STOCHASTICITY IN TOKENIZATION IMPROVES ROBUSTNESS

Anonymous authors
Paper under double-blind review

ABSTRACT

The widespread adoption of large language models (LLMs) has increased concerns about their robustness. Vulnerabilities to perturbations of the input tokenization indicate that models trained with a deterministic canonical tokenization can be brittle to adversarial attacks. Recent studies suggest that stochastic tokenization can deliver internal representations that are less sensitive to perturbations. In this paper, we analyse how stochastic tokenizations affect robustness to adversarial attacks and random perturbations. We systematically study this over a range of learning regimes (pre-training, supervised fine-tuning, and in-context learning), datasets, and model architectures. We show that pre-training and fine-tuning with uniformly sampled stochastic tokenizations improve robustness to random and adversarial perturbations. Evaluating on uniformly sampled non-canonical tokenizations reduces the accuracy of a canonically trained Llama-1b model by 29.8%. We find that training with stochastic tokenization preserves accuracy without increasing inference cost.

1 INTRODUCTION

In recent years, large language models (LLMs) have demonstrated remarkable capabilities across a wide range of natural language processing tasks, leading to their rapid and widespread adoption (Achiam et al., 2023; Team et al., 2023; Grattafiori et al., 2024; Brown et al., 2020). As their use has increased, so too have concerns about their safety and robustness (Li & Fung, 2025; Dong et al., 2024; Geh et al., 2025). Various approaches have been proposed to tackle these safety concerns, including fine-tuning (Wei et al., 2022; Agarwal et al., 2024), preference alignment (Rafailov et al., 2023; Song et al., 2024), and red teaming (Perez et al., 2022; Deng et al., 2023). However, studying their robustness to perturbations in the inputs, e.g., as adversarial attacks, remains underexplored.

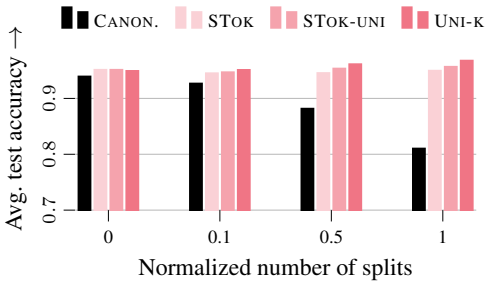


Figure 1: Effect of stochastic tokenization during testing. As the level of stochasticity increases, the accuracy of `Llama-1b` trained on LANGUAGE GAME with canonical tokenization (CANON) sharply drops, while models fine-tuned with stochastic tokenization remain robust.

Subword tokenization, a standard component of modern LLMs, has recently attracted attention in the study of LLM robustness. In LLMs, a tokenizer, such as Byte Pair Encoding (BPE) (Sennrich et al., 2016), encodes text into a sequence of tokens through a typically deterministic encoding function. The resulting token sequence serves as the model input, making it a natural point of attack. Although decoding is a many-to-one mapping, text is typically encoded using a deterministic function that returns the canonical tokenization. However, multiple token sequences can reconstruct the same string, referred to as non-canonical tokenizations. In standard training, the models are only exposed to the canonical tokenization. Recently, Zheng et al. (2025) and Kaplan et al. (2025) investigated the robustness of pre-trained LLMs against non-canonical tokenization and found that random non-canonical tokenization generally reduces performance substantially. Further, Geh et al. (2025) showed that alignment in LLMs can be broken by finding adversarial attacks in the space of tokenizations without changing the content of the query. These results indicate that LLMs are surprisingly brittle with respect to non-canonical tokenization, as also highlighted in Fig. 1.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

A potential remedy is to train with *stochastic tokenizations*, which expose the model to alternative segmentations of the same string while keeping the vocabulary fixed. BPE, for example, retains all intermediate merges, enabling multiple valid token sequences per string. BPE-dropout (Provilkov et al., 2020) introduces stochasticity by skipping merge rules at random. More recently, Sims et al. (2026) proposed STOCHASTOK, which randomly splits tokens into equivalent subtokens with some small probability. These works focus on improvements to subtoken-level understanding; consequently, the extent to which training under stochastic tokenizations affects robustness remains an open question.

In this work, we tackle the fundamental question: **Can stochastic tokenization improve robustness?** To this end, we investigate how stochastic tokenization during training affects robustness against random and adversarial tokenizations. We study various training regimes, including stochastic tokenization during pre-training, fine-tuning, and in-context learning (ICL). In our study, we use STOCHASTOK as the starting point, but find that STOCHASTOK introduces a biased sampling distribution over tokenizations. Therefore, we devise STOCHASTOK-UNI, which generates uniformly distributed tokenizations conditioned on a specific per-token split count, and UNIFORM-K, which generates uniformly distributed tokenizations with a specified number of total splits. Both can represent a larger set of non-canonical tokenizations compared to STOCHASTOK, *i.e.*, they have larger support. Our findings indicate that STOCHASTOK helps generalize across random tokenizations during testing, but for worst-case (adversarial) evaluations, we observe a significant drop in test performance compared to STOCHASTOK-UNI and UNIFORM-K. Lastly, we provide theoretical insights into the adversarial robustness introduced by stochastic tokenization during training.

Our contributions can be summarized as follows: *(i)* We study how training with stochastic tokenization affects robustness against random and adversarial tokenizations (Sec. 4). *(ii)* We analyse the sampling distribution of STOCHASTOK, finding that it is biased towards certain tokenizations and thus introduce unbiased uniform sampling schemes (Sec. 5). *(iii)* Lastly, we provide theoretical insights into adversarial robustness introduced by stochastic tokenization (Sec. 6).

2 BACKGROUND

We now introduce necessary notation and formally define relevant concepts on tokenization in LLMs.

Notation Let $\mathcal{X}$ denote an alphabet of characters x_i (or bytes), and let $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathcal{X}^*$ be a string, *i.e.*, a character sequence. The tokenization of $\boldsymbol{x}$ w.r.t. a token vocabulary $\mathcal{V}$ is a token sequence $\boldsymbol{v} = (v_1, \ldots, v_m) \in \mathcal{V}^*$ such that $v_1 \circ \cdots \circ v_m = \boldsymbol{x}$, where $\circ$ is the string concatenation. The vocabulary $\mathcal{V}$ is a finite set of string sequences $\mathcal{V} \subseteq \mathcal{X}^*$. We denote the encoding function $\tau \colon \mathcal{X}^* \to \mathcal{V}^*$ and the decoding function $\kappa \colon \mathcal{V}^* \to \mathcal{X}^*$, which satisfy $\kappa(\tau(\boldsymbol{x})) = \boldsymbol{x}$ for all $\boldsymbol{x} \in \mathcal{X}^*$.

