 2 Sensitivity Dropout

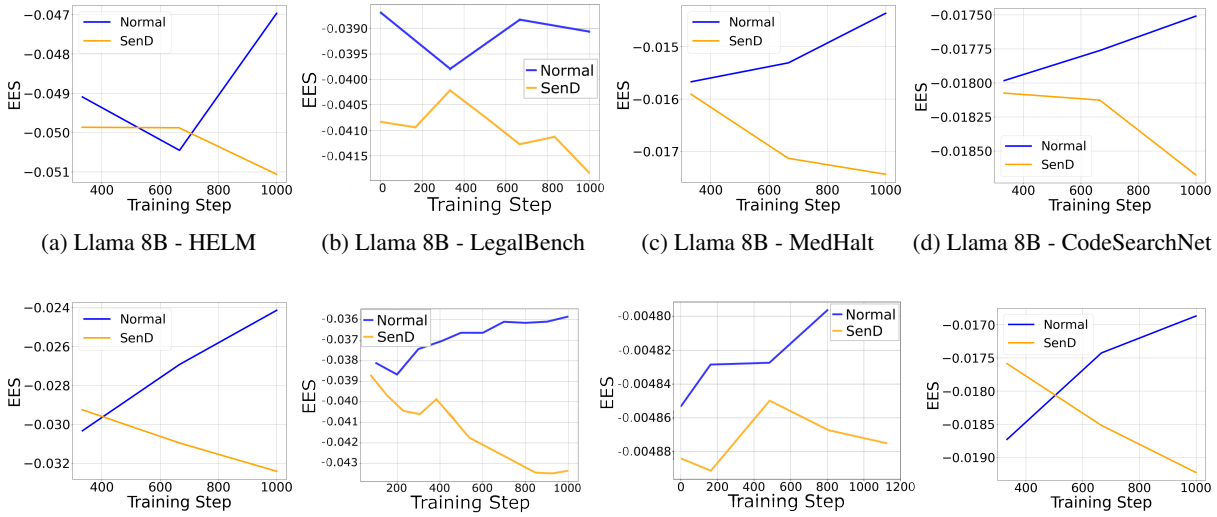
Require: ϵ denotes the acceptable range for loss convergence and δ denotes acceptable range for confabulation (EES) convergence

Initialize dataset with $\alpha\%$ training Y_t and $(100-\alpha)\%$ tracking Y_s
while Loss $> \epsilon$ and EES $> \delta$ **do** ▷ Refer to Algorithm 1 for EES
  for t in T **do** ▷ T denotes the number of checkpoints per SEI calculation
    Train LLM for one checkpoint over Y_t
    Record penultimate layer representations R_t of LLM over Y_s
  end for
  for t in $T-1$ **do**
    Compute variability V_t from R_t to R_{t+1} ▷ Refer to Equation 3
  end for
  Take average Variability $V_{avg} = \frac{1}{N_s}\sum_{i=0}^{N_s} V_i$
  $s = K$ most sensitive embedding indices $\in V_{avg}$ ▷ Refer to Equation 4
  Drop embedding indices s for next T checkpoints
end while

4.1 SenD Complexity Analysis

SenD's additional computational complexity compared to traditional transformer training time complexity per epoch comes from three independent steps, mainly from the attention mechanism and multiple inferences during training:

1. Generating penultimate layer activations: $O(c(mN)^2 dLT)$ This comes from performing c forward passes through the transformer



(a) Llama 8B - HELM (b) Llama 8B - LegalBench (c) Llama 8B - MedHalt (d) Llama 8B - CodeSearchNet

(e) Pythia 1B - HELM (f) Pythia 1B - LegalBench (g) Pythia 1B - MedHalt (h) Pythia 1B - CodeSearchNet

Figure 4: **Regular Training vs. SenD on HELM and LegalBench datasets.** The first row represents Llama 3.1 8B while the second row shows Pythia 1B models. Column one (a) and (e) is trained on the HELM dataset. Column two (b) and (f) is trained on LegalBench. Column three (c) and (g) use the MedHalt dataset. Column four (d) and (h) are trained on CodeSearchNet. In all cases training with SenD demonstrates a more controlled reduction in **EES**, optimizing for hallucination mitigation and loss stability. For results on Llama 3.2 1B training, refer to Appendix D.

model, each with complexity $O((mN)^2dL)$ due to the quadratic attention mechanism, repeated T times per epoch where m, N, d, L are the context size, dataset size, attention head dimension, and number of layers respectively.

2. Computing sensitive embedding indices: $O(IT((mN)c + logI))$ Derived from computing sensitivity for each embedding index I with cost $O((mN)c)$, plus selecting top K% indices with $O(IlogI)$ sorting cost, repeated T times.

3. EES stopping criterion: $O(N^2)$ Classical EigenScore computes a full eigendecomposition with $O(N^3)$ complexity for dense matrices where N is the hidden size of the language model. EES reduces this by using Chebyshev polynomial moments and stochastic trace estimation (Further details available in Appendix B). By replacing eigendecomposition with iterative matrix-vector multiplications (each costing $O(N^2)$) and using a fixed number of moments and trace samples, EES achieves $O(N^2)$ time complexity.

After removing insignificant terms, the total additional computational complexity per epoch is $O((mN)^2dLT)$ ultimately giving $O((mN)^2)$. Please note that this is equal to adding multiple validation steps to an epoch of an LLM's training procedure, implying that no excessive inefficient complexity is introduced by SenD, making it an efficient mitigation technique.

