Small Data, Big Noise: Adversarial Training for Robust Parameter-Efficient Fine-Tuning

Eitan Cohen
Bar-Ilan University
Ramat Gan, Israel
ceitan001@gmail.com

Idan Simai
Bar-Ilan University
Ramat Gan, Israel
idansi98@gmail.com

Uri Shaham
Bar-Ilan University
Ramat Gan, Israel
uri.shaham@biu.ac.il

Abstract

Parameter-Efficient Fine-Tuning (PEFT) has become essential for adapting foundation models to downstream NLP tasks. However, current PEFT methods often struggle with robustness to noise and performance degradation on limited training data. We propose **SDBN** (Small Data Big Noise), a unified framework that brings adversarial training to PEFT - a combination that remains less studied in the PEFT setting despite its complementary strengths - to enhance model robustness and generalization, outperforming alternative approaches. We also introduce two variants of the method that use discrete uncertainty sets: **SDBN-h**, which enumerates character-level edits and selects worst-case variants using gradients, and **SDBN-p**, which uses LLM-generated variants for robust optimization in generative tasks. Experiments across multiple benchmarks reveal substantial improvements, particularly in low-resource settings and under both word-level and character-level corruptions. This framework addresses the less explored intersection of adversarial training and parameter-efficient adaptation, without introducing additional parameters or only modest computational overhead, making PEFT deployments more reliable in real-world scenarios where data scarcity and linguistic variability often coexist.

1 Introduction

Parameter-Efficient Fine-Tuning (PEFT) has recently emerged as a promising strategy for adapting Large Language Models (LLMs) to downstream tasks, while substantially reducing both computational and storage requirements. Notable PEFT approaches include Adapter (Houlsby et al., 2019), BitFit (Ben-Zaken et al., 2021), and Low-Rank Adaptation (LoRA) (Hu et al., 2021). In particular, LoRA has garnered significant attention, spurring the development of several variants (Ren et al., 2024), (Zhang et al., 2023),(Dettmers et al., 2023) which further extend its applicability.

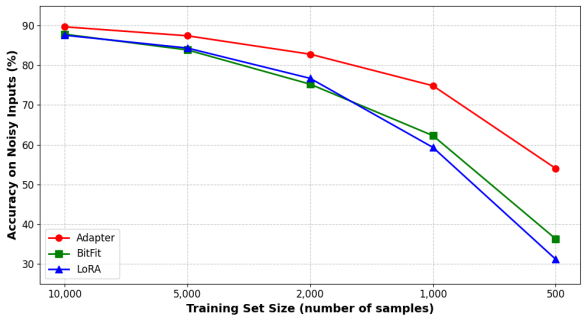


Figure 1: **Performance degradation of PEFT methods on limited training data.** Accuracy of three common PEFT methods: Adapter, BitFit, and LoRA–on a test set with *word-swapping* noise as the size of the *clean* Banking77 training data corpus is reduced (x-axis, number of examples). All methods exhibit a marked drop in noisy-test accuracy as the available training data *dwindles*, in some cases losing more than 50% below 1,000 training samples. The trend underscores the current vulnerability of PEFT models to realistic textual perturbations in low-resource settings.

While these techniques substantially reduce the computational overhead required for fine-tuning foundation models, they frequently underperform when confronted with domain shifts and noisy data (see fig. 1) - conditions that commonly appear in practical NLP deployments. Real-world text often contains imperfections such as typos, inconsistent formatting, and dialectal variations that can significantly degrade model performance. Additionally, while PEFT approaches have demonstrated impressive results on large-scale training corpora, their effectiveness may deteriorate significantly when training samples are limited - a common scenario in domains such as medical data, aerospace, extinct languages, and similar fields. Notably, (Ben-Zaken et al., 2021) shows that PEFT methods can be more efficient than full fine-tuning on small datasets, motivating a focused investigation of PEFT approaches tailored specifically for settings with limited data. These challenges–robustness to noise and

Findings of the Association for Computational Linguistics: ACL 2026, pages 40509–40532

domain shifts, as well as worse performance in low-resource settings–remain largely underexplored in current PEFT methodologies.

Motivated by these challenges, we propose SDBN (Small Data Big Noise)[1], a unified framework that integrates adversarial training principles into PEFT to enhance robustness and generalization capabilities, including robustness to tokenization-breaking *character-level* corruptions in addition to word-level noise. While adversarial training has been widely adopted in various contexts (Zhu et al., 2020; Jiang et al., 2019; Ji et al., 2024)–primarily for defense against attacks–its application to improving PEFT methods in low-resource, noisy settings remains unexplored. The proposed SDBN framework improves model performance under both noisy conditions (word- and character-level) and limited training data scenarios. Beyond continuous embedding-space perturbations, we introduce two complementary strategies for constructing discrete uncertainty sets: SDBN-h, which employs gradient-guided character-level edits to address tokenization-breaking noise, and SDBN-p, which leverages LLM-generated adversarial variants for richer semantic perturbations particularly suited to generative tasks. These discrete uncertainty-set instantiations replace norm ball uncertainty sets in regimes it cannot cover - tokenization-breaking character edits (SDBN-h) and semantic, generation-oriented variants (SDBN-p). Unlike most PEFT approaches - which do not explicitly target scenarios with restricted data or address domain shifts - SDBN achieves substantial improvements across various benchmark datasets. Notably, this framework provides robustness not only to known perturbations but also to unanticipated domain shifts that may emerge during deployment, without requiring domain-specific adaptation data. Importantly, SDBN maintains the parameter efficiency of existing PEFT methods without adding trainable parameters or extra GPU memory overhead, offering a practical solution for deploying robust language models in resource-constrained and noisy real-world environments.

Our Contributions are as follows: First, We propose **SDBN**, a framework that brings adversarial training to PEFT - a combination on **less studied in the PEFT setting** - to improve robustness in low-resource settings. Second, we introduce two variants of the method utilizing discrete uncertainty sets: **SDBN-h** based on character-level perturbation, and **SDBN-p** based on LLM-generated perturbations. Third, we demonstrate **substantial robustness gains** across multiple benchmarks under word-level and character-level corruptions, without additional parameters.

[1]**Code:** https://github.com/shaham-lab/SDBN

2 Related work

PEFT methods. Recent advances in PEFT methods-such as LoRA, Adapter, BitFit, Prompt (Liu et al., 2021) and Prefix (Li and Liang, 2021)–have facilitated the efficient adaptation of pre-trained models by fine-tuning only a limited set of parameters. In particular, LoRA employs low-rank updates, Adapter incorporates trainable bottleneck modules within each layer, and BitFit restricts modifications to bias parameters. Notably, recent evaluations demonstrate that PEFT methods like LoRA actually exhibit superior robustness to textual noise and adversarial perturbations compared to full fine-tuning (More, 2025). However, even with this inherent advantage, these models remain vulnerable to realistic corruptions in data-scarce regimes, providing a strong foundation for the targeted robust-optimization mechanisms introduced in SDBN. While these approaches reduce parameter counts, most standard PEFT implementations were not originally designed with robustness to noisy inputs (Kim et al., 2024) or data-limited scenarios as primary optimization targets. Our work explores how integrating adversarial training techniques with existing PEFT methods can address these limitations, particularly in low-resource settings.

Robust optimization. Robust optimization is widely recognized as an effective strategy for enhancing model stability in the presence of noisy data and domain shifts. (Shaham et al., 2018), (Madry et al., 2019) demonstrated that robust optimization can significantly improve a model's generalization capabilities on noisy data and downstream tasks, using neural networks; however, the robust optimization approaches discussed in their works, which directly inject noise into raw input data, require specific adaptations for the NLP domain and were not evaluated within PEFT frameworks. (Kim et al., 2024) applied robust optimization to address noisy labels but did not consider noisy input data. Moreover, neither approach focused on optimization under conditions of limited data. In contrast, the integration of adversarial training with PEFT

frameworks explored in this work addresses both noisy inputs and small datasets simultaneously.

Adversarial Training. Using adversarial training is one way to achieve robust optimization. FreeLB (Zhu et al., 2020), SMART (Jiang et al., 2019) and VAT (Miyato et al., 2021) applied adversarial training to LLMs, primarily focusing on improving generalization capabilities; however, those do not incorporate PEFT methodologies, which can render full fine-tuning impractical in certain scenarios. Recent methods such as LoFT (Fu et al., 2024) and AdvLoRA (Ji et al., 2024) apply adversarial training with LoRA on vision tasks (in the case of AdvLoRA, mainly on the visual component of vision-language tasks), leaving their applicability in NLP domains unexplored. Notably, the effectiveness of adversarial training with PEFT methods in scenarios with small datasets and noisy textual inputs remains largely uninvestigated. This work examines how adversarial training can be applied to PEFT paradigms in NLP and demonstrates enhanced performance over conventional PEFT baselines, particularly in low-resource settings.

PEFT in NLP under low resources. PEFT is now the de-facto strategy for adapting large language models to downstream tasks, yet its behaviour under noisy or limited data remains understudied. NEFtune (Jain et al., 2023) shows that injecting *uniform random* noise into token embeddings during instruction tuning (NEFTune) markedly improves *average-case* performance, focusing primarily on enhancing model generalization, but they do not analyse worst-case perturbations or data-scarce regimes. In contrast, this work applies *adversarial* training to PEFT with a different objective: adding adversarial perturbations in the embedding space and optimising adapters to minimise the resulting worst-case loss. This targeted approach yields robustness to input noise and domain shift specifically in scenarios with small, noisy datasets–precisely the conditions where PEFT methods are most practically attractive but often struggle without additional robustness mechanisms.

Robustness through data noising. Several works have shown that even small character edits (e.g., missing letters) can cause degradation in NLP models, since such changes often break tokenization and push inputs far in embedding space. EDA (Wei and Zou, 2019) proposed simple data augmentation editing actions, showing improvements on small datasets; however, these perturbations are applied randomly rather than guided by gradients. HotFlip (Ebrahimi et al., 2018) introduced gradient-guided character-level edits, but as an attack tool. WildNLP (Rychalska et al., 2019) and Aepli and Sennrich (2022) explored noise-augmented training to improve robustness to character corruptions, but without leveraging adversarial signals to identify worst-case perturbations. In contrast, our work adapts character-level perturbations within an adversarial training framework, using gradient guidance to maximize their effectiveness for improving robustness in low-resource PEFT settings.

Discrete uncertainty sets. Several prior works use discrete perturbation sets in NLP, but in settings that differ substantially from ours. Some focus on certified robustness via verification-style pipelines such as randomized masking or interval bound propagation, e.g., (Zeng et al., 2023), (Huang et al., 2019), and (Jia et al., 2019). These methods target formal guarantees against bounded substitution attacks rather than training-time robust optimization for PEFT, and are not naturally scalable to large generative models or practical PEFT training loops. Other work, such as (Zhou et al., 2021), proposes an inference-time defense against synonym substitution attacks rather than adversarial training. Closest in spirit is (Ivgi and Berant, 2021), which studies discrete adversarial training for classification; however, it assumes full fine-tuning, focuses on attack-style substitutions rather than practical small-data noise, and does not consider PEFT or generative tasks. In contrast, our work studies PEFT as the adaptation regime, uses discrete uncertainty sets within a unified training-time robust optimization framework, and evaluates them across multiple PEFT methods in low-resource noisy settings, including generative tasks.