Non-canonical tokenization Subword tokenization is the most prevalent family of tokenization methods in modern LLMs, mapping a sequence of characters to tokens. Which character sequences are merged is determined by the tokenization scheme. For instance, Byte Pair Encoding (BPE) (Gage, 1994) starts with a vocabulary that includes all individual characters. This vocabulary is then iteratively extended by merging the most frequent adjacent pairs of tokens in a given training corpus (Sennrich et al., 2016). The procedure stops once a predetermined vocabulary size is reached. However, this mapping is not unique. Given a token vocabulary $\mathcal{V}$, there may exist multiple token sequences that reconstruct the same underlying character sequence.

Example Consider the string $\boldsymbol{x} = (\texttt{revolution})$ and suppose the vocabulary $\mathcal{V} = \{\texttt{a}, \ldots, \texttt{z}, \texttt{re}, \texttt{v}, \texttt{ol}, \texttt{ution}, \texttt{revolution}\}$ contains all single characters as well as the listed merged character sequences. The canonical tokenization is the single-token decomposition $\boldsymbol{v}^c = (\texttt{revolution})$. Other possible *tokenizations* include $\boldsymbol{v} = (\texttt{re}, \texttt{v}, \texttt{ol}, \texttt{ution})$, and, at the extreme, the character-level tokenization $\boldsymbol{v}^{\text{char}} = (\texttt{r}, \texttt{e}, \texttt{v}, \texttt{o}, \texttt{l}, \texttt{u}, \texttt{t}, \texttt{i}, \texttt{o}, \texttt{n})$, all of which satisfy $v_1 \circ \cdots \circ v_m = \boldsymbol{x}$.

We denote the set of all *valid tokenizations* w.r.t. a vocabulary $\mathcal{V}$ as $\mathcal{T}_{\mathcal{V}}(\boldsymbol{x}) = \{\boldsymbol{v} : v_1 \circ \cdots \circ v_m = \boldsymbol{x}\}$, call $\boldsymbol{v}^c = \tau(\boldsymbol{x})$ the *canonical* tokenization, and refer to any $\boldsymbol{v} \in \mathcal{T}_{\mathcal{V}}(\boldsymbol{x}) \setminus \{\boldsymbol{v}^c\}$ as *non-canonical*.

STOCHASTOK Recently, Sims et al. (2026) introduced a stochastic tokenization scheme that is compatible with any tokenizer. It creates non-canonical tokenizations by randomly splitting tokens into pairs of smaller subtokens by following the merge rules reflected in the vocabulary.

Consider a string $\boldsymbol{x}$ and its canonical tokenization $\boldsymbol{v}^c = (v_1, \ldots, v_m)$. Given a desired number of splits N, STOCHASTOK starts from $\boldsymbol{v}^c$ and iteratively performs up to N split operations: at each step, a splittable token v_i from the current token sequence is selected uniformly at random and replaced with a valid pair of subtokens $(t_1, t_2) \in \mathcal{V}^2$. This procedure is repeated on the extended sequence until the split budget is exhausted or no splittable tokens remain, see Alg. 2.

Adversarial tokenization The goal of adversarial tokenization is to search for tokenizations of a string $\boldsymbol{x}$ that change an LLM's correct prediction, *i.e.*, break an alignment of an LLM. As the number of non-canonical tokenizations grows exponentially with the length of the string $|\boldsymbol{x}|$, an exhaustive enumeration is infeasible. Instead, Geh et al. (2025) proposed performing an adversarial search over local neighbourhoods in the tokenization space.

Let $d(\cdot, \cdot)$ be a token-level edit distance (Levenshtein distance (Levenshtein, 1965)) with insertion cost 1 and deletion cost 0. For a reference tokenization $\boldsymbol{v}$, define the set of tokenizations at edit distance i as $\mathcal{T}^i_{\mathcal{V}}(\boldsymbol{x}, \boldsymbol{v}) = \{\boldsymbol{u} : \boldsymbol{u} \in \mathcal{T}_{\mathcal{V}}(\boldsymbol{x}) \wedge d(\boldsymbol{v}, \boldsymbol{u}) = i\}$. The neighbourhood of $\boldsymbol{v}$ is defined as $\mathrm{Ne}(\boldsymbol{v}) = \mathcal{T}^2_{\mathcal{V}}(\boldsymbol{x}, \boldsymbol{v})$ whose cardinality is bounded by $\mathcal{O}(|\boldsymbol{v}|^2)$.

In this work, we focus on multiple-choice question answering tasks. Let $\boldsymbol{x}$ be a question string with tokenization $\boldsymbol{v}$, and $\{\boldsymbol{a}_c\}_{c=1}^C$ be a set of answer strings with tokenizations $\{\boldsymbol{v}_{\boldsymbol{a}_c}\}_{c=1}^C$. The model assigns a score, *i.e.*, $z_c(\boldsymbol{v}) = p(\boldsymbol{v}_{\boldsymbol{a}_c} \mid \boldsymbol{v})$, to each answer option. When attacking, our aim is to maximize the margin of violation

$$\boldsymbol{v}^\star = \underset{\boldsymbol{u} \in \mathcal{T}_{\mathcal{V}}(\boldsymbol{x})}{\arg\max} \left(\max_{c \neq y} z_c(\boldsymbol{u}) - z_y(\boldsymbol{u}) \right), \tag{1}$$

where y denotes the true class and c the most probable wrong class. To this end, Geh et al. (2025) proposed a greedy search over a local neighbourhood. Alg. 1 (Sec. F) summarizes the procedure used in this work, for the special case of radius-2 neighbourhoods, based on the greedy search algorithm. Adversarial tokenization assumes token-level access, which has relevance to open-weight models. The greedy attack additionally assumes access to model scores and represents a white-box setting.

3 RELATED WORK

Robustness to tokenizations Typically, LLMs are trained with deterministic subword tokenization (Sennrich et al., 2016; Kudo, 2018), which has been argued to obscure subtoken and character-level information (Chai et al., 2024; Edman et al., 2024). Recent work has demonstrated that models still encode within-word information without direct access to its characters (Itzhak & Levy, 2022; Hiraoka & Inui, 2025), with larger models performing better in general (Kaushal & Mahowald, 2022). Evidence suggests that models convert subword tokens into an implicit vocabulary, which allows models to retrieve character-level information (Feucht et al., 2024; Kaplan et al., 2025). In addition, Zheng et al. (2025) demonstrate the robustness of canonically trained LLMs to unseen non-canonical tokenizations at test time, especially after instruction-tuning. Post-training on curated datasets with canonical tokenization, however, has been argued to render models vulnerable to adversarial tokenization attacks (Geh et al., 2025).

Training with stochastic tokenizations Several works have proposed training with stochastic tokenizations to improve subword understanding. Among the first, Provilkov et al. (2020) introduce stochasticity by randomly dropping merge rules. Cognetta et al. (2024) show that BPE-dropout (Provilkov et al., 2020) and MaxMatch-dropout (Hiraoka, 2022) result in a non-uniform distribution over tokenizations, and show improvements in machine translation quality with uniform sampling. Bauwens et al. (2025) argue that uniform sampling leads to tokens with reduced character count and propose length-aware sampling. Cosma et al. (2026) sample non-canonical tokenizations, and use an auxiliary encoder to condition next-token prediction on them. Sims et al. (2026) randomly split tokens into pairs of subtokens, showing improved subword understanding and suggesting improved robustness to tokenizations.