Empirical evaluation of the additional computational complexity of SenD is conducted on the HELM dataset (Su et al., 2024) with 2,000 datapoints using the Llama 8B model (Dubey et al., 2024). We observe that for one epoch, Send training takes 61 minutes while normal training takes 55 minutes. However, in the context of adaptation and reducing the risks of hallucination, we believe this 11% increase is a worthwhile investment.

4.2 SenD Experiments

To evaluate SenD, we use Pythia 1B model (Biderman et al., 2023), Llama 3.2 1B, and Llama 3.1 8B (Dubey et al., 2024) continuing their training on specific datasets rather than restarting pretraining for efficiency. We continually train the models on the following datasets: HELM, consisting of Wikipedia text (Su et al., 2024), MedHALT, a medical dataset emulating real-world entrance exam questions (Pal et al., 2023), LegalBench consisting of data for reasoning in LLMs (Guha et al., 2023), and CodeSearchNet consisting of programming prompts (Husain et al., 2020). Note that HELM and MedHALT are specifically designed for hallucination detection/mitigation in LLMs. SenD implements the EigenScore reduction technique

Metric	MedHalt		HELM		LegalBench		CodeSearchNet	
	SenD	Normal	SenD	Normal	SenD	Normal	SenD	Normal
HellaSwag	0.73	**0.75**	0.73	**0.74**	**0.73**	0.72	**0.69**	0.40
MMLU	0.42	**0.64**	**0.67**	0.65	0.56	**0.59**	**0.26**	0.25
Token Entropy	**0.32**	0.33	**0.79**	0.95	0.49	0.49	**0.21**	0.33

Table 1: Effects of training Llama 3.1 8B model on downstream tasks with and without SenD. In HellaSwag and MMLU, a higher score depicts better performance and lower Token Entropy shows higher model confidence.

from Section 3.1 and detects SEIs using a 3- checkpoint window on a specialized hallucination tracking dataset. The distance between checkpoints and the dropout rate K are tunable hyperparameters. Given our ablation study in Appendix C, we opt for $K = 20\%$ and Threshold = 3 for the experiments. SEIs in the penultimate layer are identified based on their variability across checkpoints and are deterministically dropped for the subsequent 3 training checkpoints. This is repeated at each 3- checkpoint interval until loss convergence, effectively mitigating hallucination tendencies and oscillations. Since we use SenD in a continual manner, we freeze 24 layers for Llama 8B and 12 layers for Llama 1B and Pythia 1B to reduce the effects of forgetting on both SenD and normal training.

RQ4: How does the performance of SenD compare across Pythia and LLaMA models? Pythia and Llama training results are illustrated in Figure 4. To validate that EES accurately approximates the EigenScore metric; we compare the model's progress during training detailed in Appendix B.6. Upon confirming that, we proceed to compare the performance of Pythia 1B, Llama 3.2 1B, and Llama 3.1 8B trained using normal training to SenD. As shown in Figure 4 for Llama 8B and Pythia 1B and detailed in Appendix D for Llama 1B, across all three models and domains, training with SenD reduces EES as well as variance during training. In all cases, the final model trained with SenD achieves a lower EES compared to standard training, demonstrating its effectiveness.

RQ5: What is the effect of SenD on downstream tasks and uncertainty metrics? To assess the effectiveness of SenD on SOTA factuality metrics and downstream tasks, we evaluate several benchmarks. First, the HellaSwag (Zellers et al., 2019) and Massive Multitask Language Understanding (MMLU) (Hendrycks et al., 2021) benchmarks, implemented via LMEval Harness (Gao et al., 2024b),

Training	SenD	Normal
FactScore	**0.44**	0.39
FactScore + RAG	**0.50**	0.40
HaluEval Accuracy	**0.74**	**0.74**
HaluEval Correctness	**0.98**	**0.98**
HaluEval Exact Match	**0.75**	**0.75**

Table 2: Hallucination targeted metrics for Llama 8B. Higher values for all metrics are better. For results in Llama 1B and Pythia 1B refer to Table 4.

verify that downstream performance is maintained (Table 1 for Llama 8B; Table 3 for Llama 1B). In addition, token distribution entropy (Kossen et al., 2024), FactScore (Min et al., 2023), and HaluEval (Li et al., 2023) (Tables 1 and 4) are used to assess model certainty and factuality, where lower entropy indicates higher confidence, FactScore measures factual retention, and HaluEval evaluates hallucination tendencies in question answering. FactScore and HaluEval are run solely on the HELM dataset due to computational power restrictions. HELM was selected for these tests due to its similarity to the testing functions, giving a more accurate depiction of a training and testing scenario.

SenD does not degrade downstream performance and increases the end model's confidence. As shown in Tables 1 and 3, HellaSwag and MMLU scores remain consistent with or without SenD for Llama 8B, confirming stable language understanding. Moreover, the reduced average token distribution entropy observed in Table 1 (Llama 8B) and Table 4 (Llama 1B and Pythia 1B) indicates up to a 17% increase in test-time confidence. Additionally, FactScore improves by 11% with SenD compared to standard training without RAG (Table 2) and by 10% relative to training with RAG during inference, demonstrating better retention of factual knowledge. Finally, the HaluEval metrics experience no change, with both models achieving very high scores on accuracy, correctness, and

exact match in Tables 2 and 4 for Llama 8B and Llama 1B respectively. The consistent performance in metrics associated to hallucinations shows that not only does SenD reduce variance in training, but also provides a more confident model at test time.