3 Background on robust optimization

Robust optimization (A. Ben-Tal and Ghaoui, 2009) is a field in optimization theory that aims to improve model stability under input uncertainty. Consider a model with parameters θ and a dataset $\mathcal{D}$ of input-label pairs (x, y), where $x \in \mathbb{R}^d$ and $y \in \{1, \ldots, K\}$. For each input x (in our context- as a sentence), we define an *uncertainty set* $\mathcal{S}_x \subseteq \mathbb{R}^d$: which reflects the level of uncertainty in the input- the set of sentences that arise

from small, semantics-preserving edits to x (e.g., token deletions, swaps, or character flips); these variants capture exactly the inputs on which the model's classification is uncertain. Given a loss function $\mathcal{L}$, the robust optimization objective is:

$$\min_{\theta} \mathbb{E}_{(x,y)\sim\mathcal{D}}\Big[\max_{x+\delta\in\mathcal{S}_x} \mathcal{L}(x+\delta;\theta,y)\Big].$$

When holding θ and y fixed and viewing $\mathcal{L}(x;\theta,y)$ as a function of x we occasionally write $\mathcal{L}_{\theta,y}(x)$. The inner maximization problem aims to find a worst-case example of a given x in the uncertainty set that achieves the highest loss. Using a first-order Taylor approximation around the input x, we can express the loss on the perturbed example as:

$$\mathcal{L}_{\theta,y}(x+\delta) \approx \mathcal{L}_{\theta,y}(x) + \langle \nabla\mathcal{L}_{\theta,y}(x), \delta\rangle. \quad (1)$$

The optimal perturbation δ^* for each training example is then:

$$\delta^* = \arg\max_{\delta: x+\delta\in S_x} \langle \nabla\mathcal{L}_{\theta,y}(x), \delta\rangle. \quad (2)$$

One way to define $\mathcal{S}_x$ is by a norm ball centered at x with a small radius ϵ:

$$\mathcal{S}_x = \{x+\delta\in\mathbb{R}^d : \|\delta\|_p \le \epsilon\}. \quad (3)$$

Equation (1) explains why the perturbation $\tilde{\delta}$[2] increases the loss compared to the original input. For small perturbation (with small ϵ) the perturbed x is positively correlated with the direction of the gradient $g = \nabla\mathcal{L}_{\theta,y}(x)$ (i.e., their angle is less than $90°$), making the inner product $\langle g, \tilde{\delta}\rangle$ positive and thus increasing the loss. This strategy yields more challenging training examples and can improve robustness to domain shifts and noisy inputs.

Adversarial Training (Goodfellow et al., 2015) stands as a prominent defense strategy for enhancing model robustness against attacks. It uses adversarial examples, which are perturbed versions of the original points. In the context of robust optimization, this approach leverages approximated worst-case examples which define an uncertainty set S_x around each input x, as detailed in Section 3.

[2]Details on how the choice of p shapes δ^* (e.g., ℓ_∞, ℓ_2, ℓ_1) and its connections to standard procedures like FGSM are deferred to Appendix B.

4 Methodology

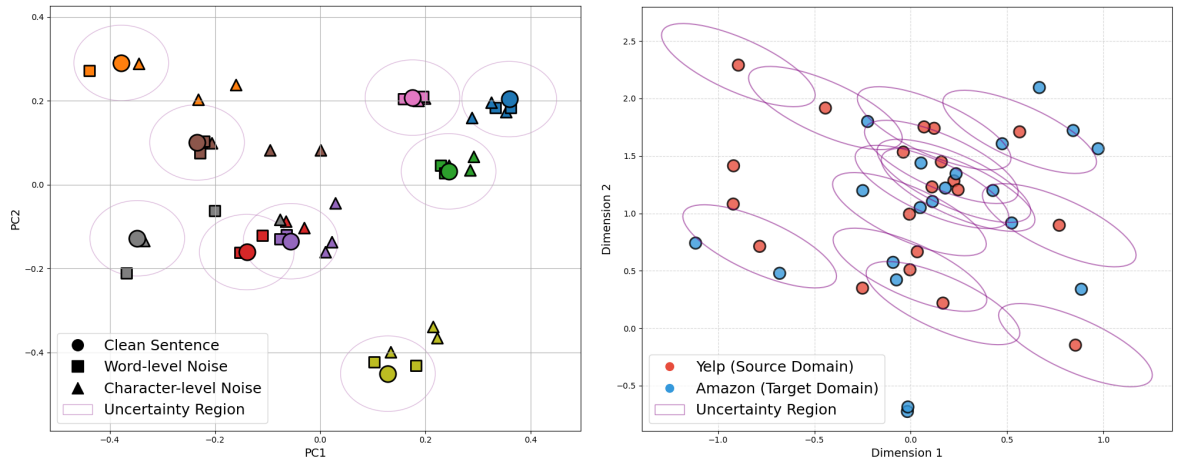
In this section, we describe how the proposed SDBN framework integrates adversarial training techniques with PEFT methods to address the challenges of noise robustness and domain shifts under limited data resources. While these techniques have been previously explored for full model fine-tuning, our focus is on demonstrating their particular value when applied to PEFT methods in low-resource, noisy settings. Conceptually, SDBN is a single robust-optimization framework instantiated with three uncertainty sets: (i) standard ℓ_∞ ball, (ii) discrete tokenization-breaking character edits (SDBN-h), and (iii) discrete LLM-generated adversarial variants (SDBN-p). We detail each variant in the following subsections.

4.1 Motivation

We address two practical robustness challenges encountered in real-world NLP deployments, particularly in the prevalent low-resource scenarios where limited training data makes these issues even more severe. The first concerns fine-grained textual perturbations that occur naturally in user-generated content but are often absent from clean training data. These include various character and word-level modifications that preserve semantic meaning while changing the visible text structure. The second involves domain shifts that occur when deployment environments differ subtly from training conditions in style or topic distribution. Unlike traditional domain adaptation scenarios where target domain data is available, we aim to build resilience against unanticipated shifts without prior exposure to the target domain. The data scarcity compounds these challenges, as models trained on small datasets tend to overfit and lack the broad exposure needed to generalize well to variations. Our objective is to enhance PEFT methods to withstand these everyday linguistic variations even when trained on limited data, maintaining high performance on clean inputs while creating more reliable models for practical applications.

4.2 Rationale

Motivated by (Shaham et al., 2018) and (Madry et al., 2019), who demonstrate that adversarial training functions as a robust optimization procedure, the integration of PEFT with robust optimization through adversarial training addresses challenges specific to low-resource settings. As illustrated



(a) Character/lexical edits (*TREC*)

(b) Domain Shift ($Yelp \nrightarrow Amazon$)

Figure 2: **Perturbation regimes addressed by Adversarial Training.** (a) PCA visualization of embedding space for TREC sentences, showing how character/word perturbations (triangles/squares) - fall within uncertainty regions (ellipses) around clean sentences (circles). This illustrates how our adversarial training approach effectively can improve the model's performance on diverse linguistic variations that occur within these uncertainty regions. Importantly, even when specific noisy examples are not encountered during training, the model develops robustness to these perturbations because they reside in the adversarially expanded uncertainty regions. (b) t-SNE plot of embeddings from Amazon and Yelp sentences, demonstrating a domain shift scenario where the uncertainty regions (ellipses) created around samples from the source domain (Yelp) encompass examples from the unseen target domain (Amazon). This visualization supports adversarial training key finding: by optimizing for worst-case examples within the uncertainty set during training, adversarial training generalizes effectively to novel domains without explicitly training on them.

in Figure 2, we can conceptualize both character/lexical variations and domain shifts as points within uncertainty regions surrounding clean examples. The PCA visualization in Figure 2(a) demonstrates how character and word-level perturbations fall within uncertainty regions surrounding their clean sentence counterparts. Similarly, Figure 2(b) shows how uncertainty regions created around samples from the source domain (Yelp) extend to cover examples from unseen target domain (Amazon).

Adversarial training optimizes robustness by training on *worst-case* perturbations within the uncertainty region. This forces stability and yields generalization to both noisy inputs and unseen domains without explicit exposure, allowing the model to handle linguistic variations beyond the training set.

Limited data scenarios. PEFT can outperform full fine-tuning on small datasets (Ben-Zaken et al., 2021; Pu et al., 2023), but limited corpora lack diversity in corruptions, dialects, and domain-specific terminology, making models *sensitive* to noise and domain shift. Adversarial training mitigates this by generating gradient-aligned *worst-case* neighbors around each example and optimizing performance *within the existing uncertainty set*. Training on these hard variants builds robustness to unseen linguistic variations and domain differences—without additional labeled data. This intuition is also consistent with our empirical findings in results 5, where adversarial training yields substantially larger gains for PEFT methods than for full fine-tuning in the same low-resource setting under noisy conditions.

Theoretical advantages of gradient-based perturbations over random noise. The effectiveness of adversarial training compared to random noise injection can be formally understood through the lens of high-dimensional geometry and its effect on the optimization landscape. While both approaches introduce perturbations during training, their impact on model robustness differs fundamentally.

As established in eq. (1), the change in loss when applying a perturbation δ to the input is approximated by $\langle \nabla \mathcal{L}_{\theta,y}(x), \delta \rangle$. In high-dimensional embedding spaces, random noise vectors (as used in methods like NEFTune and DAE (Vincent et al., 2008)) exhibit a critical limitation: they are approximately orthogonal to any fixed direction, including the gradient. This is a well-established property in high-dimensional spaces as explained by (Miyato et al., 2021), where random vectors become nearly perpendicular with high probability as dimension-

ality increases. Consequently, for random perturbations δ_{random} with constraint $\|\delta_{\text{random}}\|_p \leq \varepsilon$, the expected inner product $\mathbb{E}[\langle \nabla \mathcal{L}_{\theta,y}(x), \delta_{\text{random}} \rangle] \approx 0$. This results in minimal consistent effect on the loss, creating only weak, undirected regularization that fails to target the model's specific vulnerabilities.

In contrast, adversarial perturbations as defined in eq. (3) explicitly maximize the inner product with the gradient. The optimal perturbation $\delta^\star$ from eq. (1) ensures the largest possible first-order increase in loss within the constraint. This targeted approach creates worst-case examples that probe precisely the directions where the model is most sensitive, compelling optimization to flatten the loss landscape exactly where it is steepest. While random noise merely induces general smoothing across all directions, adversarial training systematically expands decision margins in the regions that most require robustness, producing models that generalize effectively to both natural perturbations and domain shifts as visualized in Figure 2.

