Noise Augmented Fine Tuning for Mitigating Hallucinations in Large Language Models

Afshin Khadangi[®]¹, Amir Sartipi[®]¹, Igor Tchappi[®]¹, Ramin Bahmani[®]¹

¹SnT, University of Luxembourg ¹{afshin.khadanki, amir.sartipi, igor.tchappi, ramin.bahmani}@uni.lu

April 7, 2025

Abstract

Large language models (LLMs) often produce inaccurate or misleading content—hallucinations. To address this challenge, we introduce Noise-Augmented Fine-Tuning (NoiseFiT), a novel framework that leverages adaptive noise injection based on the signal-to-noise ratio (SNR) to enhance model robustness. In particular, NoiseFiT selectively perturbs layers identified as either high-SNR (more robust) or low-SNR (potentially under-regularized) using a dynamically scaled Gaussian noise. We further propose a hybrid loss that combines standard cross-entropy, soft cross-entropy, and consistency regularization to ensure stable and accurate outputs under noisy training conditions. Our theoretical analysis shows that adaptive noise injection is both unbiased and variance-preserving, providing strong guarantees for convergence in expectation. Empirical results on multiple test and benchmark datasets demonstrate that NoiseFiT significantly reduces hallucination rates, often improving or matching baseline performance in key tasks. These findings highlight the promise of noise-driven strategies for achieving robust, trustworthy language modeling without incurring prohibitive computational overhead. Given the comprehensive and detailed nature of our experiments, we have publicly released the fine-tuning logs, benchmark evaluation artifacts, and source code online at W&B, Hugging Face, and GitHub, respectively, to foster further research, accessibility and reproducibility.

1 Introduction

Large language models (LLMs) such as GPT-3 [4] and GPT-4 [18], built upon transformer architectures [24], have revolutionized the field of natural language processing by achieving state-of-the-art performance on a diverse range of tasks. Despite their impressive capabilities, these models are known to generate content that is often inaccurate or misleading—phenomena broadly referred to as hallucinations [9, 1, 17]. In high-stakes domains such as healthcare [15] and finance [29], the reliability of LLM outputs is critical, making the mitigation of hallucinations not only a technical challenge but also a practical imperative.

Existing strategies to reduce hallucinations include reinforcement learning from human feedback (RLHF) [19], self-consistency training [27], contrastive decoding [12], and adversarial training [36]. While these methods have achieved moderate success, they often come with high computational costs, increased sensitivity to hyperparameter tuning, and limited generalization across varying domains [34, 33]. These limitations underscore the need for alternative approaches that can robustly improve model performance without incurring prohibitive resource demands.

Recent research has increasingly focused on noise injection as a means to enhance model robustness. Early work in image restoration demonstrated the efficacy of learning from noisy data [10], and this idea has since been adapted for natural language processing. In the context of LLM fine-tuning, noise injection techniques have shown promising results. For instance, methods such as noise perturbation fine-tuning for robust quantization [25] and the use of noisy embeddings to improve instruction fine-tuning [8] illustrate that controlled noise can help models generalize better under diverse and challenging conditions. Additional studies have revealed hidden capabilities of noise injection in reducing overconfidence and mitigating model biases [23, 32], and even enhancing hallucination detection [13]. Complementary evaluation frameworks, including large-scale benchmarks for hallucination evaluation [11] and rigorous

instruction-following assessments [35], further motivate the development of noise-based approaches in addressing LLM shortcomings.

In light of these advancements, we propose *Noise-Augmented Fine-Tuning (NoiseFiT)*, a novel framework that integrates adaptive noise injection into the fine-tuning process of LLMs. The key innovation of NoiseFiT lies in its use of adaptive Gaussian noise injection, which is guided by the signal-to-noise ratio (SNR) of internal representations. By selectively perturbing transformer layers where the model's representations are most stable, our approach aims to directly enhance robustness against hallucinations while minimizing disruption to overall performance. This strategy is further supported by recent studies demonstrating the benefits of noise regularization and stochastic perturbations in fine-tuning settings [26, 30, 7].

The central research questions motivating this work are:

- RQ1: How does adaptive noise injection during fine-tuning help mitigate hallucinations in the outputs of LLMs?
- RQ2: What is the relationship between the intensity of noise injection and the trade-off between robustness and task performance across diverse applications?
- RQ3: How does the proposed NoiseFiT framework affect the computational efficiency and scalability of hallucination mitigation?

To address these questions, our methodology introduces a hybrid loss objective that combines standard cross-entropy with soft target regularization and consistency-based penalties computed over multiple noisy forward passes. This formulation not only ensures that the model retains its predictive capabilities under perturbation but also enforces consistency in its outputs—a key requirement for robust performance in safety-critical applications. Furthermore, our mathematical analysis establishes theoretical properties of the adaptive noise injection mechanism, including its unbiasedness and variance-preserving characteristics.

Empirical evaluations conducted on multiple benchmarks, such as the GPQA [20], MUSR [21], IFE-val [35], BBH [22], MATH [6], and MMLU-Pro [28] datasets, validate the effectiveness of NoiseFiT. Our experiments demonstrate a significant reduction in hallucinations compared to conventional fine-tuning methods, while maintaining or even improving performance on standard language understanding tasks. These results are consistent with prior findings on the benefits of noise injection in both vision [10] and language domains [8, 23, 32].

In summary, this paper makes the following contributions:

- 1. We introduce the NoiseFiT framework that leverages adaptive noise injection based on internal SNR, addressing the critical issue of hallucinations in LLMs.
- 2. We propose a novel hybrid loss function that integrates multiple regularization techniques to ensure robust and consistent model behavior.
- 3. We provide both theoretical analysis and empirical evidence demonstrating that our approach reduces hallucinations while preserving competitive performance across various tasks.

The remainder of this paper is organized as follows. Section 2 details the NoiseFiT methodology, including the adaptive noise injection strategy. In Section 3, we present the mathematical foundations of NoiseFiT to underpin the theoretical aspects of our approach. Section 4 reports the experimental setup and results, benchmarking NoiseFiT against state-of-the-art methods. Finally, Section 5 discusses the broader implications of our findings and outlines future research directions.

Through this work, we aim to contribute a viable and efficient solution to the challenge of LLM hallucinations, with potential applications in critical domains where reliability is paramount.

2 Methods

Our approach, Noise-Augmented Fine-Tuning, is designed to reduce hallucinations in LLMs by integrating adaptive noise injection directly into the fine-tuning process and leveraging a hybrid loss function. Additionally, NoiseFiT selectively injects noise into specific layers based on a SNR criterion, thereby adapting perturbations to target layers with both stable (robust) and unstable (loose) activations, depending on their contribution to model behavior.

2.1 Dataset

The model was fine-tuned and evaluated using two distinct datasets, each comprising prompt-response pairs. The fine-tuning dataset includes 832 samples in its training split, where each sample includes a prompt and its corresponding response. These pairs cover a wide range of topics, including literature, history, geography, and science. A sample of the fine-tuning set is as follows:

- **Prompt:** "Who discovered electron?"
- Response: "J.J. Thomson discovered electron."

The test dataset consists of 208 samples, structured similarly to the training set. This data set also encompasses diverse topics, ensuring a robust evaluation of the performance of the model. A sample of the testing set is:

- **Prompt:** "Who painted the Guernica?"
- Response: "Pablo Picasso painted the Guernica."

These datasets were used to fine-tune the models and assess their ability to generate accurate responses to various questions.

2.2 Layer Selection via Signal-to-Noise Ratio

To determine which layers are best suited for noise injection, we perform the following steps:

(a) Clean and Noisy Forward Passes: A clean forward pass through the model produces hidden states for each layer:

$$\mathbf{h}_{\ell}^{\text{clean}} \in \mathbb{R}^{B \times L_{\ell} \times H}$$
,

where B is the batch size, L_{ℓ} is the sequence length at layer ℓ , and H is the hidden dimensionality. In parallel, multiple noisy forward passes (using adaptive noise injection) yield noisy hidden states:

$$\mathbf{h}_{\ell}^{\text{noisy}} \in \mathbb{R}^{B \times L_{\ell} \times H}$$
.

- (b) **SNR Computation:** In our framework, the SNR is used to identify transformer layers with robust or loose activations for targeted noise injection. We begin by defining the following notation:
 - B is the batch size, i.e., the number of examples in a mini-batch.
 - L_{ℓ} is the sequence length (number of tokens) at layer ℓ .
 - \bullet H is the hidden dimensionality, i.e., the size of each hidden state vector.

For each transformer layer ℓ :

- (a) Clean and Noisy Activations:
 - The clean hidden states are denoted by

$$\mathbf{h}_{\ell}^{\text{clean}} \in \mathbb{R}^{B \times L_{\ell} \times H}$$
.

• The noisy hidden states (obtained after adaptive noise injection) are denoted by

$$\mathbf{h}_{\ell}^{\text{noisy}} \in \mathbb{R}^{B \times L_{\ell} \times H}.$$

(b) **Signal Metric:** The signal S_{ℓ} is computed as the mean absolute activation of the clean hidden states:

$$S_{\ell} = \frac{1}{B \cdot L_{\ell} \cdot H} \sum_{b=1}^{B} \sum_{t=1}^{L_{\ell}} \sum_{i=1}^{H} \left| \left[\mathbf{h}_{\ell}^{\text{clean}} \right]_{b,t,i} \right|. \tag{1}$$

(c) Noise Metric: The noise N_{ℓ} is estimated by computing the average absolute difference between the noisy and clean activations:

$$N_{\ell} = \frac{1}{B \cdot L_{\ell} \cdot H} \sum_{b=1}^{B} \sum_{t=1}^{L_{\ell}} \sum_{i=1}^{H} \left| \left[\mathbf{h}_{\ell}^{\text{noisy}} - \mathbf{h}_{\ell}^{\text{clean}} \right]_{b,t,i} \right|.$$
 (2)

(d) **SNR Computation:** The signal-to-noise ratio for layer ℓ is then defined as:

$$SNR_{\ell} = \frac{S_{\ell}}{N_{\ell} + \epsilon},\tag{3}$$

where $\epsilon > 0$ is a small constant (e.g., 10^{-6}) to avoid division by zero.