A more detailed discussion of related work is provided in Appendix A.

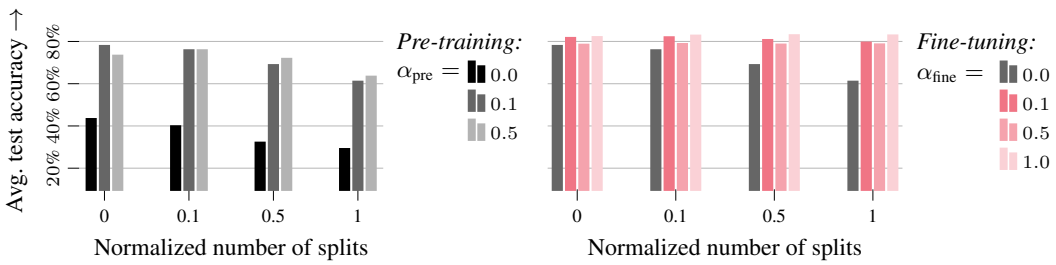


Figure 2: *(Left)* Average test accuracy of **Tiny-LLM** pre-trained either canonically or with StochasTok, and subsequently fine-tuned on Language Game using canonical tokenization. Stochastic tokenization in pre-training performs favourably in all settings and provides mild robustness to input perturbations (normalized number of splits > 0). *(Right)* Additionally fine-tuning with StochasTok tokenization ($\alpha_{\text{fine}} > 0$) after pre-training with $\alpha_{\text{pre}} = 0.1$ substantially improves robustness.

4 Does StochasTok improve robustness?

In the following section, we investigate whether training with StochasTok improves robustness to non-canonical tokenizations. Our aim is to provide a practical overview of the benefits of stochastic tokenization in different training regimes. For our evaluation setup, we use the Language Game and Cute datasets from Sims et al. (2026). These datasets specifically test subtoken-level understanding, such as counting or detecting letters within words (see Sec. E for examples) and provide a targeted stress test for tokenization robustness. We consider (i) pre-training, (ii) supervised fine-tuning, and (iii) in-context learning. In particular, we evaluate the performance on canonical tokenization during testing (clean case) and on stochastic tokenization at various levels, to assess the model's ability to solve the task in the ideal setting and its robustness to input perturbations.

4.1 Does pre-training improve robustness?

First, we test whether StochasTok pre-training improves subtoken understanding *and* robustness. Due to the computational cost of pre-training, we train **Tiny-LLM** (50 M-parameter) (Hillier et al., 2024) on OpenWebText. For larger models, we focus on fine-tuning and in-context learning. Below, we follow Sims et al. (2026) and perform canonical fine-tuning on Language Game to test whether subtoken understanding and robustness to alternative tokenizations persist. In particular, we evaluate average accuracy for StochasTok tokenizations at normalized split rates $\{0, 0.1, 0.5, 1\}$ (see Eq. (3), Sec. D.1). Zero corresponds to canonical, deterministic tokenization. Additional pre-training evaluations are provided in Sec. I.1. All test-time stochastic tokenizations in this experiment are sampled with StochasTok and reflect its sampling biases. In Sec. 5, we test the robustness to uniformly sampled tokenizations.

In Fig. 2 *(left)*, canonical pre-training ($\alpha_{\text{pre}} = 0.0$) is insufficient for solving these subtoken-level tasks. Pre-training with stochastic tokenization ($\alpha_{\text{pre}} > 0.0$) substantially improves accuracy, consistent with Sims et al. (2026). Beyond this previously observed improvement, we test whether it translates to robustness under increasingly non-canonical tokenizations. Across all pre-training settings, we observe that accuracy degrades as the normalized number of splits increases. However, accuracy with stochastic pre-training remains substantially higher compared to canonical pre-training. Results for Cute exhibit the same trend and are shown in Fig. 11, Sec. I.2. We note that favourable performance in the clean case does not necessarily provide strong robustness to alternative tokenizations.

4.2 Does fine-tuning improve robustness?

As pre-training with StochasTok alone provides mild robustness against noisy inputs, we now address two questions: (i) does fine-tuning with StochasTok after pre-training with StochasTok instil additional robustness, and if so (ii) is fine-tuning with StochasTok sufficient to provide robustness against noisy inputs? Previously, Geh et al. (2025) found that large models, pre-trained on massive, possibly noisy, text corpora, exhibit a certain robustness to non-canonical tokenizations.

However, fine-tuning on smaller, curated datasets was argued to introduce vulnerabilities. Hence, question (ii) examines if fine-tuning with STOCHASTOK prevents those vulnerabilities.

To address the first question, we compare canonical fine-tuning against fine-tuning with STOCHASTOK using varying expected split rates $\alpha_{\text{fine}} \in \{0.1, 0.5, 1\}$. In all cases, we start with stochastic pre-training using $\alpha_{\text{pre}} = 0.1$. In Fig. 2 (right), we observe that additional stochastic fine-tuning substantially improves robustness against input perturbations and does not degrade accuracy on canonical tokenizations. In Fig. 10a (Sec. I.2), we provide ablations on canonical pre-training.

Next, we test whether this effect translates to larger canonically pre-trained models. For this, we use a publicly available model checkpoint of `Llama-1b` and fine-tune a low-rank adapter (LoRA) (Hu et al., 2022). Fig. 3 demonstrates that, while the `Llama-1b` model is able to solve the subword-level tasks without stochastic pre-training, it is brittle to non-canonical tokenizations during evaluation. Fine-tuning with stochastic tokenization and a moderate expected split rate ($\alpha_{\text{fine}} = 0.1$) significantly increases robustness while preserving canonical performance. Results for `GPT2-xl` exhibit the same trend and are shown in Fig. 14, Sec. I.2.2. This suggests that stochastic tokenization during fine-tuning is an effective way to improve robustness to non-canonical tokenization, even if the base model was pre-trained canonically, and effectively reduces potential vulnerabilities to non-canonical tokenization.

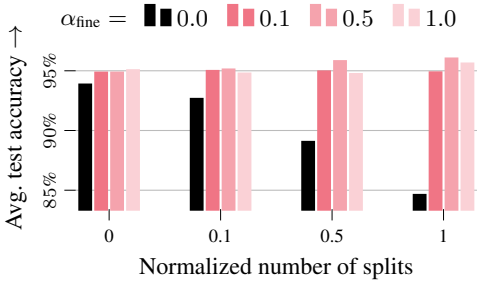


Figure 3: Average test accuracy for canonically pre-trained `Llama-1b` on LANGUAGE GAME with stochastic fine-tuning. Stochasticity improves robustness to random perturbations without hurting canonical performance.