4.3 SDBN: PEFT with Adversarial Training

In attempting to achieve robustness to noise using adversarial training, it is impossible to create adversarial examples by adding numeric perturbation to text symbols as in vision tasks (examining eq. (3), an incompatibility exists in the expression $x + \delta$, where x represents a sequence of discrete symbols while δ denotes a continuous numeric perturbation). Following established approaches in NLP adversarial training (Zhu et al., 2020),(Miyato et al., 2021),(Jiang et al., 2019), noise injection at the embedding layer is utilized rather than perturbing the raw input data. Let $E(\cdot)$ be the embedding extractor and $f(\cdot;\theta)$ be the subsequent layers of the integrated model with any PEFT method (e.g., LoRA) and θ denote the model's parameters. For an input batch (X, Y), we compute the embeddings $\mathbf{e} = E(X)$ and the clean loss: $\mathcal{L}_{\text{clean}} = \mathcal{L}\big(f_\theta(\mathbf{e}), Y\big)$. We then compute its gradient $\mathbf{g} = \nabla_{\mathbf{e}} \mathcal{L}_{\text{clean}}$ and form a perturbation: $\delta = \epsilon \cdot \text{sign}(\mathbf{g})$, yielding adversarial embeddings $\mathbf{e}_{\text{adv}} = \mathbf{e} + \delta$ and corresponding loss $\mathcal{L}_{\text{adv}} = \mathcal{L}\big(f_\theta(\mathbf{e}_{\text{adv}}), Y\big)$. Finally, we update the trainable parameters θ according to the specific PEFT method we use by $\mathcal{L}_{\text{adv}}$ (for implementation details, see section D.7).

Within the SDBN framework, this perturbation can be selected from any norm ball. Empirically, choosing from the ℓ_∞ norm ball yields the best performance and is used for the experiments in section 5. For a detailed comparison of perturbations drawn from ℓ_1, ℓ_2, and ℓ_∞ norm balls, see section C.6. For further empirical motivation and visual intuition showing why ℓ_∞ with $\epsilon = 10^{-4}$ best captures realistic noise patterns in embedding space, see the detailed analysis in section C.7. For the description of epsilon selection and the pseudo-code, see algorithm 1 and section C.1.

4.4 Character–level noise as a challenge

As fig. 2a indicates, most *word*-level edits (squares) stay within the uncertainty region around clean sentences (circles), whereas many *character*-level edits (triangles) fall outside it. Edits that *break a token* - e.g., deleting a letter -alter tokenization (splits/UNK) and push embeddings far from regions seen in training. By contrast, case changes typically preserve both tokenization (the word remains a single token) and semantics, so the resulting embedding stays close to the original. Appendix D.1 provides concrete examples of this distance gap. Hence, robustness to character-level noise is more challenging and may require complementary tools alongside embedding-space adversarial training.

Hybrid Strategy: SDBN-h. To address character-level perturbations that break tokenization and produce embeddings far outside the ℓ_p-ball used in SDBN, we define a discrete uncertainty set $\mathcal{S}_x$ as all single-character variants of a sentence x. Since $\mathcal{S}_x$ is finite, projected gradient descent cannot be applied directly. Instead, given the gradient $g = \nabla_e \mathcal{L}(f_\theta(e), y)$ from the clean embedding $e = E(x)$, we select z^* to be $\arg\max_{z \in \mathcal{S}_x} \langle g, E(z) - E(x) \rangle$ and use it as the adversarial example. Meaning, we use the gradient for selecting the perturbation but not for creating the perturbation. Each mini-batch is split: one subset is perturbed via standard ℓ_∞ FGSM in embedding space, and the other via $z^\star$, reusing the same gradient. This yields robustness to both continuous embedding perturbations and discrete character distortions with negligible overhead. For more details and full pseudo-code, see section C.2. SDBN-h extends embedding-space adversarial training to tokenization-discrete character corruptions by performing a discrete worst-case selection using the same gradient signal.

4.5 Prompt-Based Uncertainty Sets: SDBN-p

We have empirically found that the continuous embedding-space uncertainty set in SDBN is less effective for generative tasks. To tackle this,

we construct an alternative discrete uncertainty set by leveraging an LLM to generate semantic-preserving adversarial variants. Given a training example x, we prompt an LLM to generate k semantically-equivalent variants that include realistic perturbations such as paraphrases, typos, and style variations. Formally, we define $S_x^{\text{prompt}} = \{z_1, \ldots, z_k\}$, where each z_i is generated by prompting an LLM with x and instructions to produce adversarial variants.

Unlike SDBN and SDBN-h, where perturbations are small enough to be guided by clean-input gradients via Taylor approximation, the variants in S_x^{prompt} may involve significant structural changes that fall outside the local linear region. Consequently, for SDBN-p, we do not use the gradient-guided selection rule. Instead, we explicitly compute the loss for each of the k pre-computed variants and select the one that maximizes the training objective:

$$z^\star = \arg \max_{z \in S_x^{\text{prompt}}} \mathcal{L}(f(E(z); \theta), y) \qquad (4)$$

While this requires k forward passes, it ensures we identify the true worst-case semantic variant for robust optimization in generative tasks. This approach naturally captures linguistic variations that may be difficult to enumerate algorithmically, such as paraphrases and contextual rewrites, making it particularly suitable for generative tasks where output diversity is important. SDBN-p uses an LLM-generated discrete uncertainty set instead of a norm-ball uncertainty set. For more details, examples, and pseudo-code, see section C.3.

5 Results

Experimental Setup. We evaluate the proposed **SDBN** framework using two pre-trained models: BERT-base (Devlin et al., 2019) and DeBERTa-v3 (He et al., 2023) across multiple classification benchmarks: sentences datasets including 20NEWSGROUPS (Lang, 1995), BANKING77 (Casanueva et al., 2020), TREC (Singhal et al., 1999), and IMDB, as well as the word-pair semantic relation classification dataset BLESS (Baroni and Lenci, 2011). We additionally evaluate on generative tasks across SQUAD (Rajpurkar et al., 2016) and TWEETQA (Xiong et al., 2019) datasets, using LLaMA-3.2-1B (Llama Team, 2024), LLaMA-2-7B (Touvron et al., 2023) ,and Qwen-2.5-7B (Qwen Team, 2025) to assess robustness in generation-oriented tasks beyond classification. For SDBN-p, we use GPT-5.2 (OpenAI, 2025) to generate adversarial variants. For PEFT methods, we use: **Adapter**, **BitFit**, **LoRA**, and **QLoRA**. We use SDBN (ℓ_∞ uncertainty set) as the default, SDBN-h for tokenization-breaking character noise, and SDBN-p for generative tasks via precomputed LLM variant sets.

Our training protocol consists of 3 warm-up epochs with standard training followed by 10–20 epochs of either SDBN or baseline training without perturbations. We also compare to NEFTune (which was originally presented with QLoRA) and EDA (see implementation details in D.7)methods integrated with these PEFT approaches.

To simulate real-world scenarios, we evaluate under challenging conditions: **Data scarcity:** using 5% to 100% of the original training data. **Input noise:** applying word and char-level noise (for noise types details see section C.7) at test time.

The remainder of our experimental evaluation follows a systematic progression: We assess performance on clean test data across varying training set sizes to establish baseline effectiveness and robustness under word/character corruptions to demonstrate the advantages of the SDBN framework in noisy environments. We then examine cross-domain generalization using the ArSarcasm-v2 (Abu Farha et al., 2021) dataset and an NLI setting (cross-genre) to test performance under domain shifts. Due to space limits, all domain-shift results are reported in the Appendix (see D.2). Across these domain-shift evaluations, our adversarially trained PEFT approach (SDBN) maintains consistent gains over baselines. Finally, we conduct targeted analyses of the SDBN's components, comparing different perturbation strategies and examining the impact of perturbation location within the model architecture, and reporting resource costs such as runtime and memory footprint (see section D.5).

Results on clean data. We evaluate models trained with standard PEFT methods alongside their SDBN variants on clean test sets. Our experiments demonstrate that SDBN consistently improves accuracy over vanilla PEFT methods across all datasets with limited training data. This confirms our theoretical analysis from Section 4.2 that SDBN enhances generalization on clean data, not just under noisy conditions. Figure 3 illustrates the relative improvements on BANKING77 and TREC, revealing a critical insight: the benefits of SDBN become increasingly significant as training

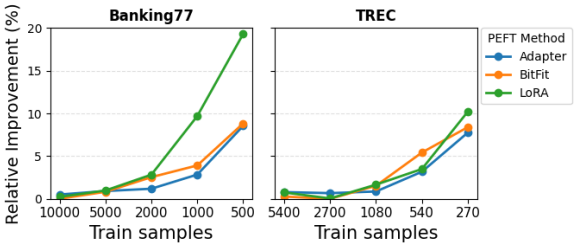


Figure 3: **Adversarial Training Effect on Limited Data Scenarios.** Relative performance improvement of adversarial training over baseline PEFT methods (e.g., 10% means SDBN-LoRA/LoRA = 1.10) across training sizes on BANKING77 and TREC. Adversarial training not only maintains but often improves clean-data performance, with gains increasing as the training set shrinks.

data size decreases. This validates our hypothesis that robust optimization techniques are particularly valuable in low-resource scenarios, where models typically struggle with generalization. Full results across all data scales and datasets are in D.1.

Results on Noisy Data. We evaluate model robustness to various linguistic perturbations using a 1,000-sample subset of BANKING77 with DeBERTa-v3, focusing on low-resource scenarios. As illustrated in Figure 4 for variable-intensity noise, where we vary the perturbation amplitude from 1-5 operations per sentence (e.g., deleting one word vs. five words), SDBN maintains consistent advantage over both vanilla PEFT and NEFTune. The performance gap remains stable as corruption severity increases, showing SDBN's superior ability to handle progressively more distorted inputs. Results for constant-intensity noise, where each sentence is corrupted by exactly one operation, show similar improvements and are provided in Appendix D.2.

These results validate our theoretical framework from section 4.2, where we represented linguistic variations as points within uncertainty regions surrounding clean examples. By optimizing for worst-case examples within these regions through gradient-based perturbations, SDBN effectively improves the model's robustness to real-world text variations, as visualized in fig. 2a. In contrast, NEFTune's random noise approach, while helpful, lacks the targeted nature of adversarial training. SDBN's systematic exploration of uncertainty regions enables more effective generalization to diverse perturbations without explicit exposure during training, making it particularly valuable for low-resource scenarios where the limited training data lacks natural linguistic diversity.

Table 1: **BLESS: accuracy (%) under character-level noise.** DeBERTa-v3 + LoRA trained on 1,000 clean samples. SDBN-h yields the best robustness on tokenization-breaking types.

(a) Clean, Delete-char, Swap-char

Method	Clean	Delete-char	Swap-char
Vanilla	89.81 ± 0.29	60.67 ± 2.19	56.82 ± 2.38
SDBN	**89.83 ± 0.24**	60.84 ± 1.85	57.22 ± 2.84
NEFTune	89.08 ± 0.54	61.19 ± 0.89	57.22 ± 0.82
SDBN-h	89.61 ± 0.30	**65.14 ± 1.19**	**62.80 ± 1.47**

(b) Double-char

Method	Double-char
Vanilla	68.51 ± 2.75
SDBN	68.66 ± 2.27
NEFTune	69.42 ± 1.29
SDBN-h	**72.54 ± 1.05**

Character Noise (SDBN-h). While SDBN improves robustness to a wide range of perturbations, its embedding-space ℓ_p-ball constraint cannot capture *tokenization-breaking* character edits, where a single change sends the resulting embedding far outside the continuous uncertainty region. To address this, SDBN-H augments standard FGSM perturbations with gradient-guided adversarial examples drawn from a discrete uncertainty set of single-character variants.