A higher S_{ℓ} indicates stronger (more robust) activations in the clean forward pass, while a lower N_{ℓ} implies that the adaptive noise injection minimally disturbs the hidden states. Therefore, a higher SNR_{ℓ} means that the layer exhibits stable activations even in the presence of noise. Based on these values, a fixed number k of layers with the highest or lowest SNR_{ℓ} are selected for noise injection during fine-tuning.

(c) **Layer Selection:** Given the SNR values $\{SNR_{\ell}\}_{\ell=1}^{L}$ (with L being the total number of layers), we select a predetermined number k of layers with the highest or lowest SNR.

2.3 Adaptive Noise Injection

Adaptive noise injection perturbs hidden states by injecting zero-mean Gaussian noise, scaled by hidden states statistics and model uncertainty. The process is broken down into following steps:

2.3.1 Basic Noise Injection

For a given hidden state vector $\mathbf{h} \in \mathbb{R}^H$, the basic noise injection is defined as:

$$\tilde{\mathbf{h}} = \mathbf{h} + \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_H),$$
 (4)

where \mathbf{I}_H denotes the $H \times H$ identity matrix.

2.3.2 Adaptive Scaling via Robust Statistics and Uncertainty

The adaptive noise scale is computed using several components:

- 1. Robust Statistics:
 - Median:

$$\mu_{\text{med}} = \text{median}(\mathbf{h}).$$

• Median Absolute Deviation (MAD):

$$MAD(\mathbf{h}) = median(|\mathbf{h} - \mu_{med}|).$$

2. Exponential Weighting: Define a weighting function that emphasizes deviations from the median:

$$w(\mathbf{h}) = \exp\left(-\beta \frac{|\mathbf{h} - \mu_{\text{med}}|}{\text{MAD}(\mathbf{h}) + \epsilon}\right),\tag{5}$$

where $\beta > 0$ is a hyperparameter controlling the sensitivity to deviations, and ϵ is a small constant to ensure numerical stability.

- 3. Uncertainty-Based Noise Factor: Two strategies are used to compute the noise factor (η) :
 - Using Logits: If logits $\mathbf{z} \in \mathbb{R}^{B \times L \times V}$ (with vocabulary size V) are available, compute the softmax probabilities:

$$p_{t,k} = \frac{\exp(z_{t,k})}{\sum_{i=1}^{V} \exp(z_{t,i})}.$$

Then, for each token position t, the entropy is:

$$H_t = -\sum_{k=1}^{V} p_{t,k} \log(p_{t,k} + \epsilon). \tag{6}$$

The average entropy over the sequence is:

$$\bar{H} = \frac{1}{L} \sum_{t=1}^{L} H_t,$$
 (7)

and the noise factor is defined as:

$$\eta = \exp(\bar{H}). \tag{8}$$

• Variance of Hidden States: In the absence of logits, a pseudo-entropy is computed from the variance of the hidden states:

$$\eta = \exp\left(\frac{\operatorname{Var}(\mathbf{h})}{\mathbb{E}[\operatorname{Var}(\mathbf{h})] + \epsilon}\right).$$
(9)

4. Effective Noise Scale: Integrate the above components along with a base noise standard deviation σ_{base} , a learnable scaling parameter α , and an externally provided gating factor noise_gate $\in [0, 1]$ to obtain:

$$\sigma_{\text{eff}} = \sigma_{\text{base}} \cdot \alpha \cdot \text{noise_gate} \cdot \text{MAD}(\mathbf{h}) \cdot w(\mathbf{h}) \cdot \eta. \tag{10}$$

5. Final Adaptive Noise Injection: The final perturbed hidden state is computed as:

$$\tilde{\mathbf{h}} = \mathbf{h} + \sigma_{\text{eff}} \cdot \boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_H).$$
 (11)

2.4 Hybrid Loss Objective with Consistency Regularization

To ensure robust representation learning under noise, the training objective is augmented with multiple loss components:

2.4.1 Cross-Entropy Loss (\mathcal{L}_{ce})

A standard cross-entropy loss is computed on the clean forward pass:

$$\mathcal{L}_{ce} = -\frac{1}{N} \sum_{t=1}^{N} \log \left(p(y_t \mid \mathbf{h}_t^{clean}) \right), \tag{12}$$

where N is the number of valid tokens and y_t is the ground-truth token at time step t.

2.4.2 Soft Cross-Entropy Loss (\mathcal{L}_{soft})

To further guide the model under noise, a soft target distribution is computed from the clean logits $\mathbf{z}^{\text{clean}}$ using temperature scaling:

$$p_t^{\text{soft}} = \operatorname{softmax}\left(\frac{\mathbf{z}_t^{\text{clean}}}{\tau}\right),$$
 (13)

where $\tau > 0$ is the temperature. For a noisy forward pass producing logits $\mathbf{z}^{\text{noisy}}$, the soft cross-entropy loss is:

$$\mathcal{L}_{\text{soft}} = \frac{1}{N} \sum_{t=1}^{N} \text{KL} \Big(p_t^{\text{soft}} \, \Big\| \, \text{softmax}(\mathbf{z}_t^{\text{noisy}}) \Big), \tag{14}$$

with $\mathrm{KL}(\cdot \| \cdot)$ denoting the Kullback–Leibler divergence.

2.4.3 Consistency Loss ($\mathcal{L}_{\text{consistency}}$)

To enforce stability across noisy passes, two independent noisy forward passes are performed yielding logits $\mathbf{z}_1^{\mathrm{noisy}}$ and $\mathbf{z}_2^{\mathrm{noisy}}$. The consistency loss is then defined as:

$$\mathcal{L}_{\text{consistency}} = \frac{1}{N} \sum_{t=1}^{N} \text{KL}\left(\text{softmax}(\mathbf{z}_{1,t}^{\text{noisy}}) \middle\| \text{softmax}(\mathbf{z}_{2,t}^{\text{noisy}})\right). \tag{15}$$

```
Algorithm 1 NoiseFiT Algorithm
```

```
1: Input: Training data \mathcal{D}, pretrained model \mathcal{M}, number of layers k, hyperparameters \lambda_{\text{ce}}, \lambda_{\text{consistency}},
   2: Output: Fine-tuned model \mathcal{M}^*
   3:
            Step 1: Data and Model Preparation
                      Load dataset and format each sample (prompt, response).
    5:
                     Initialize tokenizer and model \mathcal{M}.
   6:
    7:
             Step 2: Clean Forward Pass
   8:
                     For a batch, run a clean forward pass to obtain hidden states \mathbf{h}_{\ell}^{\text{clean}} \in \mathbb{R}^{B \times L_{\ell} \times H}.
   9:
 10:
            Step 3: Noisy Forward Passes
11:
12:
                      Enable noise hooks on selected layers.
                     Run multiple forward passes with adaptive noise injection to obtain \mathbf{h}_{\ell}^{\text{noisy}} \in \mathbb{R}^{B \times L_{\ell} \times H}
13:
14:
            Step 4: SNR Computation (for each layer \ell)
                     Compute signal: S_{\ell} = \frac{1}{B \cdot L_{\ell} \cdot H} \sum_{b,t,i} \left| \left[ \mathbf{h}_{\ell}^{\text{clean}} \right]_{b,t,i} \right|.
16:
                     Compute noise: N_{\ell} = \frac{1}{B \cdot L_{\ell} \cdot H} \sum_{b,t,i} \left[ \left[ \mathbf{h}_{\ell}^{\text{noisy}} - \mathbf{h}_{\ell}^{\text{clean}} \right]_{b,t,i} \right]
17:
                      Compute SNR: SNR_{\ell} = \frac{S_{\ell}}{N_{\ell} + \epsilon}.
 18:
19:
            Step 5: Layer Selection
20:
                     Select k layers with the highest (or lowest) SNR_{\ell} values for noise injection.
21:
22:
            Step 6: Loss Computation
23:
                      (a) Cross-Entropy Loss: \mathcal{L}_{\text{ce}} = -\frac{1}{N} \sum_{t=1}^{N} \log(p(y_t \mid \mathbf{h}_t^{\text{clean}})).
24:
                     (b) Soft Cross-Entropy Loss: \mathcal{L}_{\text{soft}} = \frac{1}{N} \sum_{t=1}^{N} \text{KL}\left(p_t^{\text{soft}} \mid\mid \text{softmax}(\mathbf{z}_t^{\text{noisy}})\right), where p_t^{\text{soft}} = \frac{1}{N} \sum_{t=1}^{N} \left( \sum_{t=1}^{N} \left(
25:
             \operatorname{softmax}\left(\frac{\mathbf{z}_t^{\text{clean}}}{\tau}\right).
                     (c) Consistency Loss: \mathcal{L}_{\text{consistency}} = \frac{1}{N} \sum_{t=1}^{N} \text{KL} \left( \text{softmax}(\mathbf{z}_{1,t}^{\text{noisy}}) \mid\mid \text{softmax}(\mathbf{z}_{2,t}^{\text{noisy}}) \right).
26:
27:
28: Step 7: Final Loss and Backpropagation
                      Compute hybrid loss: \mathcal{L}_{\rm hybrid} = \lambda_{\rm ce} \, \mathcal{L}_{\rm ce} + (1 - \lambda_{\rm ce}) \, \mathcal{L}_{\rm soft}.
29:
30:
                      Compute overall loss: \mathcal{L}_{\text{final}} = \mathcal{L}_{\text{hybrid}} + \lambda_{\text{consistency}} \mathcal{L}_{\text{consistency}}.
31:
                      Backpropagate \mathcal{L}_{\text{final}} and update model parameters.
```

2.4.4 Hybrid Loss and Final Training Objective

The hybrid loss combines the clean and soft cross-entropy losses:

$$\mathcal{L}_{\text{hvbrid}} = \lambda_{\text{ce}} \cdot \mathcal{L}_{\text{ce}} + (1 - \lambda_{\text{ce}}) \cdot \mathcal{L}_{\text{soft}}, \tag{16}$$

where $\lambda_{ce} \in [0,1]$ balances the two objectives. The final training loss, incorporating the consistency regularization, is:

$$\mathcal{L}_{\text{final}} = \mathcal{L}_{\text{hybrid}} + \lambda_{\text{consistency}} \cdot \mathcal{L}_{\text{consistency}}, \tag{17}$$

with $\lambda_{\text{consistency}}$ governing the weight of the consistency loss.