4.3 DOES IN-CONTEXT LEARNING IMPROVE ROBUSTNESS?

Lastly, we study the most restrictive scenario, where the model parameters are kept frozen. We test whether robustness to stochastic tokenizations can be improved *without parameter updates*, using in-context learning (ICL) (Yin et al., 2024). In ICL, the prediction for a query $\boldsymbol{x}^*$ is obtained by prepending solved examples as context $(\boldsymbol{x}_1, \boldsymbol{r}_1, \ldots, \boldsymbol{x}_K, \boldsymbol{r}_K, \boldsymbol{x}^*)$, while keeping model parameters fixed. Specifically, we analyse whether stochastic tokenization of the context questions, $\boldsymbol{v}_i \sim P(\cdot \mid \boldsymbol{x}_i; \alpha_{\text{ICL}})$, increases robustness to stochastic tokenizations of the query, $\boldsymbol{v}^* \sim P(\cdot \mid \boldsymbol{x}^*; \alpha_{\text{eval}})$.

We evaluate robustness to non-canonical tokenizations of the query for LANGUAGE GAME and general multiple-choice datasets. Table 1 reports results for a `Llama-8b` model using $K = 10$ context examples. We compare accuracy for canonical query tokenizations against non-canonical query tokenizations at $\alpha_{\text{eval}} = 1$. Using a canonical context *(first row)* consistently yields lower accuracy under non-canonical queries than under canonical queries. Using STOCHASTOK for context tokenization yields notable robustness gains on LANGUAGE GAME, with modest improvements on the other benchmark datasets. While ICL imposes the lowest computational cost, it also yields lower robustness gains than full fine-tuning. In fine-tuning, the model is trained with 10k training samples, each tokenized with 10 random draws by STOCHASTOK. In contrast, in ICL the model is presented with $K = 10$ examples, each with a single STOCHASTOK sample.

Having established that STOCHASTOK does improve robustness to non-canonical tokenizations, we investigate whether further gains are possible. We explore the limitations of STOCHASTOK's distribution over tokenizations and propose two alternative sampling schemes (STOCHASTOK-UNI and UNIFORM-K), considering both randomly sampled and adversarially selected tokenizations.

5 IMPROVING WITH UNIFORM SAMPLING

While training with STOCHASTOK generally improves robustness, it comes with two key limitations. Firstly, the sampling distribution $P_{\text{STOK}}(\boldsymbol{v} \mid \boldsymbol{x}; \alpha)$ generated by STOCHASTOK is non-uniform. Among segmentations with the same number of splits, it is biased towards those constructed from subtokens with fewer alternative split pairs. Secondly, the sampling process yields a distribution with incomplete support. We discuss these two limitations in detail below.

Table 1: Accuracy (%) under canonical versus StochasTok ($\alpha_{\text{eval}} = 1$) query tokenization across datasets, $K = 10$ few-shot examples.

Context tokenization	α_{ICL}	Language Game		ARC-E		COPA		CSQA		HellaSwag		PIQA		SocialIQA	
		canon.	StoK	canon.	StoK	canon.	StoK	canon.	StoK	canon.	StoK	canon.	StoK	canon.	StoK
Canonical	-	83.8	74.5	78.2	76.0	92.0	91.0	68.0	66.5	54.8	52.5	81.4	81.5	52.4	50.4
StochasTok	0.1	84.8	78.5	77.0	76.3	92.0	90.9	68.4	66.4	55.2	52.8	81.8	81.5	52.6	51.8
StochasTok	0.5	86.8	82.5	77.6	76.8	92.0	91.3	67.4	66.4	54.2	52.2	82.2	81.4	54.0	51.9
StochasTok	1.0	86.4	83.0	77.6	77.2	91.0	91.0	67.4	66.0	54.2	52.0	82.8	81.6	53.0	52.2

Bias First, let us call a subtoken pair $(v, v') \in \mathcal{V}^2$ *valid* if, when merged via a two-way merge $v \circ v' = u$, they form a token $u \in \mathcal{V}$. Now, let us denote the set of valid subtoken pairs as: $\mathcal{S}(v_i) = \{(t, t') \in \mathcal{V}^2 : t \circ t' = v_i\}$.

Recall, StochasTok recursively splits tokens v_i into (valid) token pairs (t, t'). Let us consider tokenizations obtained by performing exactly one further split. At this point, StochasTok selects the token to be split uniformly from $\{t, t'\}$. Importantly, the two subtokens may have an unequal number of valid subtoken pairs, $|\mathcal{S}(t)| \neq |\mathcal{S}(t')|$. As the subsequent split pair is sampled uniformly from $\mathcal{S}(\cdot)$, each individual tokenization is obtained via splitting t or t' with probability $1/|\mathcal{S}(t)|$ or $1/|\mathcal{S}(t')|$, respectively. Thus, the induced distribution over the tokenizations is inherently biased.

Incomplete support Let's assume we have the canonical token `revolution`. Then, a potential non-canonical tokenization with two splits performed by StochasTok is shown in Fig. 4 *(left)*. In this example, all resulting token pairs are part of the vocabulary.

Another potential tokenization is obtained by splitting the initial token directly into three subtokens, shown in Fig. 4 *(right)*, as all three subtokens, `re`, `vol`, and `ution` are contained in the vocabulary. However, neither `revol` nor `volution` is a token in the vocabulary $\mathcal{V}$. Thus, StochasTok cannot generate the tokenization through a series of two-way splits. This example illustrates the second key issue: StochasTok cannot generate all possible k-edit distance tokenizations of a token. Fig. 5 and Fig. 8 (Sec. H.2) confirm the resulting sampling bias empirically.



(a) Example two-way splits by StochasTok.

(b) Example of an unreachable three-way split.

Figure 4: StochasTok can split a token into subtokens (a) if the resulting pairs are in $\mathcal{V}$. However, there exist segmentations (b) where all subtokens are in $\mathcal{V}$, but the required pairwise splits are not.

5.1 Removing bias and increasing support

To address these limitations and understand their influence on robustness to attacks and perturbations, we propose stochastic tokenization schemes with progressively increasing uniformity and support.

StochasTok-uni As a first step, we introduce a uniform extension of StochasTok that reduces sampling bias while preserving the per-token split-count distribution induced by StochasTok, which we call StochasTok-uni. For this, we devise a two-stage procedure where (i) we generate the per-token split-count vector $\boldsymbol{S} = (S_1, \ldots, S_m)$ for a sequence of m tokens following the distribution induced by StochasTok, and (ii) for each token independently, we uniformly sample non-canonical tokenizations with S_i splits, that is, with $S_i + 1$ segments.

First, note that StochasTok introduces a split-count distribution for a canonical token following a Beta-Binomial distribution, *cf.* Sec. H.1. By doing so, StochasTok has a preference towards small split counts. To sample $\boldsymbol{S} = (S_1, \ldots, S_m)$ with $\sum_i S_i = N$ we use a Dirichlet-multinomial model, $\boldsymbol{S} \sim \text{DirMult}(N, \mathbf{1})$, with a symmetric concentration parameter.