On BLESS with DeBERTa-v3 + LoRA trained on 1,000 clean samples, SDBN-h yields the best robustness on tokenization-breaking types (see table 1), improving by about +4–7% while matching clean accuracy. We include a qualitative robustness example illustrating tokenization-breaking character noise and SDBN-h behavior in Appendix D.6.

Generative Tasks. Our initial continuous-perturbation variant was less effective on generative tasks (Appendix D.4), motivating SDBN-p. We therefore evaluate on two generative benchmarks: SQuAD with LLaMA-3.2-1B and TweetQA with LLaMA-2-7B. Table 2 shows that SDBN-p improves performance over all baselines on both tasks, under both clean and noisy evaluation. Additional generative results and details are provided in Appendix D.4.

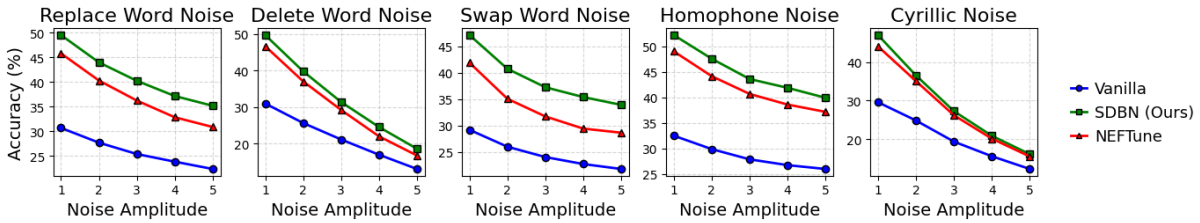


Figure 4: **Adversarial training for variable-intensity noise.** Performance comparison of LoRA PEFT implementations under variable-intensity noise with DeBERTa-v3 on 1,000 BANKING77 samples. SDBN shows superior robustness in low-resource settings.

Table 2: Generative task results under clean and noisy evaluation. **Top:** SQuAD exact match (EM) using LLaMA-3.2-1B with LoRA, trained on 200 samples. **Bottom:** TweetQA F1 using LLaMA-2-7B with LoRA, trained on 200 samples. SDBN-p uses generated adversarial variants during training. Best results in each column are shown in **bold**; second-best results are underlined.

SQuAD (EM), LLaMA-3.2-1B			
Method	Clean	Swap-Word	Homophone
Vanilla	58.92	32.44	47.28
NEFTune	59.72	32.72	48.08
EDA	58.88	32.44	47.96
FreeLB	57.00	31.64	44.04
SMART	52.64	28.44	40.52
SDBN-p	**59.84**	**35.08**	**52.20**
TweetQA (F1), LLaMA-2-7B			
Method	Clean	Delete-Char	Delete-Word
Vanilla	68.09	51.56	56.06
NEFTune	69.34	55.13	54.30
EDA	70.02	53.14	55.04
FreeLB	76.57	60.59	60.79
SMART	66.71	49.54	51.13
SDBN-p	**80.81**	**65.55**	**64.15**

PEFT vs. full fine-tuning under adversarial training. We find that adversarial training is more effective when combined with PEFT than with full fine-tuning in low-resource settings. Table 3 compares the gain of SDBN over the corresponding vanilla regime on a small subset of BANKING77 with DeBERTa-v3, evaluated on clean data and under three word-level corruptions. The gains are consistently substantial for PEFT methods, but negligible for full fine-tuning. This asymmetry suggests that PEFT's constrained parameter space may make the adversarial signal more focused and effective when data is scarce and noisy, whereas full fine-tuning is more vulnerable to overfitting adversarial examples in its much larger parameter number.

Table 3: **SDBN is substantially more effective with PEFT than with full fine-tuning.** The table reports the absolute accuracy gain (percentage points) achieved by adding SDBN to each training method on a low-resource subset of Banking77 (using DeBERTa-v3). For example, while SDBN improves LoRA by 23.6(%) on clean data, it only improves full fine-tuning by 1.3(%). This demonstrates that the adversarial signal is significantly more impactful within the constrained parameter subspaces of PEFT methods.

Method	**Clean**	**Replace**	**Delete**	**Swap**	**Avg gain**
LoRA	+23.6	+18.8	+18.7	+17.1	+19.6
BitFit	+16.0	+11.2	+12.8	+11.3	+12.8
Adapter	+13.3	+9.4	+9.8	+6.2	+9.7
Full FT	+1.3	+0.9	+0.5	+0.8	+0.9

6 Conclusion

We present SDBN (Small Data Big Noise), integrating adversarial training with PEFT to improve robustness in low-resource settings. Experiments show significant gains in robustness and accuracy across datasets and PEFT variants. We further extend the framework with two complementary strategies for discrete uncertainty set construction: SDBN-h, which incorporates gradient-guided character-level perturbations to address token-breaking typos, and SDBN-p, which leverages LLM-generated adversarial variants for generative tasks. The framework preserves PEFT's parameter efficiency while improving robustness to word-level and character-level noise and to domain shift, without requiring additional training data.

7 Limitations

While our method substantially improves robustness to both word-level and character-level perturbations, several limitations remain. Generating adversarial embeddings requires an additional forward–backward pass per mini-batch. Although this overhead does not increase GPU memory usage, it does raise per-batch runtime and may become a bottleneck when scaling to very large models or training corpora. Moreover, the choice of perturbation radius ϵ remains sensitive across datasets. While our analysis provides a principled default, more adaptive or automated tuning strategies could further enhance robustness and ease adoption.

References

Arkadi Nemirovski A. Ben-Tal and Laurent El Ghaoui. 2009. *Robust Optimization*, volume 2. Princeton University Press, Princeton, NJ.

Ibrahim Abu Farha, Wajdi Zaghouani, and Walid Magdy. 2021. Overview of the wanlp 2021 shared task on sarcasm and sentiment detection in arabic. In *Proceedings of the Sixth Arabic Natural Language Processing Workshop*.

Noëmi Aepli and Rico Sennrich. 2022. Improving zero-shot cross-lingual transfer between closely related languages by injecting character-level noise. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 4074–4083.

Marco Baroni and Alessandro Lenci. 2011. How we BLESSed distributional semantic evaluation. In *Proceedings of the GEMS 2011 Workshop on Geometrical Models of Natural Language Semantics, EMNLP 2011*, pages 1–10, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Elad Ben-Zaken, Shauli Ravfogel, and Yoav Goldberg. 2021. BitFit: Simple Parameter-efficient Fine-tuning for Transformer-based Masked Language-models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*. Association for Computational Linguistics.

Stephen Boyd and Lieven Vandenberghe. 2004. *Convex optimization*. Cambridge University Press.

Iñigo Casanueva, Tadas Temčinas, Daniela Gerz, Matthew Henderson, and Ivan Vulić. 2020. Efficient intent detection with dual sentence encoders.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. QLORA: Efficient Finetuning of Quantized LLMs. *arXiv preprint arXiv:2305.14314*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018. HotFlip: White-box adversarial examples for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Jiadong Fu, Jiang Fang, Jiyan Sun, Shangyuan Zhuang, Liru Geng, and Yinlong Liu. 2024. LoFT: LoRA-Based Efficient and Robust Fine-Tuning Framework for Adversarial Training. In *2024 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations (ICLR)*, San Diego, CA, USA.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2023. DeBERTaV3: Improving DeBERTa using electra-style pre-training with gradient-disentangled embedding sharing. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Neil Houlsby, Andrei Giurgiu, Stanisław Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-Efficient Transfer Learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97, pages 2790–2799. PMLR.

Edward Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. LoRA: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Po-Sen Huang, Robert Stanforth, Johannes Welbl, Chris Dyer, Dani Yogatama, Sven Gowal, Krishnamurthy Dvijotham, and Pushmeet Kohli. 2019. Achieving verified robustness to symbol substitutions via interval bound propagation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4083–4093.

Maor Ivgi and Jonathan Berant. 2021. Achieving model robustness through discrete adversarial training. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1529–1544.

Neel Jain, Ping-yeh Chiang, Yuxin Wen, John Kirchenbauer, Hong-Min Chu, Gowthami Somepalli, Brian R. Bartoldson, Bhavya Kailkhura, Avi Schwarzschild, Aniruddha Saha, Micah Goldblum, Jonas Geiping, and Tom Goldstein. 2023. Neftune: Noisy embeddings improve instruction finetuning. In *Preprint*.

Yuheng Ji, Yue Liu, Zhicheng Zhang, Zhao Zhang, Yuting Zhao, Gang Zhou, Xingwei Zhang, Xinwang Liu, and Xiaolong Zheng. 2024. AdvLoRA: Adversarial Low-Rank Adaptation of Vision-Language Models. *arXiv preprint arXiv:2404.13425*.

Robin Jia, Aditi Raghunathan, Kerem Gokhan Gülcehre, and Percy Liang. 2019. Certified robustness to adversarial word substitutions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4129–4142.

Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Tuo Zhao. 2019. SMART: Robust and efficient fine-tuning for pre-trained natural language models through principled regularized optimization. *arXiv preprint arXiv:1911.03437v5*.

Yeachan Kim, Junho Kim, and SangKeun Lee. 2024. Towards Robust and Generalized Parameter-Efficient Fine-Tuning for Noisy Label Learning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 5922–5936. Association for Computational Linguistics.

Ken Lang. 1995. Newsweeder: Learning to filter netnews. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 331–339.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Lam Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2021. P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. *arXiv preprint arXiv:2110.07602*. Version 3.

Llama Team. 2024. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2019. Towards deep learning models resistant to adversarial attacks.

Takeru Miyato, Andrew M Dai, and Ian Goodfellow. 2021. Adversarial Training Methods for Semi-Supervised Text Classification. *arXiv preprint arXiv:1605.07725v4*.

Ajinkya More. 2025. Investigating the robustness of parameter efficient fine tuning methods against adversarial attacks in natural language processing. Master's thesis, Purdue University, Fort Wayne, Indiana, August.

OpenAI. 2025. Update to GPT-5 system card: GPT-5.2. Technical report (PDF).

George Pu, Anirudh Jain, Jihan Yin, and Russell Kaplan. 2023. Empirical analysis of the strengths and weaknesses of peft techniques for llms. In *Proceedings of the Workshop on Understanding Foundation Models at ICLR*.

Qwen Team. 2025. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*.

Pengjie Ren, Chengshun Shi, Shiguang Wu, Mengqi Zhang, Zhaochun Ren, Maarten de Rijke, Zhumin Chen, and Jiahuan Pei. 2024. MELoRA: Mini-Ensemble Low-Rank Adapters for Parameter-Efficient Fine-Tuning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3052–3064, Bangkok, Thailand. Association for Computational Linguistics.