2.5 Fine-Tuning Algorithm

Algorithm 1 summarizes the operational mechanics of the NoiseFiT framework. It outlines the key steps involved in our approach—from data and model preparation, performing clean and noisy forward passes, and computing the SNR for each transformer layer, to select layers for adaptive noise injection and finally calculating the hybrid loss components for backpropagation. This high-level structure serves as a blueprint for implementing our NoiseFiT algorithm.

3 NoiseFiT Mathematical Foundations

In this section, we introduce the theoretical underpinnings of our NoiseFiT framework. We begin by outlining the core assumptions for unbiased noise injection and describe how these assumptions inform the variance-preserving characteristics of our approach. In particular, we provide high-level insights into why adaptive noise regularization improves generalization and stability, setting the stage for the formal lemmas and theorems that follow.

3.1 Unbiased Noise Injection and Variance Preservation

3.1.1 Zero-Mean Noise

Lemma 3.1. Let $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Then,

$$\mathbb{E}[\epsilon] = \mathbf{0}.$$

Proof. Since each component ϵ_i is drawn from a zero-mean Gaussian distribution, we have $\mathbb{E}[\epsilon_i] = 0$ for all i. Therefore, the expectation of the vector $\boldsymbol{\epsilon}$ is:

$$\mathbb{E}[\boldsymbol{\epsilon}] = [\mathbb{E}[\epsilon_1], \mathbb{E}[\epsilon_2], \dots, \mathbb{E}[\epsilon_n]] = [0, 0, \dots, 0] = \mathbf{0}.$$

3.1.2 Unbiasedness of Noisy Representations

Theorem 3.1. Given a hidden state $\mathbf{h} \in \mathbb{R}^n$ and its noisy version defined as:

$$\tilde{\mathbf{h}} = \mathbf{h} + \sigma_{eff} \cdot \boldsymbol{\epsilon},$$

where σ_{eff} is a deterministic (non-random) noise scale and $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, it follows that:

$$\mathbb{E}[\tilde{\mathbf{h}}] = \mathbf{h}.$$

Proof. To compute the expectation of $\tilde{\mathbf{h}}$, we apply the linearity of expectation, which holds for any random vectors and scalars:

$$\mathbb{E}[\tilde{\mathbf{h}}] = \mathbb{E}[\mathbf{h} + \sigma_{\text{eff}} \cdot \boldsymbol{\epsilon}] = \mathbb{E}[\mathbf{h}] + \mathbb{E}[\sigma_{\text{eff}} \cdot \boldsymbol{\epsilon}].$$

Since **h** is a fixed hidden state (not a random variable in this context), $\mathbb{E}[\mathbf{h}] = \mathbf{h}$. Next, because σ_{eff} is deterministic, it can be factored out of the expectation:

$$\mathbb{E}[\sigma_{\text{eff}} \cdot \boldsymbol{\epsilon}] = \sigma_{\text{eff}} \cdot \mathbb{E}[\boldsymbol{\epsilon}].$$

From Lemma 3.1, we know $\mathbb{E}[\epsilon] = 0$. Substituting this in:

$$\mathbb{E}[\sigma_{\text{eff}} \cdot \boldsymbol{\epsilon}] = \sigma_{\text{eff}} \cdot \mathbf{0} = \mathbf{0}.$$

Thus:

$$\mathbb{E}[\tilde{\mathbf{h}}] = \mathbf{h} + \mathbf{0} = \mathbf{h}.$$

This proves the theorem.

3.1.3 Variance Preservation

Lemma 3.2. Assume that the hidden state \mathbf{h} and the noise $\boldsymbol{\epsilon}$ are independent. Then, the covariance of the noised hidden state is given by:

$$\operatorname{Cov}[\tilde{\mathbf{h}}] = \operatorname{Cov}[\mathbf{h}] + \sigma_{eff}^2 \mathbf{I}.$$

Proof. Since $\tilde{\mathbf{h}} = \mathbf{h} + \sigma_{\text{eff}} \cdot \boldsymbol{\epsilon}$ and σ_{eff} is deterministic, we have:

$$\operatorname{Cov}[\tilde{\mathbf{h}}] = \operatorname{Cov}[\mathbf{h}] + \operatorname{Cov}[\sigma_{\text{eff}} \cdot \boldsymbol{\epsilon}] + 2\operatorname{Cov}[\mathbf{h}, \sigma_{\text{eff}} \cdot \boldsymbol{\epsilon}].$$

Because **h** and ϵ are independent, $Cov[\mathbf{h}, \sigma_{eff} \cdot \epsilon] = \mathbf{0}$. Additionally:

$$\operatorname{Cov}[\sigma_{\operatorname{eff}} \cdot \boldsymbol{\epsilon}] = \sigma_{\operatorname{eff}}^2 \operatorname{Cov}[\boldsymbol{\epsilon}] = \sigma_{\operatorname{eff}}^2 \mathbf{I}.$$

Thus:

$$\operatorname{Cov}[\tilde{\mathbf{h}}] = \operatorname{Cov}[\mathbf{h}] + \sigma_{\text{eff}}^2 \mathbf{I}.$$

3.2 Lipschitz Continuity of the Adaptive Noise Injection

Lemma 3.3. Assume that $\sigma_{eff}(\mathbf{h})$ is Lipschitz continuous with constant L_{σ} , and that \mathbf{h} belongs to a bounded set. Then, the mapping

$$\mathcal{T}(\mathbf{h}) = \mathbf{h} + \sigma_{eff}(\mathbf{h}) \cdot \boldsymbol{\epsilon},$$

for a fixed realization of ϵ , is Lipschitz continuous almost surely.

Proof. For any two hidden states \mathbf{h}_1 and \mathbf{h}_2 , we have:

$$\|\mathcal{T}(\mathbf{h}_1) - \mathcal{T}(\mathbf{h}_2)\| = \|\mathbf{h}_1 - \mathbf{h}_2 + (\sigma_{\mathrm{eff}}(\mathbf{h}_1) - \sigma_{\mathrm{eff}}(\mathbf{h}_2)) \cdot \boldsymbol{\epsilon}\|.$$

Using the triangle inequality:

$$\|\mathcal{T}(\mathbf{h}_1) - \mathcal{T}(\mathbf{h}_2)\| \le \|\mathbf{h}_1 - \mathbf{h}_2\| + |\sigma_{\text{eff}}(\mathbf{h}_1) - \sigma_{\text{eff}}(\mathbf{h}_2)| \cdot \|\boldsymbol{\epsilon}\|.$$

Since $\sigma_{\text{eff}}(\mathbf{h})$ is Lipschitz continuous:

$$|\sigma_{\text{eff}}(\mathbf{h}_1) - \sigma_{\text{eff}}(\mathbf{h}_2)| \le L_{\sigma} ||\mathbf{h}_1 - \mathbf{h}_2||.$$

Therefore:

$$\|\mathcal{T}(\mathbf{h}_1) - \mathcal{T}(\mathbf{h}_2)\| \le \|\mathbf{h}_1 - \mathbf{h}_2\| + L_{\sigma}\|\mathbf{h}_1 - \mathbf{h}_2\| \cdot \|\boldsymbol{\epsilon}\| = (1 + L_{\sigma}\|\boldsymbol{\epsilon}\|) \|\mathbf{h}_1 - \mathbf{h}_2\|.$$

For a fixed ϵ , $L = 1 + L_{\sigma} || \epsilon ||$ is a constant, proving that \mathcal{T} is Lipschitz continuous.

3.3 Stability of Hybrid Loss Gradients

Lemma 3.4. Assume that the noise injection is unbiased (Theorem 3.1) and that the noise variance is bounded by σ_{eff}^2 . Then, under smoothness conditions on the network (i.e., the network's output is a differentiable function of \mathbf{h}), the difference between the gradients computed on the clean and noisy hidden states is bounded by $O(\sigma_{\text{eff}})$. In the limit as $\sigma_{\text{eff}} \to 0$, the gradients converge.

Proof. Consider the network output $f(\mathbf{h})$. Using a first-order Taylor expansion around \mathbf{h} :

$$f(\mathbf{h} + \Delta) \approx f(\mathbf{h}) + \nabla f(\mathbf{h})^{\top} \Delta,$$

where $\Delta = \sigma_{\text{eff}} \cdot \boldsymbol{\epsilon}$. Since $\mathbb{E}[\boldsymbol{\epsilon}] = 0$, the expected perturbation is zero. The residual error in the Taylor expansion is $O(\|\Delta\|^2) = O(\sigma_{\text{eff}}^2)$, implying that the gradient difference induced by the noise is $O(\sigma_{\text{eff}})$. As $\sigma_{\text{eff}} \to 0$, this difference approaches zero.

3.4 Robustness of Consistency Loss

Lemma 3.5. The KL divergence $\mathcal{L}_{consistency} = \mathrm{KL}\Big(\mathrm{softmax}(\mathbf{z}^{(1)}) \parallel \mathrm{softmax}(\mathbf{z}^{(2)})\Big)$ is minimized if and only if $\mathrm{softmax}(\mathbf{z}^{(1)}) = \mathrm{softmax}(\mathbf{z}^{(2)}).$

Proof. First, recall that for a vector \mathbf{z} , the softmax function is defined as:

$$\operatorname{softmax}(\mathbf{z})_i = \frac{e^{z_i}}{\sum_j e^{z_j}}$$
 for each component i .

This function transforms the input vector \mathbf{z} into a probability distribution, ensuring that each component softmax(\mathbf{z})_i is positive and that the sum over all components equals 1, i.e., $\sum_i \text{softmax}(\mathbf{z})_i = 1$.

The consistency loss is defined as the KL divergence between two such probability distributions:

$$\mathcal{L}_{\text{consistency}} = \text{KL}\left(\text{softmax}(\mathbf{z}^{(1)}) \parallel \text{softmax}(\mathbf{z}^{(2)})\right) = \sum_{i} \text{softmax}(\mathbf{z}^{(1)})_{i} \log \left(\frac{\text{softmax}(\mathbf{z}^{(1)})_{i}}{\text{softmax}(\mathbf{z}^{(2)})_{i}}\right).$$

By the properties of KL divergence, $\mathrm{KL}(p||q) \geq 0$ for any probability distributions p and q, with equality if and only if p = q almost everywhere. Since $\mathrm{softmax}(\mathbf{z}^{(1)})$ and $\mathrm{softmax}(\mathbf{z}^{(2)})$ are discrete probability distributions (both positive and summing to 1), the KL divergence achieves its minimum value of zero precisely when:

$$\operatorname{softmax}(\mathbf{z}^{(1)})_i = \operatorname{softmax}(\mathbf{z}^{(2)})_i$$
 for all i .