RQ6: How does SenD perform in comparison to existing hallucination mitigation approaches? Since SenD is the first method to focus on Hallucinations during the training of LLMs, there are no baselines or SOTA methods to compare it to. However, one could treat SenD as a post-hoc method and compare it to Retrieval Augmented Generation (RAG) (Lewis et al., 2021). As shown in Table 2, when applying RAG to a SenD-trained Llama 8B model, it achieves a higher FactScore than RAG on a normally trained model. Similarly, Pythia 1B and Llama 1B have performance increases on FactScore with SenD compared to their normal counterpart with and without RAG detailed in Appendix Table 4. These results indicate that even though SenD does not outperform post-hoc methods, SenD with RAG enhances the end model's hallucination performance compared to RAG on a normally trained model and should therefore be used conjointly with SOTA methods.

5 Conclusion

In this paper, we presented a protocol to refine the current training methods of LLMs based on experiments showing oscillatory behaviour with respect to hallucinations throughout training (Figure 1). To do this we used the internal states of LLMs, specifically the penultimate layer activations during inference on a specialized dataset. We present an initial method of reducing hallucinations based on the principles of EigenScore metrics introduced by Chen et al. (2024). We showed empirically that our SEI detection method significantly reduces the EigenScore on inference of LLMs throughout various stages of training (Figure 2). Following the success of the SEI method, we moved on to the application of a hallucination reduction method on training of Pythia and Llama models in various domains. We show through training with SenD that we are able to fix the oscillatory behaviour initially seen throughout training and reduce the EES of finetuned models as shown in Figure 4 by modifying the internal mechanics of training with **Sensitivity Dropout**. At test time we achieve a 25% increase in FactScore performance and improvement of other SOTA hallucination detection metrics, verifying that SenD provides a substantial improvement to current training protocols both during and after training in Tables 1, 2, 3, and 4.

6 Limitations

Due to computational limitations, SenD has only been applied to continual training in this paper. However, the SenD training framework is applicable to all stages of training. We encourage future work to implement SenD on larger training sets, such as pretraining, to see how SenD performs in these environments. To further advance our work, we plan to scale SenD to larger datasets and models, as current experiments were limited by compute constraints with larger LLMs. Demonstrating SenD's effectiveness on larger open-source models like Meta's Llama 3.2 405B (Dubey et al., 2024) will provide crucial evidence for organizations developing state-of-the-art LLMs to incorporate SenD into their training protocols, ultimately improving model safety. Given that SenD targets variance reduction during training, we anticipate even greater gains on larger LLMs, where the higher inherent variance may amplify the regularization effect and lead to more significant improvements.

7 Acknowledgements

Funding support for project activities has been partially provided by the Canada CIFAR AI Chair. We also express our gratitude to Compute Canada and Mila clusters for their support in providing facilities for our evaluations. We would also like to thank Prakhar Ganesh for his role in developing this work at its early stages. Finally, we would like to thank all our reviewers for their insightful feedback that helped improve the clarity and rigour of our paper.

References

Jimmy Ba and Brendan Frey. 2013. Adaptive dropout for training deep neural networks. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc.

Pierre Baldi and Peter J Sadowski. 2013. Understanding Dropout. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc.

Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar van der Wal. 2023. Pythia: A Suite for Analyzing Large Language Models Across Training and Scaling. *arXiv preprint*. ArXiv:2304.01373 [cs].

Chao Chen, Kai Liu, Ze Chen, Yi Gu, Yue Wu, Mingyuan Tao, Zhihang Fu, and Jieping Ye. 2024. INSIDE: LLMs' Internal States Retain the Power of Hallucination Detection. *arXiv preprint*. ArXiv:2402.03744 [cs].

Kun Dong, Austin R. Benson, and David Bindel. 2019. Network Density of States. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1152–1161. ArXiv:1905.09758 [cs, math].

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, and 514 others. 2024. The Llama 3 Herd of Models.

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, and 5 others. 2024a. A framework for few-shot language model evaluation. Version Number: v0.4.3.

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, and 5 others. 2024b. A framework for few-shot language model evaluation.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2024c. Retrieval-Augmented Generation for Large Language Models: A Survey. *arXiv preprint*. ArXiv:2312.10997 [cs].

Neel Guha, Julian Nyarko, Daniel E. Ho, Christopher Ré, Adam Chilton, Aditya Narayana, Alex Chohlas-Wood, Austin Peters, Brandon Waldon, Daniel N. Rockmore, Diego Zambrano, Dmitry Talisman, Enam Hoque, Faiz Surani, Frank Fagan, Galit Sarfaty, Gregory M. Dickinson, Haggai Porat, Jason Hegland, and 21 others. 2023. LegalBench: A Collaboratively Built Benchmark for Measuring Legal Reasoning in Large Language Models. *arXiv preprint*. ArXiv:2308.11462.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring Massive Multitask Language Understanding. *arXiv preprint*. ArXiv:2009.03300 [cs].

Giwon Hong, Aryo Pradipta Gema, Rohit Saxena, Xiaotang Du, Ping Nie, Yu Zhao, Laura Perez-Beltrachini, Max Ryabinin, Xuanli He, Clémentine Fourrier, and Pasquale Minervini. 2024. The Hallucinations Leaderboard – An Open Effort to Measure Hallucinations in Large Language Models. *arXiv preprint*. ArXiv:2404.05904 [cs].

Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. 2023a. A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions. *arXiv preprint*. ArXiv:2311.05232 [cs].

Shenyang Huang, Jacob Danovitch, Guillaume Rabusseau, and Reihaneh Rabbany. 2023b. Fast and Attributed Change Detection on Dynamic Graphs with Density of States. *arXiv preprint*. ArXiv:2305.08750 [cs].

Hamel Husain, Ho-Hsiang Wu, Tiferet Gazit, Miltiadis Allamanis, and Marc Brockschmidt. 2020. CodeSearchNet Challenge: Evaluating the State of Semantic Code Search. *arXiv preprint*. ArXiv:1909.09436 [cs].