Barbara Rychalska, Dominika Basaj, Alicja Gosiewska, and Przemysław Biecek. 2019. Models in the wild: On corruption robustness of neural nlp systems. In *Neural Information Processing (ICONIP 2019)*, volume 11955 of *Lecture Notes in Computer Science*, pages 235–247. Springer.

Uri Shaham, Yutaro Yamada, and Sahand Negahban. 2018. Understanding adversarial training: Increasing local stability of supervised models through robust optimization. *Neurocomputing*, 307:195–204.

Amit Singhal, Steve Abney, Michiel Bacchiani, Michael Collins, Donald Hindle, and Fernando Pereira. 1999. AT&T at TREC-8. In *Proceedings of the Eighth Text REtrieval Conference (TREC-8)*. NIST.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, and 49 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning (ICML 2008)*, pages 1096–1103.

Jason Wei and Kai Zou. 2019. EDA: Easy Data Augmentation Techniques for Boosting Performance on Text Classification Tasks. *arXiv preprint arXiv:1901.11196*.

Wenhan Xiong, Jiawei Wu, Hong Wang, Vivek Kulkarni, Mo Yu, Shiyu Chang, Xiaoxiao Guo, and William Yang Wang. 2019. TWEETQA: A social

media focused question answering dataset. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5020–5031, Florence, Italy. Association for Computational Linguistics.

Jiehang Zeng, Jianhan Xu, Xiaoqing Zheng, and Xuanjing Huang. 2023. Certified robustness to text adversarial attacks by randomized [mask]. *Computational Linguistics*, 49(2):395–429.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. 2023. AdaLoRA: Adaptive Budget Allocation for Parameter-Efficient Fine-Tuning. In *International Conference on Learning Representations (ICLR)*.

Yi Zhou, Xiaoqing Zheng, Cho-Jui Hsieh, Kai-Wei Chang, and Xuanjing Huang. 2021. Defense against synonym substitution-based adversarial attacks via dirichlet neighborhood ensemble. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5482–5492.

Chen Zhu, Yu Cheng, Zhe Gan, Siqi Sun, Tom Goldstein, and Jingjing Liu. 2020. FREELB: Enhanced Adversarial Training for Natural Language Understanding. In *International Conference on Learning Representations (ICLR)*.

A Additional Preliminaries

A.1 Low-Rank Adaptation (LoRA)

LoRA (Hu et al., 2021) is a parameter-efficient fine-tuning method that adapts large-scale pre-trained models by learning low-rank updates. Consider a weight matrix $W_0 \in \mathbb{R}^{d\times k}$ in a pre-trained model. Instead of directly fine-tuning W_0, LoRA modifies it as:

$$W = W_0 + \Delta W,$$

$$\text{where } \Delta W = AB,$$

with $A \in \mathbb{R}^{d\times r}$ and $B \in \mathbb{R}^{r\times k}$, and $r \ll \min(d, k)$. This low-rank factorization significantly reduces the number of trainable parameters while retaining performance.

B Geometry of p-Norm Uncertainty Sets and Perturbation Behavior

Let $g = \nabla\mathcal{L}_{\theta,y}(x)$. Different choices of norm p determine distinct perturbation characteristics. The optimal perturbation δ^* can be approximated by a single steepest ascent step, yielding $\tilde{\delta}$ that maximizes the inner product in eq. (2). Steepest ascent determines the direction of maximal increase by optimizing the inner product with the function's gradient, subject to the given step size and the specific choice of norm, thereby defining $\tilde{\delta}$. Choosing $\tilde{\delta}$ from ℓ_∞ ball, generates perturbation where each entry of x is modified by the same amount in the direction determined by the sign of the gradient g, ($\tilde{\delta} = \epsilon \cdot \text{sign}(g)$) making it particularly suitable for attacks on vision models where imperceptible changes are crucial - every pixel is modified by a small ϵ. A practical adversarial training approach leveraging the ℓ_∞ uncertainty set is the Fast Gradient Sign Method (FGSM) (Goodfellow et al., 2015) which defines $\tilde{\delta}$ in a similar way. For ℓ_2 ball, the steepest ascent produces perturbations aligned with the direction of g, and choosing $\tilde{\delta}$ from ℓ_1 ball yields sparse perturbations where only a few entries corresponding to the components of g with largest absolute values are modified.

C Addition method

C.1 Epsilon Selection.

To maintain $\mathcal{L}_{\text{adv}}$ consistently higher than $\mathcal{L}_{\text{clean}}$, we empirically determine an appropriately small ϵ value (see section C.6). Our implementation includes an adaptive mechanism with a parameter K

Algorithm 1 SDBN – Small Data Big Noise: Adversarial Training with ℓ_∞

Require: Input batch X, labels Y, embedding extractor E, model $f(\cdot;\theta)$, initial ϵ

$\mathbf{e} \leftarrow E(X)$
$\mathcal{L}_{\text{clean}} \leftarrow \mathcal{L}(f_\theta(\mathbf{e}),\, Y)$
$\mathbf{g} \leftarrow \nabla_{\mathbf{e}}\, \mathcal{L}_{\text{clean}}$
$\delta \leftarrow \epsilon\ \text{sign}(\mathbf{g})$
$\mathbf{e}_{\text{adv}} \leftarrow \mathbf{e} + \delta$
$\mathcal{L}_{\text{adv}} \leftarrow \mathcal{L}(f_\theta(\mathbf{e}_{\text{adv}}),\, Y)$
Update θ using $\mathcal{L}_{\text{adv}}$

that adjusts ϵ: when $\mathcal{L}_{\text{adv}}$ falls below $\mathcal{L}_{\text{clean}}$, we conduct a line-search (Boyd and Vandenberghe, 2004) based search while iteratively reducing ϵ by a factor of 10 for up to K iterations until the constraint $\mathcal{L}_{\text{adv}} > \mathcal{L}_{\text{clean}}$ is satisfied. In practice, by choosing the appropriate ϵ, the constraint is satisfied using just one iteration (see section C.6).

C.2 SDBN-h

During training, each mini-batch is split into two parts: we apply SDBN (embedding-space ℓ_∞ perturbations) to the first part and SDBN-h (character-variant selection from $\mathcal{C}(x)$) to the remaining part (see Algorithm 2).

Algorithm 2 SDBN-h: Training Step

Require: Clean input x, label y, model $f(\cdot;\theta)$, loss $\mathcal{L}$, embedding $E(\cdot)$

Require: Noise-function set $\mathcal{H}$, where each $h \in \mathcal{H}$ maps $(x, i) \mapsto z$ by applying a character-level edit to x at index i

$g \leftarrow \nabla_{E(x)}\, \mathcal{L}(f(E(x);\theta),\, y)$ ▷ gradient of the clean example
Sample $h \sim \textsc{Uniform}(\mathcal{H})$
$\mathcal{C}(x) \leftarrow \{\, z_i = h(x, i)\ :\ i = 1, \ldots, |x| \,\}$ ▷ apply h at all character indices
$z^\star \leftarrow \arg\max_{z \in \mathcal{C}(x)} \langle g,\ E(z) - E(x) \,\rangle$
Update θ using $\mathcal{L}(f(E(z^\star);\theta),\, y)$

Character perturbation types. The discrete uncertainty set $\mathcal{C}(x)$ consists of single-character variants generated by the following operations:

- **Delete character:** Remove one character (e.g., "card" → "crd")
- **Swap characters:** Swap two adjacent characters (e.g., "card" → "acrd")
- **Double character:** Duplicate one character (e.g., "card" → "carrd")
- **Phonetic replacement:** Replace with phonetically similar character (e.g., "phone" → "fone")
- **Insert character:** Insert a random character (e.g., "card" → "ca1rd")
- **Cyrillic substitution:** Replace with visually similar Cyrillic character (e.g., "card" → "caяd")
- **Random capitalization:** Change case of one character (e.g., "card" → "caRd")

For each input x, one perturbation type is randomly selected, and all single-edit variants under that type form $\mathcal{S}(x)$.

C.3 SDBN-p

Algorithm 3 SDBN-p - Training Step

Require: Input x, label y, model $f(\cdot;\theta)$, loss $\mathcal{L}$, embedding $E(\cdot)$

Require: Pre-computed variants $\mathcal{P}(x) = \{z_1, \ldots, z_k\}$

$z^\star \leftarrow \arg\max_{z_i \in \mathcal{P}(x)} \mathcal{L}(f(E(z_i);\theta),\, y)$ ▷ select variant with maximum explicit loss
$\mathcal{L}_p \leftarrow \mathcal{L}(f(E(z^\star);\theta),\, y)$
Update θ using $\mathcal{L}_p$

LLM-generated adversarial variants. The discrete uncertainty set $\mathcal{P}(x)$ is pre-computed offline by prompting an LLM (e.g., GPT-5.2) to generate k adversarial variants for each training example. The variants introduce noise while preserving the semantic meaning and expected output, aiming to improve general robustness[cite: 1348]. This generation is performed once before training begins. During training, at each epoch, we explicitly compute the loss for each variant and select the one that maximizes the training objective. Because these variants can involve structural changes that fall outside the local linear region of the clean input, this explicit selection ensures we identify the true worst-case semantic neighbor. Importantly, since model parameters evolve during training, the worst-case variant may change from epoch to epoch, allowing the model to be exposed to different challenging examples throughout optimization.

Example prompt and output. (GPT-5.2)

> **Prompt to LLM:** Generate 5 adversarial variants of the following input. Each variant should

preserve the meaning and expected output, but challenge the model by increasing the loss size (cross-entropy), so that it will achieve robustness to noise.

Input: "Context: Because of its Catholic identity, a number of religious buildings stand on campus. [...] The Grotto of Our Lady of Lourdes, which was built in 1896, is a replica of the original in Lourdes, France. [...] Question: In what year was the Grotto of Our Lady of Lourdes at Notre Dame constructed? Answer:"

Expected output: "1896"

Generated variant (1 of 5): "Context: Because of its Catholic identity, a number of religious buildings stand on camtpus. The Old College building has become oneofMtwo seminaries [...] The Grotto of Our Lady of Lourdes, which was built in 1896, is a replica of the original in Lourdes, France. [...] Question: In what yea waIs the Grotto of Our Lady of Lourdes at Notre Dame constructed? Answer:"

The LLM introduces realistic character-level corruptions (e.g., "campus" $\rightarrow$ "camtpus", "year was" $\rightarrow$ "yea waIs") that break tokenization while preserving answerability.