Thus, $\mathcal{L}_{\text{consistency}}$ is minimized if and only if $\operatorname{softmax}(\mathbf{z}^{(1)}) = \operatorname{softmax}(\mathbf{z}^{(2)})$.

3.5 Bound on the Final Loss Due to Noise

We now derive a simple upper bound showing how the presence of adaptive noise injection affects the final training loss. Consider the final loss $\mathcal{L}_{\text{final}}$ in Equation (17), which we write abstractly as a function of the model parameters Θ :

$$\mathcal{L}_{\mathrm{final}}(\Theta) = \underbrace{\lambda_{\mathrm{ce}} \, \mathcal{L}_{\mathrm{ce}}(\Theta) + (1 - \lambda_{\mathrm{ce}}) \, \mathcal{L}_{\mathrm{soft}}(\Theta)}_{\mathcal{L}_{\mathrm{hybrid}}(\Theta)} + \lambda_{\mathrm{consistency}} \, \mathcal{L}_{\mathrm{consistency}}(\Theta).$$

Because each term in $\mathcal{L}_{\text{final}}$ (cross-entropy, KL divergence) is β -smooth [16] with respect to the logits, and the logits are Lipschitz continuous with respect to hidden states **h** (assuming weight matrices are bounded), we can show that random perturbations in **h** of size $\|\Delta\|$ shift the loss by at most $O(\|\Delta\|)$.

Theorem 3.2. Let $\Delta = \tilde{\mathbf{h}} - \mathbf{h}$ be the per-token perturbation introduced by noise injection. Suppose Δ has zero mean and bounded second moment with $\mathbb{E}[\|\Delta\|^2] \leq \sigma_{\max}^2$. Then for sufficiently smooth loss components, the expected deviation in \mathcal{L}_{final} is bounded by

$$\left| \mathbb{E}[\mathcal{L}_{final}(\Theta + \Delta)] - \mathcal{L}_{final}(\Theta) \right| \leq C \sigma_{\max}^2$$

for some constant C > 0 depending on the network's Lipschitz constants and the smoothness of the loss.

Sketch of Proof. Using the first-order Taylor expansion of the logits around the clean hidden states \mathbf{h} , the additional second-order remainder term is $O(\|\Delta\|^2)$. By taking expectations over zero-mean noise with bounded variance, the leading difference in $\mathcal{L}_{\text{final}}$ is $O(\mathbb{E}[\|\Delta\|^2])$. Bounding $\|\Delta\|^2$ by σ_{max}^2 yields the final result.

In essence, because our noise injection is unbiased and has bounded variance, it distorts the final loss only up to a constant factor of σ_{max}^2 . This implies that moderate noise levels do not destabilize training, aligning with our empirical observations.

3.6 Convergence in Expectation

Next, we show that under standard assumptions, NoiseFiT converges in expectation to a local minimum even in the presence of adaptive noise.

Theorem 3.3. Let Θ^* be a local minimum of \mathcal{L}_{final} . Assume:

- (a) \mathcal{L}_{final} is continuously differentiable and bounded from below.
- (b) The gradient $\nabla \mathcal{L}_{final}(\Theta)$ is L-Lipschitz continuous.
- (c) The adaptive noise injection yields unbiased hidden states in expectation (Theorem 3.1) with bounded variance.

Then, performing stochastic gradient descent (or any standard first-order optimizer) on the noised hidden states converges to Θ^* in expectation, i.e.,

$$\lim_{t \to \infty} \mathbb{E} \big[\| \nabla \mathcal{L}_{final}(\Theta_t) \| \big] = 0.$$

Proof Sketch. Under the unbiasedness assumption, each noisy gradient update can be treated as a stochastic gradient with bounded variance. Previous results in stochastic optimization [3] guarantee convergence in expectation as long as the step size schedule satisfies certain decay conditions (e.g., $\alpha_t = \frac{1}{\sqrt{t}}$). The difference between gradients of clean vs. noised hidden states is bounded by $O(\|\Delta\|)$, which is $O(\sigma_{\text{eff}})$, ensuring that the standard convergence proofs hold.

These results suggest that introducing controlled noise into the hidden states—while maintaining bounded variance—does not hinder the convergence of the optimization. In practice, the added benefits of regularization and enhanced robustness often outweigh the minor increase in gradient variance.

4 Experiments and Results

4.1 Experimental Setup

We conducted experiments to assess the effectiveness of NoiseFiT, implemented using PyTorch. Our method incorporates adaptive noise injection, a hybrid loss function, and parameter-efficient fine-tuning (PEFT) using Low-Rank Adaptation (LoRA). We used pre-trained causal language models as base models, varying across architectures including LLaMA, Qwen, Gemma, and Mistral. The fine-tuning dataset was structured to include user prompts and assistant responses in a conversational format.

The fine-tuning process leverages the following key components:

- Layer Selection based on SNR: The trainer selects layers for noise injection based on their SNR. The number of layers selected—set to 3 for most architectures, and to 3, 6, 12, and all layers in the case of Mistral—corresponded to those exhibiting the highest or lowest SNR values. These were identified using forward-pass statistics computed from both clean and noise-injected hidden states.
- Adaptive Noise Injection: During training, zero-mean Gaussian noise was injected into the hidden states of the selected layers. The noise was adaptively scaled based on hidden state statistics, using base standard deviations of 0.001, 0.01, and 0.1, further modulated by layer-specific scaling factors.
- Hybrid Loss Function: The training objective combines three components: a standard cross-entropy (CE) loss, a soft cross-entropy loss based on noisy forward passes, and a consistency loss between two independently noise-injected passes. The total loss is defined as follows:
 - The **cross-entropy loss** \mathcal{L}_{CE} is computed from a clean forward pass and weighted by a factor of $\lambda_{ce} = 0.5$.
 - The **soft cross-entropy loss** \mathcal{L}_{soft} is computed by comparing the logits from noisy forward passes to the soft targets derived from the clean logits, using a temperature-scaled softmax. This loss encourages the noisy predictions to remain close to the clean output distribution.
 - The **consistency loss** $\mathcal{L}_{\text{consistency}}$ measures the KL divergence between two independently sampled noisy forward passes, promoting stability under noise. This term is weighted by $\lambda_{\text{consistency}} = 0.1$.
- **PEFT Configuration**: LoRA was applied with a rank of 8, targeting **q_proj** and **v_proj** modules, with an alpha of 16 and dropout of 0.05.

Training was performed with the following hyperparameters:

- Batch size: 4 per device, with 4 gradient accumulation steps (effective batch size of 16).
- Learning rate: 5×10^{-5} , with a cosine scheduler.
- Epochs: 5, with a maximum of 1000 steps.

- Optimization: Paged AdamW in 32-bit [14], with mixed precision (FP16) and gradient clipping at 1.0.
- Logging: Results were tracked using Weights & Biases [2].

Experiments were conducted by varying the base model architecture, the standard deviation of the injected noise, and the layer selection strategy—specifically, selecting the k layers with the lowest or highest SNR. Model performance was evaluated across six benchmarks: MMLU-PRO, BBH, GPQA, MATH, IFEVAL, and MUSR.

In addition to these benchmarks, each model was also tested on the previously described test dataset, with evaluation performed over five rounds per prompt-response pair for each model.

4.2 Results

4.2.1 LLM Leaderboard Benchmarks

Table 1 summarizes the performance of various model configurations, spanning multiple model families (Llama 1B/3B, Qwen 0.5B, Gemma 3-1B, and Mistral 7B), derived from the leaderboard evaluation task benchmarks [5]. Each model family is evaluated under different noise injection strategies, as indicated by:

- (i) **#Layers:** The number of layers selected for adaptive noise injection (where applicable), with All denoting full-layer injection and BaseFiT indicating a fine-tuning with no noise setup.
- (ii) **STD:** The base standard deviation for noise was typically chosen from the set {0.01, 0.1}. Increasing this value results in stronger perturbations.
- (iii) **SNR:** The signal-to-noise ratio (SNR) selection strategy: *Highest* layers first (favoring more robust activations) vs. *Lowest* layers first (targeting weaker activations), or N/A (e.g., when no targeted noise injection is performed).

Below, we discuss salient trends observed across the reported metrics.

Impact of Noise Levels: As demonstrated in Table 1, injecting noise at various levels (STD=0.01, 0.1, or 0.3) can confer notable performance advantages across multiple tasks and model families. Although higher noise levels (e.g., STD=0.3) are sometimes associated with greater instability, moderate levels (STD=0.01 or 0.1) frequently yield improvements in domains such as Math or BBH. For instance, Llama3.2.1B exhibits enhanced Math accuracy (0.17) under STD=0.1 when targeting layers with high SNR, signifying that carefully calibrated noise can strengthen certain forms of reasoning. Overall, the data suggest that noise injection serves as a viable mechanism for enhancing model robustness, provided the magnitude is appropriately tuned.

Layer Selection via SNR: Examining results for *Highest* vs. *Lowest* signal-to-noise ratio noise injected layers reveals that directing noise to specific layers can accentuate its benefits. For example, Mistral-7B-v0.1 achieves its highest BBH score (45.84) when noise is restricted to three *Lowest*-SNR layers, while Llama3.2.1B attains superior Math performance (0.17) by injecting noise into the *Highest*-SNR layers. These outcomes highlight the importance of selectively targeting layers based on SNR profiles, indicating that the optimal approach may vary according to both the model architecture and the specific task objectives.

BaseFiT vs Noise-Injected Runs: Comparisons with the BaseFiT baseline (fine-tuning without noise) underscore the capacity of noise injection to surpass baseline results in multiple settings. For instance, Qwen2.5-0.5B with STD=0.01 in the high-SNR configuration outperforms BaseFiT on MMLU-Pro (18.75 vs. 17.43) and GPQA (28.36 vs. 26.89). Similarly, Gemma-3-1B realizes substantial gains in majority of the tasks under targeted noise conditions. These findings demonstrate that noise-injected configurations can frequently exceed baseline performance when the noise parameters and layer selections are carefully optimized.