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

Jannik Kossen, Jiatong Han, Muhammed Razzak, Lisa Schut, Shreshth Malik, and Yarin Gal. 2024. Semantic Entropy Probes: Robust and Cheap Hallucination Detection in LLMs. *arXiv preprint*. ArXiv:2406.15927 [cs] version: 1.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2021. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *arXiv preprint*. ArXiv:2005.11401.

Junyi Li, Jie Chen, Ruiyang Ren, Xiaoxue Cheng, Wayne Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. 2024. The Dawn After the Dark: An Empirical Study on Factuality Hallucination in Large Language Models. *arXiv preprint*. ArXiv:2401.03205 [cs].

Junyi Li, Xiaoxue Cheng, Wayne Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. 2023. HaluEval: A Large-Scale Hallucination Evaluation Benchmark for Large Language Models. *arXiv preprint*. ArXiv:2305.11747 [cs].

Lin Lin, Yousef Saad, and Chao Yang. 2014. Approximating spectral densities of large matrices. *arXiv preprint*. ArXiv:1308.5467 [math].

Potsawee Manakul, Adian Liusie, and Mark J. F. Gales. 2023. SelfCheckGPT: Zero-Resource Black-Box Hallucination Detection for Generative Large Language Models. *arXiv preprint*. ArXiv:2303.08896 [cs].

Sewon Min, Kalpesh Krishna, Xinxi Lyu, Mike Lewis, Wen-tau Yih, Pang Wei Koh, Mohit Iyyer, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2023. FActScore: Fine-grained Atomic Evaluation of Factual Precision in Long Form Text Generation. *arXiv preprint*. ArXiv:2305.14251 [cs].

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't Give Me the Details, Just the Summary! Topic-Aware Convolutional Neural Networks for Extreme Summarization. *arXiv preprint*. ArXiv:1808.08745 [cs].

Ankit Pal, Logesh Kumar Umapathi, and Malaikannan Sankarasubbu. 2023. Med-HALT: Medical Domain Hallucination Test for Large Language Models. *arXiv preprint*. ArXiv:2307.15343 [cs, stat].

Vipula Rawte, Amit Sheth, and Amitava Das. 2023. A Survey of Hallucination in Large Foundation Models. *arXiv preprint*. ArXiv:2309.05922 [cs].

Bikash Santra, Angshuman Paul, and Dipti Prasad Mukherjee. 2020. Deterministic dropout for deep neural networks using composite random forest. *Pattern Recognition Letters*, 131:205–212.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.

Weihang Su, Changyue Wang, Qingyao Ai, Yiran HU, Zhijing Wu, Yujia Zhou, and Yiqun Liu. 2024. Unsupervised Real-Time Hallucination Detection based on the Internal States of Large Language Models. *arXiv preprint*. ArXiv:2403.06448 [cs].

Ziwei Xu, Sanjay Jain, and Mohan Kankanhalli. 2024. Hallucination is Inevitable: An Innate Limitation of Large Language Models. *arXiv preprint*. ArXiv:2401.11817 [cs].

Hongbin Ye, Tong Liu, Aijia Zhang, Wei Hua, and Weiqiang Jia. 2023. Cognitive Mirage: A Review of Hallucinations in Large Language Models. *arXiv preprint*. ArXiv:2309.06794 [cs].

Tianyu Yu, Yuan Yao, Haoye Zhang, Taiwen He, Yifeng Han, Ganqu Cui, Jinyi Hu, Zhiyuan Liu, Hai-Tao Zheng, Maosong Sun, and Tat-Seng Chua. 2024. RLHF-V: Towards Trustworthy MLLMs via Behavior Alignment from Fine-grained Correctional Human Feedback. *arXiv preprint*. ArXiv:2312.00849 [cs].

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a Machine Really Finish Your Sentence? *arXiv preprint*. ArXiv:1905.07830 [cs].

A Additional Experiments

A.1 Drastic Embedding Changes leading to Sensitive Embedding Indices

Looking at internal states of the model allows getting a deeper understanding of the dynamics that could be leading to the oscillatory behaviour seen in Figure 1. To do this, we record the net change (Definition 3.2) between checkpoints of the penultimate layer where one checkpoint would be the correct answer and the next would hallucinate. This net change with respect to various different input texts is plotted in Figure 5. It can be observed that there were specific embedding activations that experienced drastically more change relative to the rest of the embeddings. This is the main source of motivation to further define SEIs (Definition 3.3).

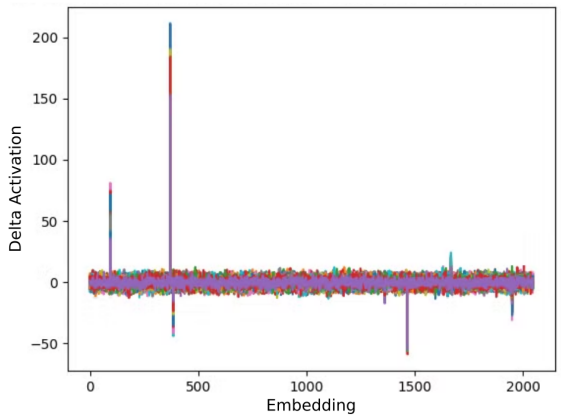


Figure 5: **Net change of sentence embeddings** between checkpoints 125,000 and 143,000. Each different colour is a different input text. As depicted, there are specific embedding indices that go through drastic changes between the two checkpoints of the training regardless of the input.