C.4 Algorithm ℓ_2

Algorithm 4 Adversarial Training with ℓ_2

Require: Input batch X, labels Y, embedding extractor E, model $f(\cdot;\theta)$, initial ϵ

$e \leftarrow E(X)$
$\mathcal{L}_{\text{clean}} \leftarrow \mathcal{L}(f_\theta(e), Y)$
$g \leftarrow \nabla_e \mathcal{L}_{\text{clean}}$
$\delta \leftarrow \epsilon \cdot \frac{g}{\|g\|_2}$
$e_{\text{adv}} \leftarrow e + \delta$
$\mathcal{L}_{\text{adv}} \leftarrow \mathcal{L}(f_\theta(e_{\text{adv}}), Y)$
Update θ using $\mathcal{L}_{\text{adv}}$

C.5 Algorithm ℓ_1

Algorithm 5 Adversarial Training with ℓ_1

Require: Input batch X, labels Y, embedding extractor E, model $f(\cdot;\theta)$, initial ϵ

$e \leftarrow E(X)$
$\mathcal{L}_{\text{clean}} \leftarrow \mathcal{L}(f_\theta(e), Y)$
$g \leftarrow \nabla_e \mathcal{L}_{\text{clean}}$
$i^\star \leftarrow \arg\max_i |g_i|$
$\delta \leftarrow 0$ ▷ same shape as e
$\delta_{i^\star} \leftarrow \epsilon \cdot \text{sign}(g_{i^\star})$ ▷ add perturbation only at the max-magnitude entry
$e_{\text{adv}} \leftarrow e + \delta$
$\mathcal{L}_{\text{adv}} \leftarrow \mathcal{L}(f_\theta(e_{\text{adv}}), Y)$
Update θ using $\mathcal{L}_{\text{adv}}$

Table 4: Evaluation of different ℓ_p norms for varying ϵ values on *Banking77* (1,000 samples). Based on these results, we chose ℓ_∞ with $\epsilon = 10^{-4}$ as the best option.

ℓ_p / ϵ	10^1	10^0	10^{-1}	10^{-2}
ℓ_∞	2.26±0.71	2.75±0.95	2.80±1.30	4.41±3.09
ℓ_1	65.58±1.54	67.20±1.00	68.13±0.96	67.79±0.49
ℓ_2	9.38±11.16	70.24±2.23	70.18±0.85	68.32±1.57

(a) Larger ϵ values

ℓ_p / ϵ	10^{-3}	10^{-4}	10^{-5}
ℓ_∞	63.03±5.72	**72.86±1.24**	67.43±0.87
ℓ_1	67.09±1.60	67.03±1.09	67.62±1.14
ℓ_2	67.80±1.18	66.88±1.77	67.38±1.34

(b) Smaller ϵ values

C.6 Evaluate Epsilon and Norms

Table 5: **Perturbation Layer Impact on Model Performance.** Classification accuracy comparison when applying adversarial noise at different layers in BERT-base with LoRA, trained on 1000 samples from Banking77. Perturbations at the embedding layer dramatically outperform those at any encoder layer, highlighting that input-level modifications more effectively capture realistic linguistic variations.

Perturbed Layer	**Accuracy** (%)
Encoder 1	6.71 ± 2.18
Encoder 5	6.17 ± 1.73
Encoder 10	6.32 ± 1.95
Embeddings	**57.64 ± 5.04**

C.7 Justification for Using ℓ_∞ Norm with $\epsilon = 10^{-4}$

To justify our design choice, we analyzed how realistic textual noise affects sentence embeddings compared to clean data. Specifically, we took clean sentences and applied multiple noise types (e.g., word deletion, swap, replacement, case changes, character edits). For each noisy sentence, we computed the difference between its embedding vector and that of the corresponding clean sentence. We then visualized these embedding-level perturbations in two complementary ways: a histogram of coordinate-wise differences (Fig. 5) and a heatmap of difference magnitudes across embedding indices and noise types (Fig. 6).

The histogram in Fig. 5 shows that the perturbation values are distributed in a narrow range, approximately symmetric around zero, with no heavy tails. This indicates that noise does not create sparse, extreme deviations, but rather small,

bounded shifts across many embedding dimensions. Similarly, the heatmap in Fig. 6 demonstrates that all noise types induce perturbations of comparable magnitude, uniformly spread across embedding coordinates. Together, these observations suggest that realistic textual noise behaves like a low-magnitude, approximately uniform distribution across embedding dimensions.

These empirical findings motivate the use of the ℓ_∞ norm for adversarial perturbations, since the ℓ_∞ ball constrains each coordinate to be shifted by the same amount, reflecting the bounded, coordinate-wise nature of realistic noise. From our norm comparison study (Table 4), we found that ℓ_∞ with $\epsilon = 10^{-4}$ yielded the best trade-off: large enough to consistently increase the adversarial loss L_{adv} over the clean loss L_{clean}, yet small enough to preserve semantic proximity to the clean embeddings. Thus, this choice is both theoretically justified by the geometry of observed perturbations and empirically validated by robustness improvements.

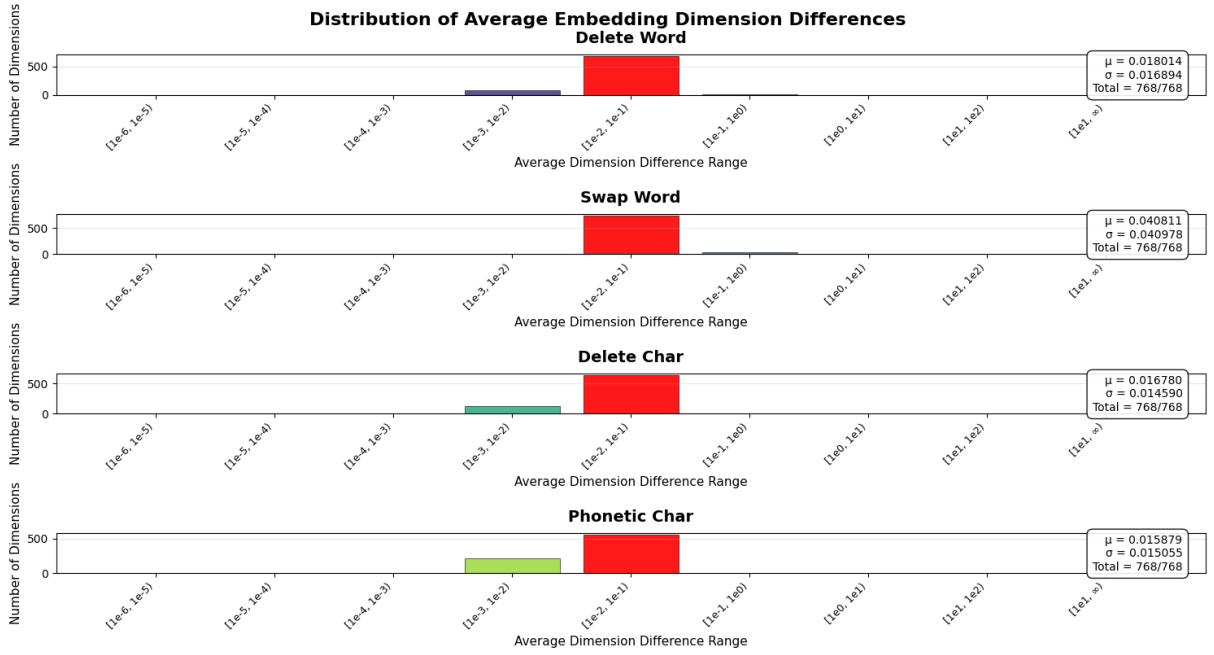


Figure 5: Histogram of embedding differences between noisy and clean sentences across multiple noise types. The values cluster in a narrow, symmetric range around zero, indicating that realistic noise introduces small but bounded shifts across embedding dimensions rather than sparse or extreme deviations.

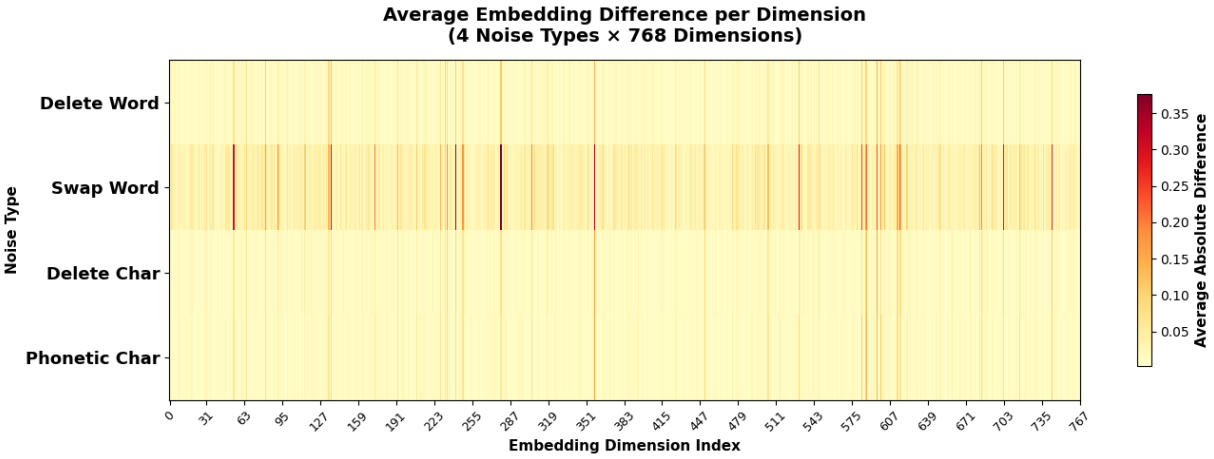


Figure 6: Heatmap of coordinate-wise embedding differences for multiple noise types. Each row is a noise type; each column is an embedding dimension. Similar magnitudes across rows indicate roughly uniform low-magnitude perturbations consistent with an ℓ_∞ ball.

Table 6: Types of noise perturbations used for robustness evaluation

Noise Type	**Description**	**Example**
Slang	Replace formal words with slang or informal abbreviations	"I don't know" → "idk"
Case	Change one character from lowercase to uppercase or vice versa	"my card" → "My card"
Pronoun	Replace nouns with pronouns	"my card" → "this"
Replace Word	Replace words with semantically similar alternatives	"purchase" → "buy"
Delete Word	Remove a word from the sentence	"I want my card" → "I want card"
Swap Words	Exchange the positions of two adjacent words	"my card" → "card my"
Homophone	Replace words with phonetically similar alternatives	"there" → "their"
Cyrillic	Replace Latin characters with visually similar Cyrillic characters	"like" → "li$\underline{\text{к}}$e"
Keyboard-char	Replace a character with a neighboring key on the keyboard (common typo)	"time" → "rime"

D Additional Results

D.1 Token-Breaking Examples

A *token-breaking* edit is any character-level change that forces the tokenizer to split what was a single token into several fragments. The table below shows three such edits.

String	Tokenizer output	#
reactivate	`_reactivate`	1
reactviate	`_react via te`	3
card	`_card`	1
crd	`_c rd`	2
would	`_would`	1
woudd	`_wo ud d`	3

- A single, well-trained token is replaced by several rare ones; their embeddings lie far from the original word in representation space.
- The longer sequence alters positional patterns, further distancing the whole sentence from its clean counterpart.
- Because these new embeddings sit *outside* the uncertainty region sampled during adversarial training, the defence becomes much less effective against such edits.

These observations explain the large gap between character-noise points (triangles) and word-level/clean points (circles, squares) in fig. 2a.