For the second step, let $\boldsymbol{v}^c = (v_1^c, \ldots, v_m^c)$ denote the canonical token sequence. Recall that each token $v_i^c \in \mathcal{V}$ corresponds to a character sequence which we denote by $\tilde{\boldsymbol{x}}_i \coloneqq \kappa(v_i^c) \in \mathcal{X}^*$. Then, conditional on the split counts $\boldsymbol{S}$, we sample tokenizations uniformly from $\mathcal{T}_{\mathcal{V}}^{S_i+1}(\tilde{\boldsymbol{x}}_i, v_i^c)$, that is, from the set of tokenizations consisting of $S_i + 1$ segments.

Inspired by Zheng et al. (2025), we recursively construct a tree, where each root-to-leaf path corresponds to a non-canonical tokenization $\boldsymbol{v} \in \mathcal{T}_{\mathcal{V}}(\tilde{\boldsymbol{x}}_i)$. To ensure that the path length equals the number of segments $S_i + 1$, we remove all paths longer or shorter than $S_i + 1$. Next, we perform uniform sampling by descending the tree, selecting each outgoing edge with probability proportional to the number of leaves in its subtree. The sampling cost scales linearly with the number of segments, while constructing the tree scales exponentially with the sequence length of the token, $\mathcal{O}\left(2^{|\tilde{\boldsymbol{x}}|-1}\right)$. In our case, the cost remains manageable as we only construct the tree for each token v_i^c independently.

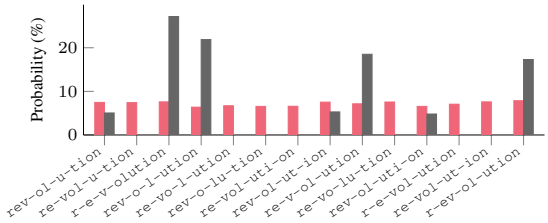


Figure 5: Histogram of segmentation probabilities. StochasTok ▮▮ induces a biased distribution over a subset of valid segmentations, whereas StochasTok-uni ▮▮ samples segmentations approximately uniformly and has full support.

Fig. 5 compares the empirical distribution over segmentation probabilities for StochasTok and StochasTok-uni. Note that StochasTok covers only a subset of all non-canonical tokenizations, while StochasTok-uni (i) provides full support and (ii) results in a uniform distribution over tokenizations of a given split number.

Uniform-k Note that, while StochasTok-uni removes the key limitations of StochasTok, it remains restricted to per-token level splits. Also, the tokenizations are sampled uniformly *conditioned* on the per-token split count. As the per-token split count is created to match StochasTok, it is biased towards lower split numbers per token. Hence, we introduce Uniform-k, which samples *marginally* uniformly among non-canonical tokenizations with k added tokens over the *entire input sequence*, *i.e.*, $\boldsymbol{v} \sim \mathrm{Unif}(\{\boldsymbol{v} \in \mathcal{T}_{\mathcal{V}}(\boldsymbol{x}) : |\boldsymbol{v}| = |\boldsymbol{v}^c| + k,\ \boldsymbol{v} \neq \boldsymbol{v}^c\})$. Creating the tokenization tree recursively for the entire string $\boldsymbol{x}$ is generally infeasible due to exponential scaling w.r.t. the length $|\boldsymbol{x}|$. To address this, we encode the tokenizations with a Multi-valued Decision Diagram (MDD) (Geh et al., 2024). By decomposing and reusing subsequences, the MDD encodes all tokenizations with a construction cost scaling linearly w.r.t. the tokenizer vocabulary and the sequence length, $\mathcal{O}(|\boldsymbol{x}| \cdot |\mathcal{V}|)$. Sampling is performed by following edges with probability proportional to the number of tokenizations of the sub-MDD. To sample uniformly at a fixed number of splits k, we use the multi-rooted MDD (MRMDD) of Geh et al. (2025), whose construction cost is upper bounded by $\mathcal{O}(|\boldsymbol{x}|^2 \cdot |\mathcal{V}|)$. We adopt their MRMDD algorithm, but condition on sampling the desired output length $|\boldsymbol{v}^c| + k$ to match the total split count N of StochasTok and StochasTok-uni. Note, however, that uniform sampling induces a different per-token split-count distribution (see Sec. H). By construction, Uniform-k can create subtokens across canonical token boundaries. Thus, it provides even higher flexibility than StochasTok-uni.

Lastly, we include full uniform sampling as a baseline, denoted Uniform. This offers complete support over $\mathcal{T}_{\mathcal{V}}(\boldsymbol{x})$, but substantially increases computational cost and the number of splits generated. In Sec. H.1 we present empirical results for the number of splits generated by each variant.

5.2 Fine-tuning robustness

We now analyse if increased uniformity over non-canonical tokenizations improves robustness during fine-tuning. Fig. 6 reports the drop in accuracy when switching from canonical queries to uniformly sampled tokenizations. As in Sec. 4.2, canonical fine-tuning results in the largest drop in accuracy (-29.8%). Further note that increasing the number of splits, *i.e.*, increasing α_{fine}, improves robustness for all methods. Next, recall that both StochasTok-uni and Uniform-k produce uniform samples conditioned on the number of splits, but with different distributions over split-counts and different support. Despite those differences, we observe that both approaches generally result in a smaller drop in accuracy than StochasTok, indicating that increased uniformity and support are beneficial. In addition, Fig. 1 further highlights the benefits of a uniform sampling distribution.

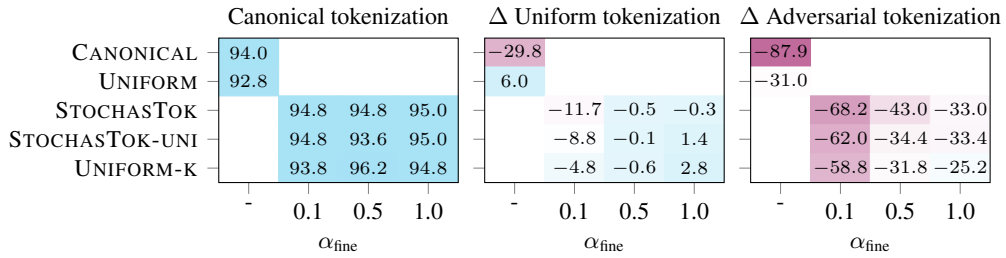


Figure 6: Canonical accuracy *(left)* and accuracy change under uniform *(center)* and adversarial tokenization *(right)* across different fine-tuning strategies for `Llama-1b` on LANGUAGE GAME.

5.3 IN-CONTEXT LEARNING ROBUSTNESS

Having shown that more uniform sampling improves robustness during fine-tuning, we now investigate the ICL setting. Following Sec. 4.3, we use a frozen model with $K = 10$ in-context examples, each tokenized once using STOCHASTOK, STOCHASTOK-UNI, UNIFORM-K, or UNIFORM. Results in Table 10 (Sec. I.3) show mild gains over canonical context tokenization, but little difference between stochastic schemes, except for UNIFORM, which provides the highest degree of stochasticity. Thus, it may be needed to use a larger number of in-context examples to induce robustness via ICL.