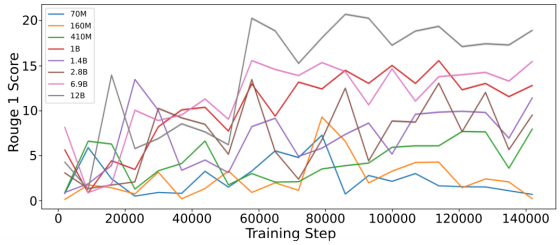


Figure 6: XSum Rouge 1 Score metric results on Pythia suite.

A.2 Hallucination Oscillations Across Model Sizes

Figures 6, 7, and 8 show our study of hallucination oscillations during the training of Pythia models.

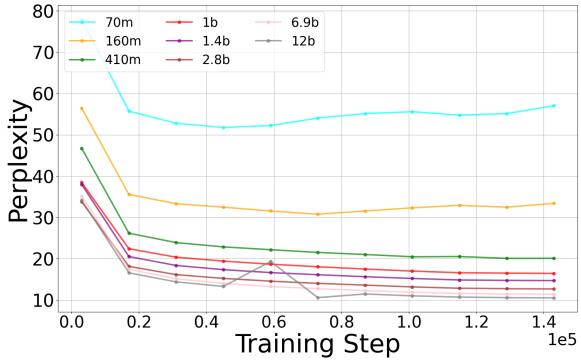


Figure 7: Perplexity (PPL) metric results on Pythia suite.

An overall observation across the plots is that as opposed to our intuitive expectation which is a linear decrease of the hallucination detection metric when the model scales linearly, neither the oscillations during the training of the model decrease, nor the end model reaches its optimal state in terms of the hallucination metric.

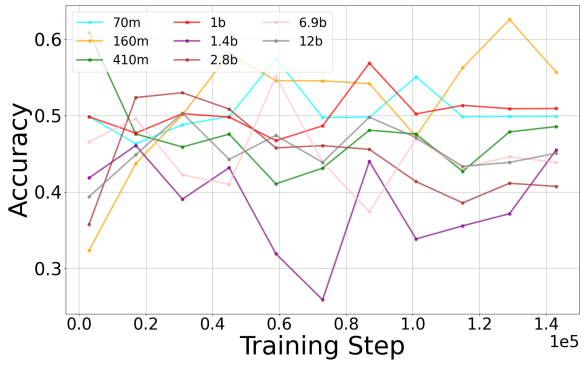
B Efficient EigenScore (EES) Derivation

B.1 Background: Chebyshev polynomials

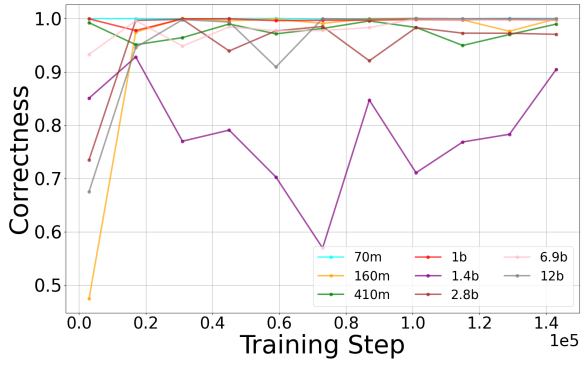
Chebyshev polynomials are a sequence of orthogonal polynomials in the interval $[-1, 1]$ – orthogonality property shown in equation 8 – that are widely used in numerical analysis, approximation theory, and other areas of applied mathematics. In this work, we are mainly concerned with the Chebyshev polynomials of the first kind with the recurrence relation shown in equation 9. Note that this recurrence could also be applied to matrices. Any function f defined in the interval $[-1, 1]$ can be approximated with the Chebyshev expansion as shown in 10.

$$\int_{-1}^{1} \frac{2}{(1+\delta_{0n})\pi\sqrt{1-x^2}} T_m(x)T_n(x)\,dx = \delta_{mn}, \quad \text{where} \quad \delta_{mn} = \begin{cases} 1 & \text{if } m = n, \\ 0 & \text{if } m \neq n, \end{cases} \tag{8}$$

$$\begin{aligned} T_0(x) &= 1, \\ T_1(x) &= x, \\ T_{n+1}(x) &= 2x \cdot T_n(x) - T_{n-1}(x), \quad \text{for } n \geq 1. \end{aligned} \tag{9}$$



(a) HaluEval Accuracy metric results.



(b) HaluEval Correctness metric results.

Figure 8: Ablation studies on various HaluEval metrics for hallucination detection on Pythia suite.

$$f(x) = \sum_{n=0}^{\infty} c_n T_n(x), \tag{10}$$

$$\text{where } c_n = \frac{2}{\pi} \int_{-1}^{1} \frac{f(x) T_n(x)}{\sqrt{1-x^2}}\, dx \text{ for } n > 0, \tag{11}$$

$$c_0 = \frac{1}{\pi} \int_{-1}^{1} \frac{f(x)}{\sqrt{1-x^2}}\, dx. \tag{12}$$

B.2 Background: DOS and KPM

Let H be a symmetric matrix $H \in \mathbb{R}^{N\times N}$ with an eigendecomposition $H = Q\Lambda Q^T$, where $\Lambda = \text{diag}(\lambda_1, \cdots, \lambda_N)$ and $Q = [q_1, \cdots, q_N]$ is orthogonal. The spectral density induced by H is the generalized function:

$$\mu(\lambda) = \frac{1}{N} \sum_{i=1}^{N} \delta(\lambda - \lambda_i), \tag{13}$$

where δ is the Dirac delta function. For any analytic test function f, the integral of f with respect to μ is:

$$\int f(\lambda)\mu(\lambda)\, d\lambda = \text{trace}(f(H)). \tag{14}$$

Dong et al. (2019) introduced KPM as a numerical technique to approximate DOS. KPM approximates DOS by expanding it in terms of chebyshev polynomials. Requiring the matrix's spectrum to be supported in the interval $[-1, 1]$, KPM approximates DOS with the following formula, λ being the eigen value of the matrix H and d_m approximated by Stochastic Trace Estimation:

$$\mu^{\approx}(\lambda) = \sum_{m=1}^{\infty} d_m T_m^*(\lambda), \tag{15}$$

$$\text{where} \quad d_m = \frac{1}{N} \text{trace}(T_m(H)), \tag{16}$$

$$\text{and} \quad d_m \approx \frac{1}{N} \frac{1}{N_z} \sum_{j=1}^{N_z} \mathbf{z}_j^T T_m(H) \mathbf{z}_j, \tag{17}$$

$$\text{and} \quad T_m^*(x) = \frac{2}{(1+\delta_{0m})\pi\sqrt{1-x^2}} T_m(x). \tag{18}$$

In the application for hallucination detection, we can use equation 14 to derive a formula for the EigenScore approximation using the properties of Chebyshev polynomials and DOS.

B.3 Stochastic Trace Estimation on Embedding Matrix

We are interested in computing the d_m term of DOS relying solely on the embedding matrix E therefore we need to rewrite d_m as follows:

$$d_m = \frac{1}{K} \frac{1}{N_z} \sum_{j=0}^{\infty} z_j^T T_m(E^T E) z_j \tag{19}$$

where T_m can be computed using the Chebyshev polynomials of matrix $C = E^T E$.

$$\begin{aligned} T_0(E^T E)\mathbf{z}_j &= I\mathbf{z}_j = \mathbf{z}_j, \\ T_1(E^T E)\mathbf{z}_j &= E^T E \mathbf{z}_j, \\ T_{m+1}(E^T E)\mathbf{z}_j &= 2E^T E T_m(E^T E)\mathbf{z}_j - \ldots \\ &\quad \ldots T_{m-1}(E^T E)\mathbf{z}_j \end{aligned}$$

Each term can be computed with a matrix-vector multiplication.

B.4 EES Integral Calculation

Given the orthogonality of the Chebyshev polynomials, we can simplify the integral mentioned in proposition 1. To approximate the EigenScore, we

will expand $\log(\lambda)$ in terms of Chebyshev polynomials and use their orthogonality to simplify the integral.

Expanding and Integrating

To approximate the integral:

$$\frac{1}{K}\int \log(\lambda)\mu(\lambda)\,d\lambda \tag{20}$$

Substitute the Chebyshev Expansion for DOS:

$$\mu(\lambda) \approx \sum_{m=0}^{M} d_m T_m^*(\lambda) \tag{21}$$

where:

$$T_m^*(\lambda) = w(\lambda)T_m(\lambda) = \frac{2}{\pi\sqrt{1-\lambda^2}(1+\delta_{0m})}T_m(\lambda)$$

Distribute $\log(\lambda)$ in the integral:

$$\frac{1}{K}\int \log(\lambda)\left(\sum_{m=0}^{M} d_m T_m^*(\lambda)\right)d\lambda \tag{22}$$

$$= \frac{1}{K}\sum_{m=0}^{M} d_m \int \log(\lambda)T_m^*(\lambda)\,d\lambda \tag{23}$$

Evaluate the Integral Using Orthogonality:

To simplify the integral,

$$\int \log(\lambda)T_m^*(\lambda)\,d\lambda \tag{24}$$

First, express $\log(\lambda)$ as a series of Chebyshev polynomials:

$$\log(\lambda) = \sum_{m=0}^{\infty} c_m T_m(\lambda) \tag{25}$$

Then:

$$\begin{aligned}&\int_0^1 \log(\lambda)T_m^*(\lambda)\,d\lambda \\ &= \int_0^1 \left(\sum_{m=0}^{\infty} c_m T_m(\lambda)\right)T_m(\lambda)\,d\lambda\end{aligned} \tag{26}$$

Note: The lower bound of the integral is 0 as the matrix is defined in the spectrum $[0, 1]$.

Using the orthogonality, we get:

$$c_m = \int_0^1 \log(\lambda)T_m^*(\lambda)\,d\lambda \tag{27}$$

So the integral simplifies to:

$$\frac{1}{K}\sum_{m=0}^{M} d_m c_m \tag{28}$$

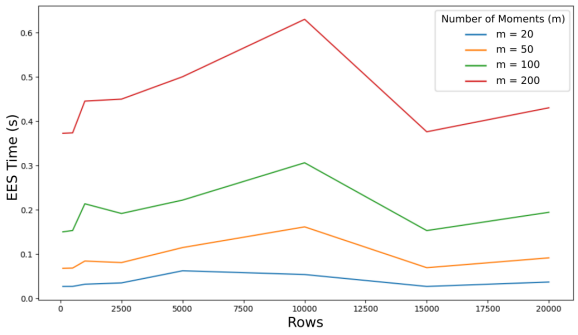


Figure 9: **Effect of changing number of moments on EES** calculation time (seconds). More moments gives more accurate approximation but higher computation time.

B.5 Efficient EigenScore Moments

Figure 9 presents the effect of using different moment values as the number of matrix rows increases with respect to time. This is an important hyperparameter to tune as increasing the number of moments on EES correlates to having a more accurate and representative approximation of the EigenScore. We observe that as moments in EES increase, the time to calculate EES increases. From this result, we conclude that selecting a moment value of under 50 would provide a balanced trade-off between accuracy and calculation time.