Task-Specific Observations: Across a diverse set of evaluation benchmarks, the impact of noise injection varies by task type and model architecture, but notable patterns emerge. In several cases, injecting moderate levels of noise appears to improve performance, suggesting it may act as a form of regularization or stimulus for deeper reasoning:

- Math: Table 1 shows that moderate noise (STD=0.01 or 0.1) can substantially improve Math accuracy. For example, Llama3.2.1B's score rises from 0.05 to 0.17 under STD=0.1 (highest-SNR layers), and Gemma-3-1B reaches 5.08 (vs. 4.74) under STD=0.01 (highest SNR). These improvements suggest that carefully tuned noise benefits numerical reasoning.
- BBH and MMLU-Pro: These broader language understanding benchmarks often show moderate fluctuation with noise, yet select configurations demonstrate that noise can push performance above the baseline. In Llama-3.2-3B, for example, BBH rises to 39.38 under STD=0.1 (highest SNR), exceeding the BaseFiT score of 38.27. Meanwhile, Mistral-7B-v0.1 attains 45.84 on BBH when injecting noise into three lowest-SNR layers at STD=0.1, reflecting a noticeable jump compared to the baseline (44.34). On MMLU-Pro, for example, Qwen2.5-0.5B at STD=0.01 (highest SNR) rises to 18.75 from a baseline of 17.43. These results confirm that noise, particularly at moderate levels, can be harnessed to refine performance in language understanding tasks.
- **GPQA:** Detailed inspection of the GPQA results shows consistent gains under targeted noise strategies. Llama-3.2-3B moves from 25.41 (BaseFiT) to 27.83 (STD=0.1, Highest SNR), while Qwen2.5-0.5B increases from 26.89 (BaseFiT) to 28.36 (STD=0.01, Highest SNR). Gemma-3-1B also achieves its top GPQA score (28.39) with STD=0.01 in the Highest-SNR layers, surpassing the base 28.00. Notably, Mistral-7B-v0.1 records 30.54 (STD=0.3, three Lowest-SNR layers), indicating that, with the right noise level and layer selection, graduate-level question-answering performance can be enhanced across various model families.
- IFEval and MUSR: Noise can also yield performance improvements in following instructions (IFEval) and multistep soft reasoning (MUSR). In Gemma-3-1B, IFEval rises from 37.52 (BaseFiT) to 39.37 (STD=0.01, Highest SNR), and MUSR increases from 32.75 to 33.41 under the same setting. Likewise, Mistral-7B-v0.1 achieves up to 14.42 on IFEval (STD=0.001, 12 Lowest-SNR layers) compared to 11.46 at BaseFiT, and elevates MUSR from 38.84 (BaseFiT) to 41.49 (STD=0.1, 12 Lowest-SNR layers). While not all configurations consistently outperform the baseline, these cases confirm that strategic noise injection can benefit tasks requiring interpreting instructions and multistep reasoning.

Consistency Across Model Families: Despite variability in architecture and scale (Llama, Qwen, Gemma, Mistral), several consistent observations emerge:

- Efficacy of moderate noise across model-task configurations: While high noise magnitudes (e.g., STD=0.3) can induce instability in performance, moderate perturbation levels (STD=0.01 or 0.1) frequently yield consistent gains across a range of tasks and architectures.
- Targeted perturbation via layer-wise selection: Constraining noise application to specific layer subsets—such as those with the *highest* or *lowest* signal-to-noise ratios—enables more precise control over performance modulation, highlighting the utility of structurally selective noise injection.
- Augmenting BaseFiT with stochastic refinement: Although BaseFiT establishes a robust baseline, many noise-augmented configurations achieve comparable or superior results on specific benchmarks, suggesting that noise injection can function as an effective complement or enhancement to traditional fine-tuning methodologies.

In conclusion, the evidence indicates that noise injection constitutes a promising method for further refining language models. When STD values and layer selections are chosen carefully, noise-driven perturbations can yield notable improvements in multiple task categories, enhancing both robustness and adaptability across varied model families.

4.2.2 Test Dataset

In order to evaluate our fine-tuning approach, we used a test dataset consisting of 208 unique prompts. We employed the same models as the base, incorporating a PEFT adapter. The entire generation procedure was accelerated across multiple GPUs. We employed GROK 3.0 Think [31] to assess the hallucination performance in the generated responses.

Prompt Formatting and Generation: To generate model responses, we formatted each user prompt with specific delimiters (<|im_start|>user ... <|im_end|>) followed by the assistant token. We used the following generation configuration:

	Max. New Tokens	Temperature	\mathbf{Top} - p	\mathbf{Top} - k	Rep. Penalty	
Value	50	0.5	0.9	40	1.2	

This setup allowed us to obtain diverse responses while mitigating overly repetitive outputs. Each local process repeated the generation step for five rounds, independently producing slightly varied outputs for each prompt.

The results presented in Tables A.1 through A.5 illustrate the effects of noise injection on the performance of various models across multiple categories. A consistent trend observed across models is that noise-injection under various scenarios, often outperform their respective base models, suggesting that controlled noise can improve the models' ability to produce less hallucinated responses and handle diverse inputs, potentially by mimicking real-world data variability.

For example, in the Llama-3.2-1B model (Table A.1), the overall performance improves from 48.6% in the base model to 55.8% with noise injection using an STD of 0.1 to high SNR layers. Similarly, the Llama-3.2-3B model (Table A.2) achieves its best overall performance of 70.2% with an STD of 0.01 to high SNR layers, compared to 60.0% in the base model. For the Mistral-7B-v0.1 model (Table A.5), injecting noise into 3 layers with an STD of 0.1 and low SNR yields the highest overall performance of 78.4%, surpassing the base model's 70.6%. However, the impact of noise injection varies across categories; while categories like 'Geography – Landmark Locations' and 'Language' consistently exhibit high performance, others, such as 'Animals' and 'Art (Painting Subjects)', show mixed responses, indicating that the optimal noise parameters may depend on the specific category.

5 Discussion and Conclusions

Our experiments demonstrate that NoiseFiT significantly enhances the performance of LLMs across various leaderboard benchmarks, showcasing its ability to improve model robustness while reducing hallucinations. Across diverse model architectures, including Llama-3.2-1B, Llama-3.2-3B, and Mistral-7B-v0.1, NoiseFiT consistently delivered competitive or superior results on key metrics such as MMLU-Pro, BBH, GPQA, and MUSR compared to baseline fine-tuning methods. For example, in the Llama-3.2-3B model, NoiseFiT improved MMLU-Pro from 25.81 to 25.85 and BBH from 38.27 to 38.56 with noise injection. Similarly, for Mistral-7B-v0.1, MMLU-Pro increased from 30.01 to 30.28, and MUSR rose from 38.84 to 39.51. Even in the Llama-3.2-1B model, where results were more variable—such as a slight dip in BBH from 31.90 to 31.48—the overall performance remained competitive, with gains in other metrics like MUSR underscoring NoiseFiT's effectiveness.

The leaderboard results reveal nuanced trends across tasks and configurations. For instance, in Mistral-7B-v0.1, injecting noise into three layers with the lowest SNR, and with a standard deviation of 0.1 yielded an impressive MUSR score of 41.49, highlighting the potential of tailored noise parameters. Conversely, metrics like Math and IFEval exhibited mixed outcomes depending on the model and noise settings—Llama-3.2-1B, for example, saw slight declines in GPQA from 24.84 to 24.22—suggesting that NoiseFiT's impact is task-dependent. This variability emphasizes the need for task-specific optimization of noise parameters, such as STD, SNR, and the number of layers perturbed, to maximize performance across diverse benchmarks. Nevertheless, the overarching trend across MMLU-Pro, BBH, and MUSR indicates that NoiseFiT strengthens LLM reliability without sacrificing capability.

Complementing these leaderboard findings, evaluations on the custom test dataset of 208 prompts across 17 categories further illuminate NoiseFiT's effectiveness in mitigating hallucinations, particularly in knowledge-intensive tasks. For instance, in the Mistral-7B-v0.1 model, NoiseFiT improved performance in categories like "Medical (Disease Causes)" from 93.3% to 100.0% and "Scientific Discoveries"

from 81.2% to 88.2%, demonstrating its ability to enhance factual accuracy. However, performance in categories such as "Animals" and "Art (Painting Subjects)" remained challenging, with scores below the base model, underscoring the task-dependent nature of NoiseFiT's impact. This mirrors the variability observed in leaderboard metrics like Math and IFEval, reinforcing that while NoiseFiT excels in certain domains, its effectiveness can vary, necessitating tailored approaches for optimal results.

Further insights from the test dataset across multiple models—Llama-3.2-1B, Llama-3.2-3B, Gemma-3-1B, Qwen2.5-0.5B, and Mistral-7B-v0.1—reveal NoiseFiT's broader efficacy in reducing hallucinations beyond standard fine-tuning (BaseFiT). For Llama-3.2-3B, NoiseFiT achieved 70.2% overall performance (3 layers, STD 0.01, highest SNR) compared to BaseFiT's 66.4%, while Qwen2.5-0.5B jumped from 28.8% to 36.6% (3 layers, STD 0.1, highest SNR). In Mistral-7B-v0.1, the optimal configuration (3 layers, STD 0.1, lowest SNR) reached 78.4%, surpassing BaseFiT's 77.2%, with noise injection into fewer layers proving more effective than broader applications (e.g., 12 layers or all layers at 76.4%).

NoiseFiT's success on both benchmarks and the test dataset is bolstered by its adaptive noise injection mechanism, which targets layers based on their SNR to optimize regularization. This approach, paired with a hybrid loss function combining cross-entropy, soft cross-entropy, and consistency regularization, ensures robust training under noisy conditions. Our theoretical analysis further supports these empirical gains, demonstrating that NoiseFiT's noise injection is unbiased, preserves variance, and provides strong convergence guarantees. Practically, NoiseFiT achieves these improvements efficiently, avoiding significant computational overhead, which positions it as a scalable solution for enhancing LLM performance.

While NoiseFiT excels in improving leaderboard standings and reducing hallucinations in specific tasks, its requirement for task-specific tuning of noise parameters presents a limitation, as this process can be resource-intensive. Future work could address this by developing automated techniques—such as hyperparameter optimization or meta-learning—to streamline parameter selection. Exploring its application in high-stakes fields like healthcare or finance could further validate its potential to elevate LLM reliability where precision is paramount. These advancements, complemented by observed reductions in hallucinations across both benchmarks and task-specific evaluations, affirm NoiseFiT as a promising method for refining LLM capabilities.