Table 7: **Adversarial Training Improves PEFT Across Resource Settings.** Performance comparison of standard PEFT methods vs their adversarial training counterparts (SDBN) on text classification tasks using BERT-base. Results show accuracy (%) when training on varying percentages (100%, 50%, 20%, 10%, 5%) of each dataset. Bold numbers indicate the better performing variant for each method pair. Adversarial training improves classification performance across all PEFT methods, datasets, and resource settings, with particularly significant gains in low-resource scenarios.

Data %	Method	Banking77	TREC	20NewsGroups	IMDB
100%	Adapter	92.82±0.40	91.16±0.39	68.67±0.52	87.71±0.33
	SDBN-Adapter	**93.29±0.10**	**91.88±0.56**	**69.62±0.49**	**88.96±0.20**
	BitFit	**90.67±0.45**	**90.20±0.55**	66.86±0.46	87.03±0.95
	SDBN-BitFit	90.66±0.34	89.96±0.81	**67.57±0.29**	**87.87±1.07**
	LoRA	91.56±0.18	91.44±1.04	68.84±0.27	88.30±0.13
	SDBN-LoRA	**91.77±0.51**	**92.12±0.16**	**69.60±0.25**	**88.89±0.26**
50%	Adapter	90.58±0.32	90.12±0.68	66.93±0.36	86.68±0.43
	SDBN-Adapter	**91.41±0.17**	**90.72±0.79**	**67.80±0.22**	**87.97±0.23**
	BitFit	87.67±0.40	88.08±0.48	65.46±0.40	86.84±0.25
	SDBN-BitFit	**88.39±0.30**	88.08±0.52	**65.89±0.41**	**87.46±0.11**
	LoRA	89.05±0.23	**90.20±1.15**	66.42±0.31	87.52±0.14
	SDBN-LoRA	**89.92±0.40**	90.16±0.67	**67.85±0.23**	**87.98±0.16**
20%	Adapter	86.18±0.69	83.68±1.26	64.64±0.45	85.52±0.42
	SDBN-Adapter	**87.20±0.76**	**84.40±0.99**	**65.33±0.45**	**86.70±0.15**
	BitFit	79.59±0.87	78.40±1.01	61.61±0.73	85.13±0.75
	SDBN-BitFit	**81.61±0.86**	**79.60±0.97**	**62.70±0.70**	**86.44±0.16**
	LoRA	82.47±0.75	81.88±1.83	63.17±0.50	86.21±0.26
	SDBN-LoRA	**84.81±0.75**	**83.24±1.72**	**64.28±0.50**	**86.56±0.23**
10%	Adapter	78.26±0.96	74.40±1.93	63.16±0.37	84.78±0.30
	SDBN-Adapter	**80.48±0.97**	**76.76±1.53**	**63.47±0.45**	**85.39±0.17**
	BitFit	67.45±1.31	65.68±1.83	56.78±1.38	84.01±0.35
	SDBN-BitFit	**70.08±0.71**	**69.24±0.80**	**58.92±0.92**	**84.56±0.95**
	LoRA	66.42±1.97	72.20±1.57	60.13±0.75	85.46±0.31
	SDBN-LoRA	**72.86±1.24**	**74.72±1.43**	**61.56±0.53**	**86.00±0.15**
5%	Adapter	58.69±1.72	59.32±4.29	59.74±1.04	84.08±0.37
	SDBN-Adapter	**63.70±1.50**	**63.92±3.85**	**61.07±1.00**	**84.89±0.43**
	BitFit	44.30±1.81	51.96±1.61	52.29±0.96	82.51±0.66
	SDBN-BitFit	**48.21±2.10**	**56.32±3.66**	**53.88±0.70**	**83.64±0.90**
	LoRA	36.29±3.89	56.04±5.03	55.86±2.06	84.63±0.25
	SDBN-LoRA	**43.30±2.57**	**61.76±1.64**	**58.27±0.93**	**85.22±0.22**

D.2 Performance over noisy test set

Constant-Intensity Noise Results. Figure 7 presents results for constant-intensity perturbations, where exactly one operation (word deletion, swap, replacement, etc.) is applied per sentence. SDBN consistently outperforms both vanilla PEFT and NEFTune across all noise types, with improvements ranging from 3-8% absolute accuracy. These single-operation perturbations represent the minimal corruptions commonly found in real-world text, demonstrating SDBN's effectiveness even for subtle linguistic variations.

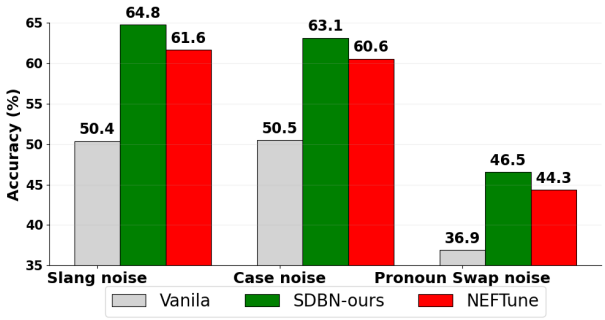


Figure 7: **Adversarial training for constant-intensity noise.** Performance comparison of BitFit PEFT implementations under constant-intensity noise conditions with DeBERTa-v3 on 1,000 samples from the *Banking77* dataset. These results demonstrate the superior robustness of gradient-based adversarial training in low-resource settings.

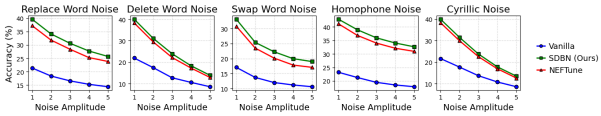


Figure 8: **Adversarial training for variable-intensity noise, Adapter PEFT.** Performance comparison of Adapter PEFT implementations under variable-intensity noise conditions in different amplitudes with DeBERTa-v3 on 1,000 samples from Banking77 dataset. These results demonstrate the superior robustness of Adversarial training gradient-based to noise in low-resource settings.

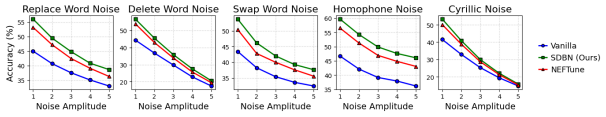


Figure 9: **Adversarial training for variable-intensity noise, BitFit PEFT.** Performance comparison of BitFit PEFT implementations under variable-intensity noise conditions in different amplitudes with DeBERTa-v3 on 1,000 samples from Banking77 dataset. These results demonstrate the superior robustness of Adversarial training gradient-based to noise in low-resource settings.

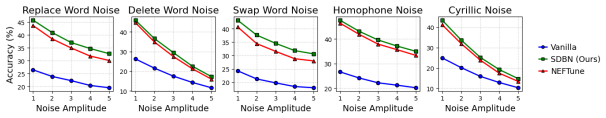


Figure 10: **Adversarial training for variable-intensity noise, QLoRA PEFT.** Performance comparison of QLoRA PEFT implementations under variable-intensity noise conditions in different amplitudes with DeBERTa-v3 on 1,000 samples from Banking77 dataset. These results demonstrate the superior robustness of Adversarial training gradient-based to noise in low-resource settings.

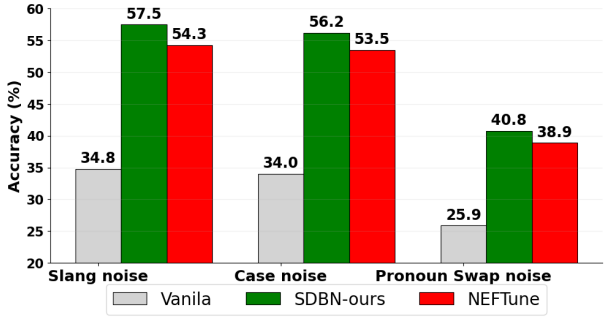


Figure 11: **Adversarial training for constant-intensity noise, LoRA PEFT.** Performance comparison of LoRA PEFT implementations under constant-intensity noise conditions with DeBERTa-v3 on 1,000 samples from Banking77 dataset. These results demonstrate the superior robustness of Adversarial training gradient-based to noise in low-resource settings.

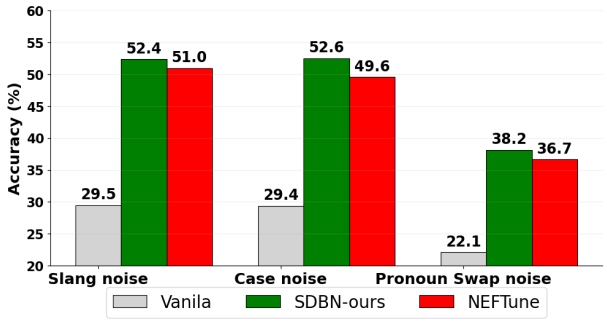


Figure 12: **Adversarial training for constant-intensity noise, QLoRA PEFT.** Performance comparison of QLoRA PEFT implementations under constant-intensity noise conditions with DeBERTa-v3 on 1,000 samples from Banking77 dataset. These results demonstrate the superior robustness of Adversarial training gradient-based to noise in low-resource settings.

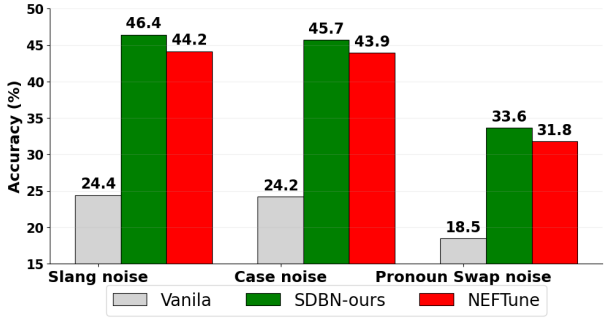


Figure 13: **Adversarial training for constant-intensity noise, Adapter PEFT.** Performance comparison of Adapter PEFT implementations under constant-intensity noise conditions with DeBERTa-v3 on 1,000 samples from Banking77 dataset. These results demonstrate the superior robustness of Adversarial training gradient-based to noise in low-resource settings.

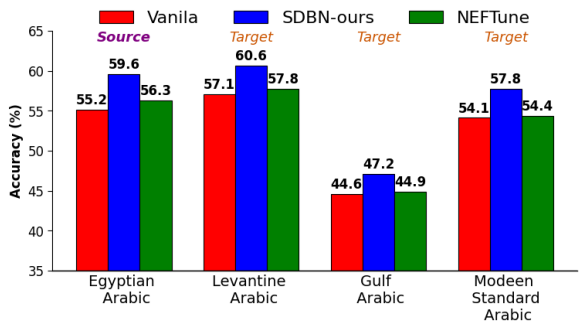


Figure 14: **Domain Shifts.** Comparison of LoRA PEFT methods on Arabic dialect sentiment classification using multilingual BERT-base trained on 270 samples of Egyptian Arabic without exposure to the other dialects during training. SDBN consistently outperforms alternatives across all dialects, with notable advantages on both the source and target domains.