B.6 EigenScore and EES training trajectories

To demonstrate the high correlation between EigenScore and EES, we record the progress of Pythia 1B training on the HELM dataset using both EigenScore and EES hallucination metrics (Figure 10). Albeit a different scale and window, the trajectories, magnitude and shape of the graphs are nearly identical while EES takes only 4 minutes to calculate and EigenScore takes approximately 8, an astounding 2x increase in compute speed. These results show that our metric closely resembles the target metric while greatly reducing the required computational resources.

C Ablation study on K and Step Thresholding for SenD

Figure 12 shows the ablation study done on K and Figure 11 illustrates the ablations study done on the Step Threshold for SenD experiments. As depicted, $K = 20\%$ and Threshold = 3 are chosen for our experiments except for Llama 3.1 8B model which due to its larger size requires more embedding indices to be dropped, hence adapting to $K = 30\%$.

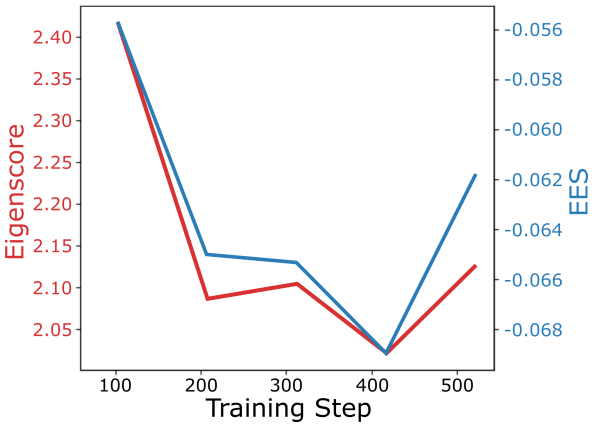


Figure 10: Performance of SenD on Pythia 1B wih HELM dataset computed with both EES and regular EigenScore. EES is able to closely track the true EigenScore performance metric, showing that it is a good approximator.

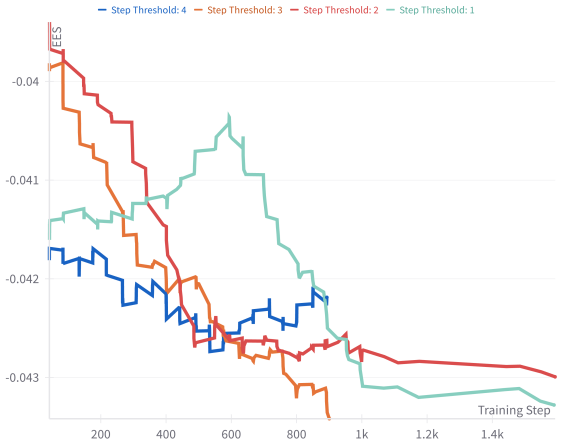


Figure 11: Ablation on Step Threshold $\in \{1, 2, 3, 4\}$ on the Pythia 1B model with the LegalBench dataset. The fastest drop in EES is achieved by setting Threshold = 3, therefore we choose Threshold = 3 for our experiments. Results are averaged over 5 multiple runs.

D Additional Pythia 1B, Llama 3.2 1B, and Llama 3.1 8B Training with and without SenD

Here, we present additional experimental results of training Pythia and Llama on multiple domains. Figure 13 supplements the results discussed in Section 4 by illustrating the training procedures on an additional model, Llama 1B

In the Pythia 1B setting, the EES achieved with training using SenD remains consistently lower than that of normal training and exhibits fewer oscillations throughout the training process. In the Llama 3.1 8B setting, while both approaches show an increase in the EES metric during training, the final model trained with SenD achieves a lower EES, indicating a reduced likelihood of hallucinations in this domain.

E SenD performance across different models, datasets, and metrics

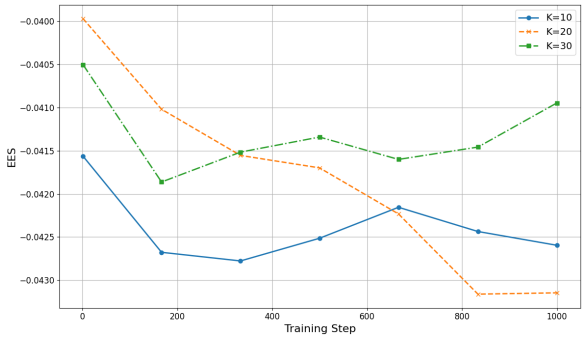
Here we present a more in depth look at SenD performance, looking at its effect beyond just Llama 3.1 8B. Here we present in Table 3 the results from running downstream tasks HellaSwag and MMLU are presented for Llama models with sizes 8B and 1B. we can see that although the normally trained models are performing better, the scale of performance increase is negligible, in most cases being within 2% of SenD's performance. Given this negligible difference, we are confident that the SenD tuning is not drastically affecting the model's performance on downstream tasks. We also present in Table 4 the results of hallucination based metrics for Llama 8B, Llama 1B, and Pythia 1B. We see that although the HaluEval metrics do not change, they are very good for both models. On the other hand, FactScore is significantly increased when using SenD both with and without RAG. This demonstrates SenD's ability to produce factual information more consistently and the additive power of using both SenD and RAG together.