Table 1: Leaderboard benchmark evaluation results of model families across different #Layers, STD, and SNR

and SNR Model	#Layers	STD	SNR	MMLU-Pro	BBH	GPQA	Math	IFEval	MUSR
	3	0.01	Highest	12.28	31.48	24.22	0.00	7.58	32.46
Llama3.2.1B	3	0.01	Lowest	12.05	31.91	24.90	0.07	7.02	32.07
a3.	BaseFiT	N/A	N/A	11.86	31.90	24.84	0.07	8.69	32.21
mg.	3	0.1	Highest	11.79	31.72	23.99	0.17	7.95	33.54
ΓΪ	3	0.1	Lowest	11.67	31.05	24.56	0.05	6.47	32.75
m	3	0.01	Lowest	25.85	38.56	25.91	0.80	9.43	34.05
2-3	BaseFiT	N/A	N/A	25.81	38.27	25.41	1.31	9.98	34.59
ين. ن	3	0.1	Highest	25.52	39.38	27.83	1.32	8.87	33.79
ma	3	0.01	Highest	25.29	38.64	26.49	1.52	9.80	35.65
Cla	3	0.1	Lowest	25.27	38.39	28.08	1.04	10.91	34.06
<u> </u>	3	0.01	Highest	18.75	31.84	28.36	0.45	14.60	34.20
0.5	3	0.01	Lowest	18.56	31.45	25.13	0.58	13.31	35.40
2.5	3	0.1	Lowest	18.51	31.80	25.07	0.42	13.68	34.59
Gemma-3-1BQwen2.5-0.5BLlama-3.2-3B	3	0.1	Highest	18.29	31.77	28.08	0.68	12.20	34.05
M C	BaseFiT	N/A	N/A	17.43	32.17	26.89	0.70	15.16	34.59
M	BaseFiT	N/A	N/A	14.92	35.11	28.00	4.74	37.52	32.75
က်	3	0.001	Lowest	14.85	34.25	27.88	4.40	34.57	33.95
na-	3	0.1	Highest	13.63	35.14	27.51	2.15	29.21	31.01
l iii	3	0.01	Lowest	14.59	34.54	26.94	4.45	38.08	33.02
5	3	0.01	Highest	14.37	34.84	28.39	5.08	39.37	33.41
	6	0.1	Highest	30.28	44.63	29.46	2.69	13.31	39.51
	12	0.001	Highest	30.14	44.32	29.01	2.49	13.68	39.37
	All	0.01	N/A	30.14	45.12	29.17	2.70	13.86	40.98
	BaseFiT	N/A	N/A	30.01	44.34	29.29	2.97	11.46	38.84
	12	0.1	Lowest	30.01	43.59	28.86	2.52	11.65	41.49
	12	0.01	Lowest	30.00	43.74	28.83	3.05	13.68	39.50
	3	0.1	Lowest	29.97	45.84	28.07	3.01	13.49	39.89
<u> </u>	3	0.01	Lowest	29.95	45.47	25.16	3.35	13.68	39.48
Mistral-7B-v0.1	3	0.01	Highest	29.95	45.56	26.09	2.63	10.91	39.11
7B-	12	0.01	Highest	29.93	44.05	28.86	3.30	14.05	39.77
J	12	0.1	Highest	29.93	44.37	28.90	2.71	13.31	40.30
str	3	0.1	Highest	29.75	44.62	28.36	3.10	10.17	37.65
Mi	6	0.01	Lowest	29.74	45.08	29.66	3.43	11.83	40.18
	6	0.01	Highest	29.72	44.67	28.74	2.78	13.68	38.97
	6	0.1	Lowest	29.67	44.53	30.04	3.19	12.20	38.71
	6	0.001	Lowest	29.59	45.61	30.18	2.26	12.75	39.64
	All	0.001	N/A	29.57	44.51	28.74	2.26	11.65	38.58
	3	0.3	Highest	29.56	44.65	29.67	3.22	12.38	39.38
	3	0.3	Lowest	29.53	44.53	30.54	2.75	10.91	39.51
	12	0.001	Lowest	29.24	45.16	28.48	1.69	14.42	41.23
	All	0.1	N/A	29.00	44.81	29.69	2.45	11.65	40.31

References

- [1] Yejin Bang, Samuel Cahyawijaya, Nayeon Lee, Wenliang Dai, Dan Su, Bryan Wilie, Holy Lovenia, Ziwei Ji, Tiezheng Yu, Willy Chung, Quyet V. Do, Yan Xu, and Pascale Fung. A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity, 2023.
- [2] Lukas Biewald. Experiment tracking with weights and biases, 2020. Software available from wandb.com.
- [3] Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. SIAM review, 60(2):223–311, 2018.
- [4] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, et al. Language models are few-shot learners, 2020.
- [5] Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 07 2024.
- [6] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset, 2021. URL https://arxiv.org/abs/2103.03874, 2024.
- [7] Hang Hua, Xingjian Li, Dejing Dou, Cheng-Zhong Xu, and Jiebo Luo. Improving pretrained language model fine-tuning with noise stability regularization. *IEEE Transactions on Neural Networks and Learning Systems*, 2023.
- [8] Neel Jain, Ping-yeh Chiang, Yuxin Wen, John Kirchenbauer, Hong-Min Chu, Gowthami Somepalli, Brian R Bartoldson, Bhavya Kailkhura, Avi Schwarzschild, Aniruddha Saha, et al. Neftune: Noisy embeddings improve instruction finetuning. arXiv preprint arXiv:2310.05914, 2023.
- [9] 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 Comput. Surv., 55(12), March 2023.
- [10] Jaakko Lehtinen, Jacob Munkberg, Jon Hasselgren, Samuli Laine, Tero Karras, Miika Aittala, and Timo Aila. Noise2noise: Learning image restoration without clean data. arXiv preprint arXiv:1803.04189, 2018.
- [11] Junyi Li, Xiaoxue Cheng, Wayne Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. Halueval: A large-scale hallucination evaluation benchmark for large language models. arXiv preprint arXiv:2305.11747, 2023.
- [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. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 12286–12312, Toronto, Canada, July 2023. Association for Computational Linguistics.
- [13] Litian Liu, Reza Pourreza, Sunny Panchal, Apratim Bhattacharyya, Yao Qin, and Roland Memisevic. Enhancing hallucination detection through noise injection. arXiv preprint arXiv:2502.03799, 2025.
- [14] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101, 2017.
- [15] Michael Moor, Oishi Banerjee, Zahra Shakeri Hossein Abad, Harlan M Krumholz, Jure Leskovec, Eric J Topol, and Pranav Rajpurkar. Foundation models for generalist medical artificial intelligence. Nature, 616(7956):259–265, 2023.
- [16] Yu Nesterov. Smooth minimization of non-smooth functions. *Mathematical programming*, 103:127–152, 2005.

- [17] Mengjia Niu, Hao Li, Jie Shi, Hamed Haddadi, and Fan Mo. Mitigating hallucinations in large language models via self-refinement-enhanced knowledge retrieval, 2024.
- [18] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, et al. Gpt-4 technical report, 2024.
- [19] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, et al. Training language models to follow instructions with human feedback. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc.
- [20] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. In First Conference on Language Modeling, 2024.
- [21] Zayne Sprague, Xi Ye, Kaj Bostrom, Swarat Chaudhuri, and Greg Durrett. Musr: Testing the limits of chain-of-thought with multistep soft reasoning. arXiv preprint arXiv:2310.16049, 2023.
- [22] Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. arXiv preprint arXiv:2210.09261, 2022.
- [23] Cameron Tice, Philipp Alexander Kreer, Nathan Helm-Burger, Prithviraj Singh Shahani, Fedor Ryzhenkov, Jacob Haimes, Felix Hofstätter, and Teun van der Weij. Noise injection reveals hidden capabilities of sandbagging language models. arXiv preprint arXiv:2412.01784, 2024.
- [24] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc.
- [25] Dongwei Wang and Huanrui Yang. Taming sensitive weights: Noise perturbation fine-tuning for robust llm quantization. arXiv preprint arXiv:2412.06858, 2024.
- [26] Song Wang, Zhen Tan, Ruocheng Guo, and Jundong Li. Noise-robust fine-tuning of pretrained language models via external guidance. arXiv preprint arXiv:2311.01108, 2023.
- [27] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models, 2023.
- [28] Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. Mmlu-pro: A more robust and challenging multitask language understanding benchmark. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024.
- [29] Shijie Wu, Ozan Irsoy, Steven Lu, Vadim Dabravolski, Mark Dredze, Sebastian Gehrmann, Prabhanjan Kambadur, David Rosenberg, and Gideon Mann. Bloomberggpt: A large language model for finance. arXiv preprint arXiv:2303.17564, 2023.
- [30] Weibin Wu, Yuxin Su, Xixian Chen, Shenglin Zhao, Irwin King, Michael R Lyu, and Yu-Wing Tai. Boosting the transferability of adversarial samples via attention. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1161–1170, 2020.
- [31] xAI. Grok 3 (think mode) [computer software], 2025. Accessed: March 2025 to April 2025.
- [32] Abhay Kumar Yadav and Arjun Singh. Symnoise: Advancing language model fine-tuning with symmetric noise. arXiv preprint arXiv:2312.01523, 2023.
- [33] Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. *Defending against neural fake news*. Curran Associates Inc., Red Hook, NY, USA, 2019.
- [34] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, et al. A survey of large language models, 2025.

- [35] Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. arXiv preprint arXiv:2311.07911, 2023.
- [36] Kaijie Zhu, Jindong Wang, Jiaheng Zhou, Zichen Wang, Hao Chen, Yidong Wang, Linyi Yang, Wei Ye, Yue Zhang, Neil Zhenqiang Gong, and Xing Xie. Promptrobust: Towards evaluating the robustness of large language models on adversarial prompts, 2024.