6 ROBUSTNESS TO ADVERSARIAL TOKENIZATION

Next, we assess adversarial robustness using the adversarial attack mechanism of Geh et al. (2025), following the approach outlined in Sec. 2. We provide additional results for `Tiny-LLM` in Figs. 12 and 13 (Sec. I.2.1), for `GPT2-xl` in Fig. 15 (Sec. I.2.2), and for `Gemma-1b` with a larger vocabulary in Sec. I.2.5. We also compare against BPE-DROPOUT for `Tiny-LLM`, `Llama-1b`, and `Qwen-0.6b` in Fig. 21 (Sec. I.2.6). Finally, we include an ablation over model sizes in Sec. I.2.4. All findings are consistent with those presented here.

Fig. 6 shows the results for the `Llama-1b` model fine-tuned canonically or with stochastic tokenization and evaluated either on canonical tokens *(left)* or under adversarial attacks *(right)*. We see that non-canonical tokenization during fine-tuning substantially improves adversarial robustness, *i.e.*, accuracy under canonical fine-tuning drops from 94% to 6.1% (almost 90 percentage points), whereas under UNIFORM-K it only drops from 94.8% to 69.6%. In Sec. I.2.3, we disentangle the effects of uniformity and support for STOCHASTOK and STOCHASTOK-UNI, finding that increased support is the main source of robustness. However, an additional factor is the induced split-count distribution: STOCHASTOK-UNI remains biased towards fewer splits per token, whereas UNIFORM-K induces a broader distribution (see Sec. H), exposing the model to more diverse tokenizations.

Lastly, we evaluate whether stochastic tokenization during ICL provides any robustness to adversarial attacks. Table 11 (Sec. I.3) summarizes the experiments and indicates that, surprisingly, ICL with stochastic tokenization can provide mild robustness to adversarial tokenization.

6.1 THEORETICAL ANALYSIS

Beyond the empirical results, we aim to theoretically characterize how tokenization affects robustness. We focus on evasion attacks, *i.e.*, adversarial perturbations applied at test time. Let $f\colon \mathcal{V}^* \to \mathcal{Y}$ be a trained classifier and $v \in \mathcal{V}^*$ an input with output $y = f(v)$. An attacker aims to find a perturbed input $\tilde{v}$ such that $f(\tilde{v})$ differs from the original prediction. Here, f represents the LLM *after* the tokenizer, comprising the embedding function and subsequent transformer blocks.

As the generated adversarial perturbation should be "similar" to the original input, we consider a similarity budget ε (the maximum possible changes to the input), comprising the set of valid adversarial perturbations defined by a neighbourhood $\mathcal{B}(x, \varepsilon)$. In our setting $\mathcal{B}(x, \varepsilon) = \mathcal{T}_{\mathcal{V}}^{k}(x)$ with edit distance at most k. Now, the adversarial aim is to find, within $\mathcal{B}(x, \varepsilon)$, points that result in the

worst prediction (the model's *adversarial risk*), which can be formulated as

$$\mathcal{R}_\varepsilon[f] = \mathop{\mathbb{E}}_{x\in\mathcal{D}_\mathcal{X}}\left[\sup_{\tilde{v}\in\mathcal{B}(x,\varepsilon)} d_\mathcal{Y}\left(f\left(\tilde{v}\right), f\left(\tau(x)\right)\right)\right] \tag{2}$$

where $d_\mathcal{Y} = \|\cdot\|_2$ denotes distance in output space $\mathcal{Y}$. From a defence perspective, the objective is to ensure that the risk remains small, thereby keeping predictions within the same output region and making it harder to find a valid perturbation. This goal can be formulated as:

Definition 1 (Adversarial Robustness)**.** *The classifier f is said to be* (ε, γ)-robust *if its adversarial risk with respect to the classifier f satisfies: $\mathcal{R}_\varepsilon[f] \leq \gamma$.*

Definition 1 uses an upper-bound perspective (γ) as computing the exact risk is generally intractable. A smaller value of γ indicates stronger robustness guarantees. Consequently, comparing the resulting bounds provides a principled way to assess and contrast the model's robustness. For analytical tractability, we consider a 1-layer Transformer-based model in which all activation functions are assumed to be 1-Lipschitz, which is the case for many common activations, *e.g.*, `tanh` and `sigmoid` (Virmaux & Scaman, 2018). An L-layer transformer can be represented as a composition of L single-layer functions, with the overall Lipschitz factor obtained by multiplying the corresponding layerwise factors. We additionally assume the output of the embedding function to be bounded in terms of its norm (*e.g.*, $[0,1]^{n\times d}$, with n being the number of tokens and d the hidden dimension). In this setting, the model's underlying robustness, γ, can be derived and is given by the following.

Theorem 6.1. *Let $f\colon \mathcal{V}^* \to \mathcal{Y}$ be a classifier following our considered problem setup. Then f is* (ε, γ)-robust*, with $\gamma = 2\|W\|\left(\frac{d}{d-1}\right)^2 C_1C_2\sqrt{k}$, where W is the weight matrix of the embedding layer and C_1C_2 is proportional to the Lipschitz constant of the attention block (Ennadir et al., 2025).*

Details and proof of Theorem 6.1 are given in Sec. G. The bound depends on the weight norms and the perturbation radius k, which is linked to the tokenization. Additionally, the bound depends explicitly on the perturbation neighbourhood, defined through the token-level edit distance, linking it to the support of the sampling distribution. Our considered methods differ in their support over this neighbourhood, with incomplete support leaving the model vulnerable to uncovered tokenizations.

In connection to this, we empirically evaluated the sensitivity of each layer by comparing unit-normalized embeddings for 50 uniformly sampled non-canonical tokenizations of the top 1k most frequent English words. Fig. 22 (Sec. I.2.7) shows the average distance for canonical and stochastic tokenization schemes. We find that stochastic tokenization results in smaller mean normalized distances, indicating smaller local Lipschitz constants and suggesting reduced vulnerability.

7 Conclusion & discussion

In this paper, we address the fundamental question: **'Can stochastic tokenization improve robustness?'**, and find that stochastic tokenization during training can provide robustness against random perturbations and adversarial tokenization attacks. To this end, we first examined the robustness of StochasTok (Sims et al., 2026) and found that it provides some robustness, but the sampling distribution is both biased and incomplete, making it vulnerable to adversarial attacks. Hence, we develop alternative sampling mechanisms, specifically StochasTok-uni and Uniform-k, both of which ensure a uniform sampling distribution (conditional in the case of StochasTok-uni) with support spanning all non-canonical tokenizations within some neighbourhood. Uniform sampling often performs favourably compared to StochasTok and provides stronger adversarial robustness, supported by our theoretical analysis of adversarial risk. A short practitioner's guide discussing computational overhead is provided in Sec. B.