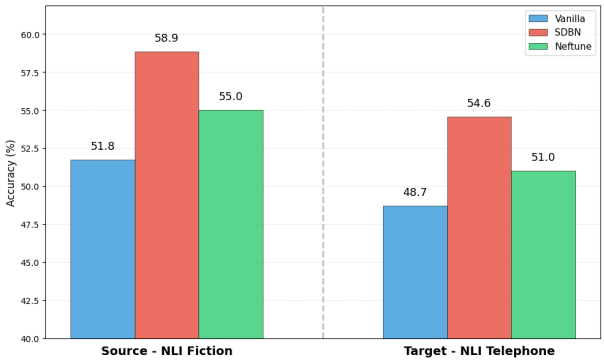


Figure 15: **NLI Classification Performance: Source vs Target Domains.** Performance comparison of LoRA PEFT methods on NLI classification tasks using DeBERTa-v3-large trained on only 77 samples from the source domain (NLI Fiction) without exposure to the target domain during training. Results show mean classification accuracy across 20 random seeds. SDBN consistently outperforms both vanilla LoRA and NEFTune across both domains, demonstrating superior robustness and generalization in this text classification scenario. While NEFTune shows improvement over vanilla LoRA through random noise injection, SDBN's gradient-based adversarial perturbations provide more robustness, particularly valuable in this extreme low-resource classification setting.

D.3 Domain Shifts.

Domain Shifts. In a *low-resource* setup- 270 training sentences from a single Arabic dialect and *no exposure to the target dialect*-we evaluate cross-domain robustness on the ArSarcasm-v2 benchmark. Despite this tiny-data constraint fig. 14, SDBN generalizes better to the unseen dialect than both vanilla and NEFTune.

This improvement accords with the rationale in section 4.2: gradient-based perturbations explore uncertainty regions around each source example that extend beyond the source dialect. The t-SNE visualization in fig. 2b shows these regions overlapping semantically related sentences from the unseen dialect, effectively "covering" the target domain without additional supervision.

Optimizing on these worst-case neighbors equips SDBN to handle domain shifts that would otherwise demand costly labeled data–an advantage that becomes particularly decisive when the available corpus is as small as a few hundred examples.

See section D.2 for further experiments and results on domain shifts.

D.4 Generative Tasks

Table 8: **SQuAD generative QA results (EM / F1).** Models trained on 500 clean SQuAD examples with DeBERTa-v3 and tested on clean data and noisy variants. Two noise types at test time: DELETE-CHAR and DELETE-WORD. Results are averaged over 5 runs. SDBN outperforms both vanilla PEFT and NEFTune.

	Clean	Del-Char	Del-Word
Vanilla	55.0/69.8	51.6/66.2	49.3/63.9
SDBN	**57.8/72.2**	**54.1/68.6**	**51.7/66.3**
NEFTune	55.0/69.6	51.5/66.1	48.9/63.5

Table 9: Additional generative results. TweetQA F1 using LLaMA-3.2-1B with LoRA, trained on 200 samples. Reported values are mean±std over random seeds. Best results in each column are shown in **bold**; second-best results are underlined.

Method	Clean	Keyboard-Char	Cyrillic
Vanilla	65.5±1.2	43.2±1.7	26.8±1.3
EDA	65.1±3.0	45.3±2.7	33.4±3.0
NEFTune	66.9±1.3	44.7±0.6	33.0±2.8
SMART	66.6±0.6	43.9±2.0	21.2±3.2
SDBN-p	**67.2±2.8**	**50.6±2.0**	**41.2±0.9**

Table 10: Additional generative results. SQuAD exact match (EM) using Qwen-2.5-7B with LoRA, trained on 200 samples. Reported values are mean±std over random seeds. Best results in each column are shown in **bold**; second-best results are underlined.

Method	Clean	Keyboard-Char	Double-Char
Vanilla	58.00±3.00	47.44±3.13	48.28±3.63
EDA	59.48±1.88	50.08±2.29	50.32±2.28
NEFTune	58.12±2.76	47.92±3.35	48.60±3.49
FreeLB	53.92±1.81	49.32±2.74	49.68±2.64
SDBN-p	**72.84±4.18**	**69.04±3.82**	**69.52±3.11**

For **SQuAD**, we adopt a compact adversarial-training schedule consisting of 1 warm-up epoch followed by 10 adversarial/method-specific epochs. For each training example, we construct 5 variants, and the same fixed set of variants is reused throughout training.

For **TweetQA**, we use a longer and more dynamic training schedule with 5 warm-up epochs followed by 16 adversarial/method-specific epochs. For each training example, we generate 10 variants; unlike SQuAD, these variants are regenerated at every adversarial epoch, so each epoch exposes the model to a new set of perturbations. This dynamic setup provides more diverse adversarial supervision and broader coverage of linguistic variation across training.

D.5 Method Analysis

We analyze the primary implementation choices that underlie the effectiveness of our approach. First, we compared perturbations constrained by the ℓ_1, ℓ_2, and ℓ_∞ norms across a range of magnitudes. The ℓ_∞ norm with $\epsilon = 10^{-4}$ produced the best performance over other norms. Full details of the norm comparison and ϵ tuning appear in section C.6.

Next, we explored *where* to inject the perturbations along the Transformer stack. When the noise was added to hidden activations in intermediate encoder layers, accuracy deteriorated sharply. Restricting the perturbation to the *embedding layer*, however, preserved the model's semantic processing and yielded the strongest robustness gains. Applying perturbations solely at the *embedding layer* keeps the noise interpretable as realistic textual edits–such as swapping two words, inserting an extra token, or deleting a character–thereby exposing the model to linguistic variation. Injecting noise deeper in the encoder, by contrast, corrupts high-level semantic representations that no valid sentence could produce, which explains the strong performance drop we observe. Detailed results for every injection point are reported in section C.6.

Table 11: **Runtime per batch (ms).** Mean wall-clock time (milliseconds) on an NVIDIA H200 for a full training step (forward–backward) on a batch of 32 *Banking77* sentences across PEFT methods. Lower is faster.

Approach	LoRA	QLoRA	BitFit	Adapter
Vanilla	116.21	148.96	92.68	95.79
SDBN	163.05	189.70	123.40	131.22
SDBN-h	170.71	206.64	140.48	150.25
SDBN-p	211.91	247.98	171.87	178.85

Runtime. Table 11 indicates that SDBN variants introduce a *bounded* and *predictable* compute overhead relative to vanilla PEFT. SDBN increases per-batch latency by about $1.40\times$, $1.27\times$, $1.33\times$, and $1.37\times$ for LoRA, QLoRA, BitFit, and Adapter, respectively (e.g., LoRA: $116.21 \rightarrow 163.05$ ms). The hybrid SDBN-h adds only a modest margin over SDBN—approximately 5–15% across methods (LoRA: $163.05 \rightarrow 170.71$ ms; QLoRA: $189.70 \rightarrow 206.64$ ms; BitFit: $123.40 \rightarrow 140.48$ ms; Adapter: $131.22 \rightarrow 150.25$ ms). SDBN-p incurs a larger but still bounded overhead—approximately 30–39%

over SDBN (LoRA: $163.05 \rightarrow 211.91$ ms; QLoRA: $189.70 \rightarrow 247.98$ ms; BitFit: $123.40 \rightarrow 171.87$ ms; Adapter: $131.22 \rightarrow 178.85$ ms). Overall, the added cost remains practical for deployment: the methods retain parameter efficiency and deliver the robustness gains reported earlier while keeping per-batch latency within practical bounds.

Method	Peak Memory (GiB)
Vanilla	68.60
SDBN	68.70
SDBN-h	68.70
SDBN-p	68.60

Table 12: Peak GPU memory usage during training on H200.

Memory Efficiency. Table 12 reports peak GPU memory during training. All SDBN variants introduce negligible memory overhead ($\leq$0.15%) compared to vanilla fine-tuning, confirming that our framework enhances robustness without additional memory cost. Notably, SDBN-p requires no extra memory since adversarial variants are precomputed offline.

D.6 A Qualitative Example.

Table 13 demonstrates that gradient-based adversarial training (SDBN) reliably improves robustness on word-level edits over Vanilla and NEFTune while preserving clean-data accuracy. Consistent with our rationale in section 4.2, optimizing against worst-case neighbors teaches the model to correctly handle semantically preserving rewrites that lie within the local uncertainty set. Importantly, the hybrid variant SDBN-h which augments SDBN with discrete character perturbations—substantially mitigates tokenization-breaking edits (character changes), closing the gap on character-level noise that defeats both baselines. Overall, these examples indicate that SDBN strengthens robustness around clean inputs, and SDBN-h extends this robustness to fine-grained character corruptions, yielding the most consistent behavior across the noisy variants in Table 13.

D.7 Setup Details

For all experiments, we utilized the Hugging Face `bert-base` model as our pre-trained backbone. The implementation of Parameter-Efficient Fine-Tuning (PEFT) methods was conducted using the PEFT library with PyTorch. We employed the AdamW optimizer with a learning rate of 1×10^{-4}, applied only to trainable parameters. The LoRA rank was set to 12 for BERT models and 4 for DeBERTa models across both the baseline and our adversarial training method. The adapter bottleneck dimension was set to 16 for both methods. Each experimental setting (dataset-method-dataset size) was run five times with five different random seeds to ensure robustness. During fine-tuning, we used: Batch size: 32 Dropout: 0.1 Evaluation was performed using the TextAttack library to assess model noise robustness. All experiments were conducted using an NVIDIA GPU. In section D.1 results show mean accuracy across 5 random seeds. In section D.2 results show mean accuracy across 10 random seeds without outliers (excluding the best and worst results for each method to reduce bias when working with small datasets).

EDA baseline. We use EDA as a lightweight data-noising baseline. To keep training compute comparable, we use conservative augmentation rates (α_{sr}=0.05, α_{ri}=0.05, α_{rs}=0.05, p_{rd}=0.05) and generate a single augmented example per input (`num_aug`=1). To match the training compute of other methods (i.e., the same number of optimizer steps), we do not expand the dataset; instead, in each mini-batch we replace the clean input with its EDA-augmented variant with probability 0.5, following the original EDA procedure. Note that EDA was originally proposed as a general data augmentation method and was not presented in combination with PEFT in its original work. The augmented dataset is generated once before training and reused across all epochs, following the original EDA offline augmentation procedure

Table 13: **A qualitative example.** A DeBERTa-v3 encoder was fine-tuned with LoRA on 1,000 *clean* BANKING77 training examples. We show one clean test example ("Original") and several noisy variants (word- and character-level). Cells report correctness (✓ = correct, × = incorrect). Adversarial training improves robustness overall (SDBN), and its hybrid variant (SDBN-h) is especially effective on character-level noise that disrupts tokenization.

Noise type	**Example**	**SDBN**	**SDBN-h**	**Vanilla**	**NEFTune**
Original	*How do I re-add a card to the app?*	✓	✓	✓	✓
Case char	***h**Ow **D**o **i** **Re**-**aDd** A **caRd** **tO** **tHe** app?*	×	✓	×	×
Replace word	*How do **me** **reinsert** a **ticket** to the **tool**?*	✓	✓	✓	×
Insert char	*H**1**o**2**w d**3**o I r**4**e-add a card to the app?*	×	✓	×	×
Delete word	*How do I re-add **...** ?*	✓	✓	×	✓
Delete char	*How do I re-**dd** a card **t** the app?*	✓	✓	×	×