A Technical Appendix

A.1 Test Performance Analysis

This appendix presents detailed experimental results for NoiseFiT in mitigating hallucinations of LLMs based on the test dataset. The evaluated models include Llama-3.2-1B, Llama-3.2-3B, Gemma-3-1B, Qwen2.5-0.5B, and Mistral-7B-v0.1. For each model, performance is assessed across 17 distinct categories of prompts, encompassing a total of 208 prompts, under multiple configurations: the base model, the base model with fine-tuning (denoted BaseFiT), and several noise-injected variants using NoiseFiT. These NoiseFiT configurations vary by the number of layers affected, the standard deviation (STD) of the injected noise (e.g., 0.001, 0.01, 0.1), and the signal-to-noise ratio (SNR), where 'L' denotes the lowest SNR (highest noise relative to signal were selected for noise injection) and 'H' denotes the highest SNR (lowest noise relative to signal were selected for noise injection). Performance metrics are averaged across five runs per prompt to ensure statistical reliability (Supplementary Material).

The tables in this appendix (Tables A.1 to A.5) provide category-wise performance scores alongside overall performance metrics for each model and configuration. This enables a comprehensive evaluation of how NoiseFiT mitigates hallucinations across different tasks and setups.

A.1.1 Analysis of Results

The results in this appendix highlight the effectiveness of NoiseFiT in mitigating hallucinations in LLMs, demonstrating both general trends across models and specific insights tailored to this task. Below, we analyze these findings, with an in-depth focus on the Mistral-7B-v0.1 model due to its comprehensive set of noise injection configurations.

General Performance Trends Across Models: Fine-tuning the base models (BaseFiT) generally improves performance over the untrained base models, serving as a foundational step in reducing hallucinations by better aligning the model with the training data. For Llama-3.2-1B, overall performance increases from 48.6% to 54.0%; for Llama-3.2-3B, from 60.0% to 66.4%; for Qwen2.5-0.5B, from 26.4% to 28.8%; and for Mistral-7B-v0.1, from 70.6% to 77.2%. However, Gemma-3-1B shows a decline from 50.6% to 47.6% with BaseFiT, suggesting that standard fine-tuning alone may not always mitigate hallucinations effectively and could even exacerbate them in some cases.

NoiseFiT, designed specifically to tackle hallucinations, frequently enhances performance beyond BaseFiT, particularly in categories prone to factual inaccuracies. For Llama-3.2-3B, the best NoiseFiT variant (3 layers, STD 0.01, highest SNR) achieves 70.2%, surpassing BaseFiT's 66.4%. Qwen2.5-0.5B improves significantly from 28.8% (BaseFiT) to 36.6% (3 layers, STD 0.1, highest SNR). In Gemma-3-1B, NoiseFiT recovers performance to 54.6% (3 layers, STD 0.1, highest SNR) from BaseFiT's 47.6%, exceeding the base model's 50.6%. These improvements indicate that NoiseFiT's noise injection enhances the model's ability to generalize, reducing its tendency to hallucinate by regularizing its learned representations.

Mistral-7B-v0.1: The Mistral-7B-v0.1 model, with its 7 billion parameters, provides a robust case study for evaluating NoiseFiT's impact on hallucination mitigation, as it was tested with noise applied to 3 layers, 12 layers, or all layers, across various STDs and SNRs. The key findings are as follows:

- Optimal Number of Layers for Noise Injection: Injecting noise into fewer layers (specifically 3 layers) consistently outperforms configurations with noise applied to 12 layers or all layers in mitigating hallucinations. The highest overall performance of 78.4% is achieved with 3 layers, STD 0.1, and lowest SNR, compared to 77.2% for BaseFiT. In contrast, the best 12-layer configuration (STD 0.1, lowest SNR) yields 76.4%, and the best all-layer configuration (STD 0.1) also reaches 76.4%. This suggests that targeting a small, critical subset of layers with noise injection enhances the model's ability to distinguish correct from incorrect information, effectively reducing hallucinations, while broader noise application may disrupt learned hidden states excessively.
- Impact of Noise Level and SNR: Within the 3-layer configurations, higher noise levels (STD 0.1) paired with lower SNR (more noise relative to signal) outperform lower noise levels or higher SNR settings. For example, 3 layers with STD 0.1 and lowest SNR achieves 78.4%, while the same STD with highest SNR yields 77.4%, and STD 0.01 with lowest and highest SNR scores 76.5% and 77.3%, respectively. This indicates that substantial noise, when carefully applied, acts as a strong regularizer in Mistral-7B-v0.1, reducing overconfidence in incorrect outputs and thus mitigating hallucinations. The preference for lower SNR underscores the benefit of higher noise intensity in this context.

- Category-wise Performance Variations: NoiseFiT significantly improves performance in categories where hallucinations are particularly prevalent. In "Medical (Disease Causes)," performance reaches 100.0% across multiple 3-layer configurations (e.g., STD 0.1, lowest SNR), up from 93.3% in BaseFiT. "Scientific Discoveries" improves from 81.2% to 88.2% (3 layers, STD 0.01, lowest SNR), "Who Invented" from 82.1% to 85.2% (3 layers, STD 0.1, highest SNR), and "Sports (Famous Players)" from 74.7% to 92.0% (3 layers, STD 0.01, lowest SNR). These gains highlight NoiseFiT's effectiveness in enhancing factual accuracy and reducing hallucinations in knowledge-intensive tasks. However, categories like "Animals" (BaseFiT: 34.1%, best NoiseFiT: 43.6% with 12 layers, STD 0.001, highest SNR, still below base's 63.5%) and "Art (Painting Subjects)" (BaseFiT: 32.2%, best NoiseFiT: 40.0% with 12 layers, STD 0.1, highest or lowest SNR, below base's 42.2%) show persistent challenges, indicating that NoiseFiT may not fully mitigate hallucinations in tasks requiring nuanced or context-sensitive understanding.
- Comparison with Other Models: Unlike smaller models like Llama-3.2-1B (best: STD 0.1, highest SNR, 55.8%) or Qwen2.5-0.5B (best: STD 0.1, highest SNR, 36.6%), where higher SNR (less noise) often performs better, Mistral-7B-v0.1 favors lower SNR (more noise) in its optimal configuration. This difference likely reflects Mistral's larger capacity, allowing it to benefit from higher noise levels as a stronger regularizer against hallucinations, whereas smaller models may be more sensitive to noise, requiring lower levels to maintain stability.
- Layer Selection Implications: The superior performance of the 3-layer configuration suggests an optimal subset exists—possibly layers critical with high variance. Broader noise application (12 layers or all layers) reduces effectiveness (e.g., 12 layers, STD 0.001, lowest SNR: 77.0%; all layers, STD 0.001: 74.4%), emphasizing the importance of layer selection strategy for noise injection in mitigating hallucination.

These findings demonstrate that for Mistral-7B-v0.1, injecting significant noise (STD 0.1, lowest SNR) into a small, targeted set of layers (3 layers) optimizes performance, slightly surpassing BaseFiT and outperforming broader noise applications in reducing hallucinations. The category-wise analysis reveals substantial benefits in factual, knowledge-based tasks, though challenges persist in areas like "Animals" and "Art," suggesting limitations in NoiseFiT's applicability across those domains for our specific test dataset.

In conclusion, NoiseFiT proves to be a promising technique for mitigating hallucinations in LLMs, particularly in knowledge-intensive categories, by leveraging noise injection to enhance robustness and reduce overconfidence in incorrect outputs. However, its effectiveness varies across tasks and models, necessitating task-specific optimization of noise injection parameters and further research to address remaining challenges in certain domains.

A.2 Computational Efficiency and Scalability Analysis

To evaluate the computational efficiency and scalability of the proposed NoiseFiT framework compared to the common fine-tuning (BaseFiT), we analyzed a series of GPU performance metrics recorded during the experiments. The metrics under consideration include:

- GPU Memory Allocated (%) Indicates the percentage of total GPU memory used.
- GPU Power Usage (%) Reflects the power consumption during model training.
- GPU Temperature (°C) Monitors the thermal performance of the GPU.
- Time Spent Accessing Memory (%) Measures the relative time the GPU spent in memory operations.
- GPU Utilization (%) Captures the overall usage of the GPU computational resources.

For each metric, we computed the mean and standard deviation over multiple experimental runs. Table A.6 summarizes the performance for both BaseFiT (Base) and NoiseFiT configurations. The results indicate that the NoiseFiT framework exhibits a mixed performance profile across the evaluated GPU metrics:

• Memory and Power Efficiency: While NoiseFiT requires a higher GPU memory allocation (61.3% vs. 35.5%), it achieves reduced power usage (64.0% vs. 67.3%). This suggests that, despite the increased memory demand, NoiseFiT benefits from lower energy consumption during training.

Table A.1: Category-wise Performance of Llama-3.2-1B Configurations. Performance is averaged across 5 runs per prompt (208 prompts total). For noise-injected cases (3 layers), the first value is the standard deviation (STD), with 'L' indicating Lowest SNR and 'H' indicating Highest SNR.

	Llama3.2.1B (3 layers if noise injected)							
Category	Base	BaseFiT	0.1, L	0.1, H	0.01, L	0.01, H		
Medical (Disease Causes)	76.6	50.0	53.4	60.0	56.6	63.4		
Geography – Landmarks	65.4	69.0	81.8	85.4	80.0	81.8		
Geography – Capitals	66.6	75.0	50.0	75.0	98.4	80.0		
Geography – Currency	34.6	66.6	65.4	74.6	77.4	74.6		
Geography – Landmark Locations	53.4	100.0	100.0	91.6	91.6	100.0		
Language	80.0	100.0	100.0	100.0	100.0	100.0		
History (Year Events)	54.6	91.0	100.0	100.0	87.2	85.4		
History (When Events)	48.4	98.4	100.0	96.6	96.6	93.4		
Inventions	40.0	25.0	41.2	40.0	38.8	22.6		
Animals	14.2	34.2	29.4	36.4	22.4	24.8		
Music/Composers	43.4	53.4	46.6	50.0	50.0	60.0		
Scientific Discoveries	57.6	36.4	36.4	41.2	35.2	29.4		
Who Invented	45.2	32.6	42.2	33.6	35.8	37.8		
Sports (Famous Players)	74.6	30.6	26.6	29.4	30.6	32.0		
Art (Painting Subjects)	22.2	13.4	12.2	10.0	13.4	12.2		
Literature	60.0	70.6	60.0	64.2	67.4	53.6		
Miscellaneous	80.0	100.0	100.0	100.0	100.0	100.0		
Overall	48.6	54.0	53.4	55.8	55.4	52.4		

- Thermal Performance and Memory Operations: The GPU temperature and the time spent accessing memory are marginally elevated in the NoiseFiT setup (58.6°C and 50.7%, respectively) compared to BaseFiT (57.8°C and 49.0%). These slight differences indicate that thermal management and memory operation times remain largely comparable between the two approaches.
- Overall Utilization: The slightly lower overall GPU utilization observed with NoiseFiT (75.5% vs. 77.2%) implies that similar or improved performance may be achieved with a reduced computational load, which is beneficial for scalability.