Model	Task	Training	HellaSwag	MMLU
Llama 8B	MedHalt	SenD	0.73	0.42
		Normal	**0.75**	**0.64**
	HELM	SenD	0.73	**0.67**
		Normal	**0.74**	0.65
	LegalBench	SenD	**0.73**	0.56
		Normal	0.72	**0.59**
	CodeSearchNet	SenD	**0.69**	**0.26**
		Normal	0.40	0.25
Llama 1B	MedHalt	SenD	0.59	0.40
		Normal	0.59	**0.43**
	HELM	SenD	0.59	0.43
		Normal	0.59	**0.44**
	LegalBench	SenD	0.57	0.34
		Normal	0.57	**0.35**
	CodeSearchNet	SenD	0.58	0.42
		Normal	**0.59**	0.42

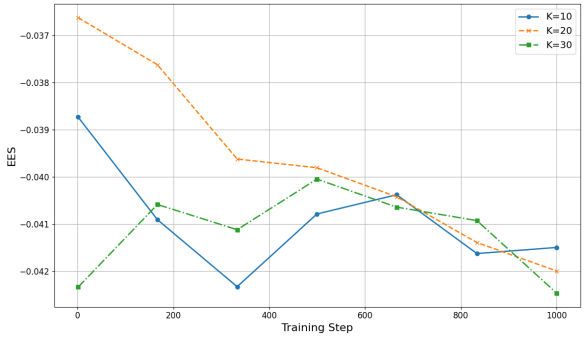
Table 3: **Final Model Downstream Performance**: SenD vs. Normal Training on Llama 8B and 1B on downstream tasks HellaSwag and MMLU. A higher score is better for both of these metrics.

Model	Llama 8B		Llama 1B		Pythia 1B	
Training	SenD	Normal	SenD	Normal	SenD	Normal
FactScore	**0.44**	0.39	**0.35**	0.30	**0.07**	0.05
FactScore + RAG	**0.50**	0.40	0.40	0.40	**0.28**	0.25
HaluEval Accuracy	0.74	0.74	0.49	0.49	**0.016**	0.014
HaluEval Correctness	0.98	0.98	0.99	0.99	0.027	0.027
HaluEval Exact Match	0.75	0.75	0.49	0.49	**0.589**	0.496
Entropy of Tokens	**0.79**	0.95	1.01	1.01	**1.44**	1.49

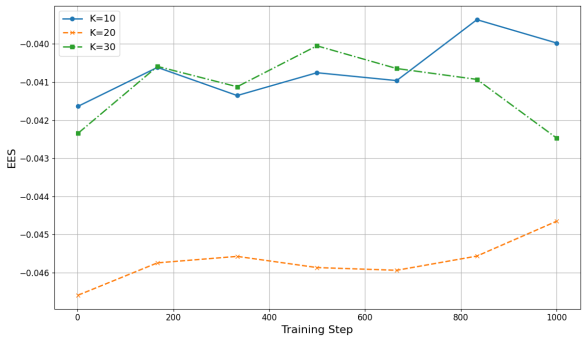
Table 4: **Final Model Hallucination Performance: SenD vs. Normal Training (Pythia 1B, Llama 8b, and Llama 1B)**. HaluEval refers to a QA task. Differing factors between the two FactScore tasks (100 and 1k) refers to the number of testing points.



(a) Ablation study on K using the LegalBench dataset.

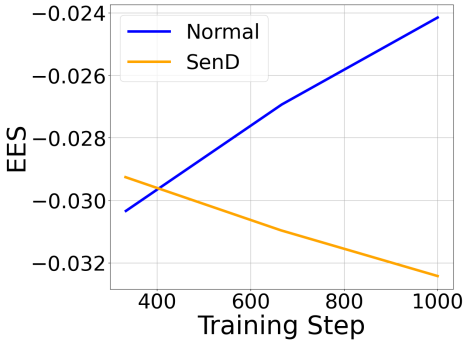


(b) Ablation study on K using the MedHalt dataset.

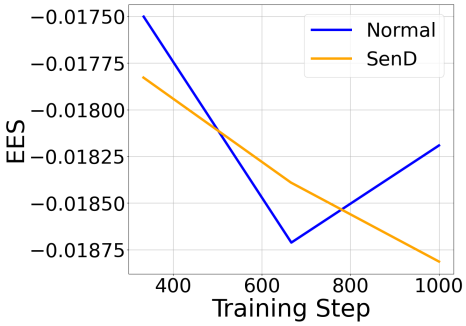


(c) Ablation study on K using the HELM dataset.

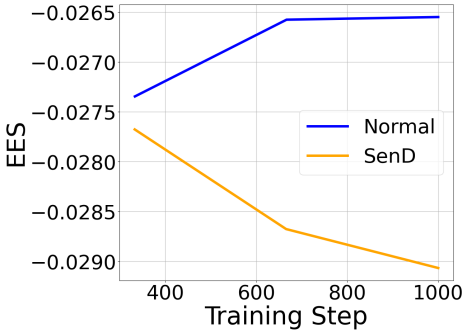
Figure 12: Ablation on dropout rate $K \in \{10\%$ orange, 20% blue, 30% green$\}$ using the Pythia 1B model averaged over 10 runs on the LegalBench dataset. $K = 20\%$ achieves optimal performance in reducing EES throughout training for HELM and LegalBench and almost equalizes $K = 30\%$ in stabilizing the halluciantion oscillations, therefore we choose $K = 20\%$ for our experiments.



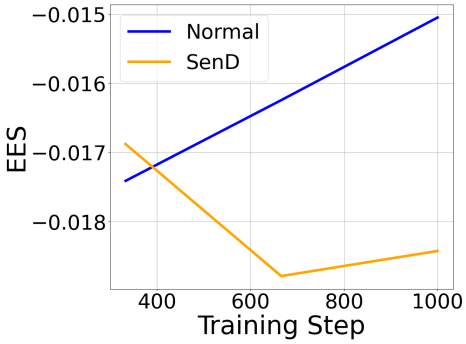
(a) Llama 1B - HELM



(b) Llama 1B - LegalBench



(c) Llama 1B - MedHalt



(d) Llama 1B - CodeSearchNet

Figure 13: Evaluation results for Llama 1B across different benchmarks. SenD consistently outperforms normal training by reducing EES in a more controlled manner.