In summary, the performance trade-offs observed with the NoiseFiT suggest a viable balance between computational efficiency and resource allocation. Although NoiseFiT demands higher memory usage and shows marginal increases in thermal metrics, its reduced power consumption and overall GPU utilization indicate that it can mitigate hallucinations while decreasing the computational overhead associated with training. These benefits are especially critical when scaling large language models in resource-constrained environments, thereby enhancing both the practicality and the environmental sustainability of deploying such systems.

Table A.2: Category-wise Performance of Llama-3.2-3B Configurations. Performance is averaged across 208 prompts total. For noise-injected cases (3 layers), the first value is the standard deviation (STD), with 'L' indicating Lowest SNR and 'H' indicating Highest SNR.

Llama3.2.3B (3 layers if noise injected)								
Category	Base	BaseFiT	0.1, L	0.1, H	0.01, L	0.01, H		
Medical (Disease Causes)	73.4	63.4	80.0	63.4	70.0	80.0		
Geography – Landmarks	92.8	100.0	98.4	94.6	96.7	85.4		
Geography – Capitals	78.3	91.6	85.0	91.6	91.6	91.7		
Geography – Currency	84.0	100.0	98.6	100.0	98.7	100.0		
Geography – Landmark Locations	86.7	96.6	100.0	100.0	100.0	100.0		
Language	80.0	100.0	100.0	100.0	100.0	100.0		
History (Year Events)	85.5	90.9	63.6	81.8	89.1	96.4		
History (When Events)	70.0	91.6	91.6	91.6	91.6	98.3		
Inventions	33.8	48.8	47.6	50.0	43.8	37.5		
Animals	27.1	22.4	11.8	15.2	16.5	32.9		
Music/Composers	76.7	60.0	66.6	50.0	63.3	63.3		
Scientific Discoveries	49.4	48.2	49.4	58.8	52.9	55.3		
Who Invented	51.6	66.4	75.8	83.2	82.1	74.7		
Sports (Famous Players)	10.7	49.4	54.6	57.4	53.3	60.0		
Art (Painting Subjects)	45.6	25.6	16.6	21.2	16.7	20.0		
Literature	83.2	77.8	81.0	84.2	83.2	93.7		
Miscellaneous	100.0	100.0	100.0	100.0	100.0	100.0		
Overall	60.0	66.4	65.6	68.2	68.0	70.2		

Table A.3: Category-wise Performance of Gemma-3-1B Configurations. Performance is averaged across 208 prompts total. For noise-injected cases (3 layers), the first value is the standard deviation (STD), with 'L' indicating Lowest SNR and 'H' indicating Highest SNR.

	ayers if noi	se injected)				
Category	Base	BaseFiT	0.1, H	0.01, H	0.001, L	0.01, L
Medical (disease causes)	43.4	76.7	60.0	70.0	66.6	70.0
Miscellaneous	80.0	100.0	100.0	20.0	100.0	80.0
Geography – Landmarks	43.6	85.5	92.8	80.0	83.6	76.4
Geography – Capitals	78.4	100.0	100.0	100.0	100.0	100.0
Geography – Currency	73.4	93.3	86.6	97.4	96.0	92.0
Language	80.0	100.0	100.0	100.0	100.0	100.0
History (Year events)	83.6	81.8	81.8	81.8	91.0	80.0
History (When events)	76.6	41.7	76.6	81.6	60.0	40.0
Inventions	55.0	32.5	40.0	37.6	31.2	27.6
Geography – Landmark Locations	70.0	66.7	100.0	100.0	100.0	20.0
Animals	31.8	7.1	21.2	10.6	9.4	9.4
Music/Composers	26.6	23.3	26.6	26.6	40.0	33.4
Scientific Discoveries	42.4	36.5	29.4	27.0	24.8	24.8
Who Invented	65.2	45.3	63.2	57.8	50.6	53.6
Sports (Famous Players)	26.6	30.7	32.0	28.0	26.6	30.6
Art (Painting Subjects)	7.8	8.9	7.8	16.6	14.4	7.8
Literature	44.2	31.6	40.0	35.8	31.6	34.8
Overall	50.6	47.6	54.6	53.2	51.0	43.8

Table A.4: Category-wise Performance of Qwen2.5-0.5B Configurations. Performance is averaged across 208 prompts total. For noise-injected cases (3 layers), the first value is the standard deviation (STD), with 'L' indicating Lowest SNR and 'H' indicating Highest SNR.

	Qwen2.5-0.5B (3 layers if noise injected)								
Category	Base	BaseFiT	0.1, L	0.1, H	0.01, L	0.01, H			
Medical (disease causes)	66.7	76.6	80.0	80.0	86.6	73.4			
Miscellaneous	60.0	100.0	100.0	100.0	20.0	100.0			
Geography – Landmarks	21.8	31.0	16.4	40.0	18.2	36.4			
Geography – Capitals	31.7	75.0	85.0	86.6	76.6	71.6			
Geography – Currency	38.7	50.6	74.6	77.4	69.4	78.6			
Language	80.0	100.0	20.0	100.0	80.0	100.0			
History (Year events)	43.6	45.4	63.6	58.2	51.0	61.8			
History (When events)	38.3	68.4	73.4	73.4	61.6	66.6			
Inventions	23.8	5.0	11.2	10.0	11.2	7.6			
Geography – Landmark Locations	60.0	93.4	88.4	93.4	81.6	80.0			
Animals	17.6	5.8	3.6	18.8	9.4	7.0			
Music/Composers	0.0	0.0	0.0	0.0	0.0	0.0			
Scientific Discoveries	28.2	11.8	11.8	13.0	7.0	14.2			
Who Invented	21.0	10.6	11.6	7.4	11.6	8.4			
Sports (Famous Players)	9.3	6.6	9.4	17.4	9.4	20.0			
Art (Painting Subjects)	3.3	3.4	1.2	3.4	2.2	1.2			
Literature	16.8	8.4	26.4	25.2	20.0	20.0			
Overall	26.4	28.8	33.0	36.6	30.2	33.0			

Table A.5: Category-wise Performance of Mistral-7B-v0.1 Configurations. Performance is typically averaged across 208 prompts. For noise-injected cases, column labels show #Layers, (STD, SNR); with 'L' indicating Lowest SNR and 'H' indicating Highest SNR.

								Mistral-7B-v	0.1						
Category	Base	BaseFiT	3 (0.1, L)	3 (0.1, H)	3 (0.01, L)	3 (0.01, H)	12 (0.001, L)	12 (0.001, H)	12 (0.01, L)	12 (0.01, H)	12 (0.1, L)	12 (0.1, H)	All (0.001)	All (0.01)	All (0.1)
Medical (Disease Causes)	90.0	93.3	100.0	96.6	100.0	100.0	83.4	83.4	83.4	83.4	83.4	83.4	83.4	83.4	83.4
Miscellaneous	80.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
Geography – Landmarks	90.9	96.4	89.1	94.6	94.6	98.2	98.2	100.0	100.0	96.4	72.8	100.0	100.0	100.0	78.2
Geography – Capitals	93.3	100.0	100.0	100.0	100.0	100.0	100.0	100.0	96.6	100.0	100.0	100.0	100.0	100.0	100.0
Geography – Currency	81.3	100.0	100.0	100.0	98.6	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
Language	80.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
History (Year Events)	89.1	96.4	90.9	91.0	91.0	90.9	100.0	87.2	92.8	89.0	98.2	96.4	100.0	100.0	98.2
History (When Events)	91.7	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
Inventions	37.5	58.8	53.8	56.2	50.0	52.5	58.8	67.6	56.2	58.8	55.0	43.8	45.0	47.6	56.2
Geography – Landmark Locations	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
Animals	63.5	34.1	38.8	25.8	29.4	36.5	35.2	43.6	33.0	35.2	34.2	30.6	24.8	31.8	34.2
Music/Composers	73.3	80.0	66.7	70.0	66.6	66.7	83.4	66.6	83.4	83.4	83.4	83.4	83.4	83.4	83.4
Scientific Discoveries	57.6	81.2	82.4	83.6	88.2	81.2	76.4	76.4	76.4	76.4	74.2	82.4	75.2	65.8	76.4
Who Invented	66.3	82.1	82.1	85.2	84.2	78.9	81.0	69.4	73.6	73.6	76.8	73.6	79.0	73.6	77.8
Sports (Famous Players)	48.0	74.7	90.7	89.4	92.0	77.3	82.6	81.4	69.4	77.4	85.4	77.4	80.0	88.0	82.6
Art (Painting Subjects)	42.2	32.2	37.8	28.8	33.4	30.0	31.2	23.4	38.8	33.4	40.0	40.0	27.8	38.8	38.8
Literature	81.1	75.8	78.9	68.4	82.2	76.8	71.6	73.6	63.2	75.8	80.0	67.4	70.6	75.8	70.6
Overall	70.6	77.2	78.4	77.4	76.5	77.3	77.0	75.6	74.4	75.8	76.4	75.2	74.4	75.8	76.4

Table A.6: Summary of GPU performance metrics statistics comparing Base Fine Tuning (BaseFiT) and Noisy Fine Tuning (NoiseFiT) workflows. Values represent the mean \pm standard deviation across multiple runs.

Metric	${\bf Base FiT}$	NoiseFiT
GPU Memory Allocated (%)	35.5 ± 0.0	61.3 ± 0.5
GPU Power Usage (%)	67.3 ± 14.0	64.0 ± 16.1
GPU Temperature (°C)	57.8 ± 2.2	58.6 ± 1.2
Time Spent Accessing Memory (%)	49.0 ± 13.5	50.7 ± 19.4
GPU Utilization (%)	77.2 ± 20.6	75.5 ± 18.3