Limitations Our analysis is currently limited to question-answering tasks; extending it to open-ended generation and instruction following remains important, particularly for assessing jailbreak or alignment robustness in more realistic settings. Such an evaluation would require studying post-trained models, where robustness may depend on the specific alignment procedure and training pipeline, and is therefore beyond the scope of this work. Furthermore, extending the sensitivity analysis in Section I.2.7 to include a mechanistic interpretability analysis is a promising direction for future work. Finally, our current analysis is based on English datasets. The sensitivity of tokenizers for other languages with higher fertility remains an open but important question for future work.

AI USE STATEMENT

In this work, we have used generative AI tools to create or modify scientific figures, obtain feedback on the research paper to improve readability and language, propose alternative wording, and provide feedback on the presentation of experiments and results. We have also used generative AI to assist with code writing and cleanup. The LLM-generated code was verified and tested for correctness. We have not used generative AI to generate synthetic datasets, develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, assist with translation, or clean and reformat datasets. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024.

Gül Sena Altıntaş, Malikeh Ehghaghi, Brian Lester, Fengyuan Liu, Wanru Zhao, Marco Ciccone, and Colin Raffel. Toksuite: Measuring the impact of tokenizer choice on language model behavior. *arXiv preprint arXiv:2512.20757*, 2025.

Thomas Bauwens, David Kaczér, and Miryam De Lhoneux. GRaMPa: Subword regularisation by skewing uniform segmentation distributions with an efficient path-counting markov model. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 24228–24257, 2025.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 7432–7439, 2020.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Yekun Chai, Yewei Fang, Qiwei Peng, and Xuhong Li. Tokenization falling short: On subword robustness in large language models. In *Findings of the Association for Computational Linguistics (EMNLP)*, pp. 1582–1599, 2024.

Jonathan H Clark, Dan Garrette, Iulia Turc, and John Wieting. Canine: Pre-training an efficient tokenization-free encoder for language representation. *Transactions of the Association for Computational Linguistics*, 10:73–91, 2022.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? Try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Marco Cognetta, Vilém Zouhar, and Naoaki Okazaki. Distributional properties of subword regularization. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 10753–10763, 2024.

Adrian Cosma, Stefan Ruseti, Emilian Radoi, and Mihai Dascalu. Training language models with homotokens leads to delayed overfitting. *arXiv preprint arXiv:2601.02867*, 2026.

Boyi Deng, Wenjie Wang, Fuli Feng, Yang Deng, Qifan Wang, and Xiangnan He. Attack prompt generation for red teaming and defending large language models. In *Findings of the Association for Computational Linguistics (EMNLP)*, pp. 2176–2189, 2023.

Zhichen Dong, Zhanhui Zhou, Chao Yang, Jing Shao, and Yu Qiao. Attacks, defenses and evaluations for LLM conversation safety: A survey. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 6734–6747, 2024.

Lukas Edman, Helmut Schmid, and Alexander Fraser. CUTE: Measuring LLMs' understanding of their tokens. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 3017–3026, 2024.

Sofiane Ennadir, Levente Zólyomi, Oleg Smirnov, Tianze Wang, John Pertoft, Filip Cornell, and Lele Cao. Pool me wisely: On the effect of pooling in transformer-based models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Sheridan Feucht, David Atkinson, Byron C Wallace, and David Bau. Token erasure as a footprint of implicit vocabulary items in LLMs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 9727–9739, 2024.

Philip Gage. A new algorithm for data compression. *The C Users Journal*, 12(2):23–38, 1994.

Renato Geh, Honghua Zhang, Kareem Ahmed, Benjie Wang, and Guy Van den Broeck. Where is the signal in tokenization space? In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 3966–3979, 2024.

Renato Geh, Zilei Shao, and Guy Van den Broeck. Adversarial tokenization. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 20738–20765, 2025.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The Llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Dylan Hillier, Leon Guertler, Cheston Tan, Palaash Agrawal, Chen Ruirui, and Bobby Cheng. Super tiny language models. *arXiv preprint arXiv:2405.14159*, 2024.

Tatsuya Hiraoka. MaxMatch-dropout: Subword regularization for wordpiece. In *Proceedings of the 29th International Conference on Computational Linguistics*, pp. 4864–4872, 2022.

Tatsuya Hiraoka and Kentaro Inui. Spelling-out is not straightforward: LLMs' capability of tokenization from token to characters. In *Findings of the Association for Computational Linguistics (EMNLP)*, pp. 13340–13353, 2025.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations (ICLR)*, 2022.

Itay Itzhak and Omer Levy. Models in a spelling bee: Language models implicitly learn the character composition of tokens. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 5061–5068, 2022.

Guy Kaplan, Matanel Oren, Yuval Reif, and Roy Schwartz. From tokens to words: On the inner lexicon of LLMs. In *The Thirteenth International Conference on Learning Representations (ICLR)*, 2025.

Ayush Kaushal and Kyle Mahowald. What do tokens know about their characters and how do they know it? In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2487–2507, 2022.

Taku Kudo. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 66–75, 2018.

VI Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. In *Dokl. Akad. Nauk SSSR*, volume 163, pp. 845–848, 1965.

Miles Q Li and Benjamin CM Fung. Security concerns for large language models: A survey. *Journal of Information Security and Applications*, 95:104284, 2025.

Artidoro Pagnoni, Ramakanth Pasunuru, Pedro Rodriguez, John Nguyen, Benjamin Muller, Margaret Li, Chunting Zhou, Lili Yu, Jason E Weston, Luke Zettlemoyer, et al. Byte latent transformer: Patches scale better than tokens. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9238–9258, 2025.

Ethan Perez, Saffron Huang, Francis Song, Trevor Cai, Roman Ring, John Aslanides, Amelia Glaese, Nat McAleese, and Geoffrey Irving. Red teaming language models with language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 3419–3448, 2022.

Ivan Provilkov, Dmitrii Emelianenko, and Elena Voita. BPE-dropout: Simple and effective subword regularization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 1882–1892, 2020.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S Gordon. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *AAAI spring symposium: logical formalizations of commonsense reasoning*, pp. 90–95, 2011.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the association for computational linguistics (Volume 1: Long papers)*, pp. 1715–1725, 2016.

Anya Sims, Thom Foster, Klara Kaleb, Tuan-Duy H Nguyen, Joseph Lee, Jakob N Foerster, Yee Whye Teh, and Cong Lu. StochasTok: Improving fine-grained subword understanding in LLMs. In *The Fourteenth International Conference on Learning Representations (ICLR)*, 2026.

Kevin Slagle. Spacebyte: Towards deleting tokenization from large language modeling. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Feifan Song, Bowen Yu, Minghao Li, Haiyang Yu, Fei Huang, Yongbin Li, and Houfeng Wang. Preference ranking optimization for human alignment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 18990–18998, 2024.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. CommonsenseQA: A question answering challenge targeting commonsense knowledge. 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)*, pp. 4149–4158, 2019.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: A family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Aladin Virmaux and Kevin Scaman. Lipschitz regularity of deep neural networks: Analysis and efficient estimation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. Finetuned language models are zero-shot learners. In *The Tenth International Conference on Learning Representations (ICLR)*, 2022.

Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. ByT5: Towards a token-free future with pre-trained byte-to-byte models. *Transactions of the Association for Computational Linguistics*, 10:291–306, 2022.

Qingyu Yin, Xuzheng He, Chak Tou Leong, Fan Wang, Yanzhao Yan, Xiaoyu Shen, and Qiang Zhang. Deeper insights without updates: The power of in-context learning over fine-tuning. In *Findings of the Association for Computational Linguistics (EMNLP)*, pp. 4138–4151, 2024.

Lili Yu, Dániel Simig, Colin Flaherty, Armen Aghajanyan, Luke Zettlemoyer, and Mike Lewis. Megabyte: Predicting million-byte sequences with multiscale transformers. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800, 2019.

Brian Siyuan Zheng, Alisa Liu, Orevaoghene Ahia, Jonathan Hayase, Yejin Choi, and Noah A Smith. Broken tokens? Your language model can secretly handle non-canonical tokenizations. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

A Related work

Robustness to tokenizations Typically, LLMs are trained with deterministic subword tokenization (Sennrich et al., 2016; Kudo, 2018), which has been argued to obscure subtoken and character-level information (Chai et al., 2024; Edman et al., 2024). Recent work has demonstrated that models still encode within-word information without direct access to its characters (Itzhak & Levy, 2022; Hiraoka & Inui, 2025), with larger models performing better in general (Kaushal & Mahowald, 2022). Evidence suggests that models convert subword tokens into an implicit vocabulary, which allows models to retrieve character-level information (Feucht et al., 2024; Kaplan et al., 2025). In addition, Zheng et al. (2025) demonstrate the robustness of canonically trained LLMs to unseen non-canonical tokenizations at test time, especially after instruction-tuning. Post-training on curated datasets with canonical tokenization, however, has been argued to render models vulnerable to adversarial tokenization attacks (Geh et al., 2025).

In this work, we systematically analyse tokenization brittleness via canonical fine-tuning for canonical pre-training, stochastic-tokenization pre-training, and large pretrained LLMs. We then study whether stochastic-tokenization fine-tuning improves robustness to both random and adversarial non-canonical tokenizations.

Training with stochastic tokenizations Several works have proposed training with stochastic tokenizations to improve subword understanding. Among the first, Provilkov et al. (2020) introduce stochasticity by randomly dropping merge rules. Cognetta et al. (2024) show that BPE-dropout (Provilkov et al., 2020) and MaxMatch-dropout (Hiraoka, 2022) result in a non-uniform distribution over tokenizations, and show improvements in machine translation quality with uniform sampling. Bauwens et al. (2025) argue that uniform sampling leads to tokens with reduced character count and propose length-aware sampling. Cosma et al. (2026) sample non-canonical tokenizations, and use an auxiliary encoder to condition canonical next-token prediction on them. They demonstrate improved data efficiency and delayed overfitting in language-model training. Sims et al. (2026) randomly split tokens into pairs of subtokens and keep the model architecture and tokenizer vocabulary the same. By including stochastic tokenization in model pre-training, they show improved subword understanding and suggest improved robustness to non-canonical tokenizations.

Our work complements these approaches by studying stochastic tokenization across training regimes, focusing on both subword understanding and robustness to non-canonical tokenizations. We show that the distribution over tokenizations affects both average and adversarial robustness.

Tokenization-free models Another line of work avoids tokenization altogether by operating on a character or byte-level directly. This removes dependency on a learnt tokenizer, improves handling of out-of-vocabulary words, and preserves character-level information. However, this results in byte sequences that are substantially longer than token sequences, motivating work on reducing the effective sequence length and making such models practically scalable (Clark et al., 2022; Xue et al., 2022; Yu et al., 2023; Slagle, 2024; Pagnoni et al., 2025).

While these approaches avoid brittleness from deterministic subword tokenization by construction, they address a different setting from ours. Our analysis focuses on improving robustness while keeping the model architecture and tokenizer vocabulary unchanged.

B PRACTITIONER'S GUIDE

We found that canonical tokenization results in brittle models that are vulnerable to non-canonical tokenizations or adversarial attacks. Further, we found that fine-tuning with stochastic tokenization substantially increases robustness, without additional cost during inference. This indicates that, when robustness to tokenizations is relevant, fine-tuning should preferably be performed with stochastic tokenization instead of canonical tokenization.

In practice, the main computational cost is incurred before and during training. First, stochastic tokenizations need to be generated. In our setup, we precompute them in parallel on CPU. For long inputs, MRMDD construction is the main preprocessing bottleneck; this cost is incurred offline and can be parallelized independently across inputs. For very long inputs, the construction cost can additionally be bounded through blockwise processing, with the split budget allocated across blocks. Second, there is training-time overhead from processing non-canonical tokenizations, which depends on the number of sampled tokenizations per original example and the number of splits. This makes two properties practically important: the sampling scheme should cover relevant non-canonical tokenizations without strong bias, and it should allow direct control over the number of splits. Direct adversarial training is also a possible defence, but requires a model-dependent discrete search with multiple forward passes per input and therefore incurs substantially greater training-time computation than our offline stochastic preprocessing.

C TRAINING DETAILS

Each experiment was run on a single NVIDIA GPU (V100/A100/H100/H200, 16–141GB memory) on an academic compute cluster. Stochastic tokenizations were precomputed in parallel on 16 CPU cores.

For LLMs trained from scratch, we use the architecture proposed in Hillier et al. (2024) and used in Sims et al. (2026). The architectural details are summarized in Table 2. The pre-training setup is summarized in Table 3. The fine-tuning step uses full parameter updating, with details specified in Table 4.

Additionally, we use pre-trained weights and architectures from Hugging Face for
GPT2-xl (https://huggingface.co/openai-community/gpt2-xl),
Llama-1b (https://huggingface.co/meta-llama/Llama-3.2-1B),
Llama-3b (https://huggingface.co/meta-llama/Llama-3.2-3B),
Llama-8b (https://huggingface.co/meta-llama/Llama-3.1-8B),
Qwen-0.6b (https://huggingface.co/Qwen/Qwen3-0.6B), and
Gemma-1b (https://huggingface.co/google/gemma-3-1b-pt).

For the fine-tuning step, we train LoRA adapters with the training setup as specified in Table 5.

Table 2: Model architecture of **Tiny-LLM**.

Model architecture (Tiny-LLM)	
Number of layers	8
Hidden dimension	512
Attention heads	16 (grouped, group size 4)
Feed-forward network	SwiGLU, dimension 1320
Normalization	RMSNorm (no bias)
Positional encoding	Rotary positional embeddings (RoPE)
Context length	512
tokenizer type	GPT
Vocabulary size	50,257
Embedding tying	Input/output embeddings tied

Table 3: Pre-training setup for **Tiny-LLM** on OpenWebText.

Pre-training (Tiny-LLM)	
Training objective	Autoregressive language modeling
Dataset	OpenWebText
tokenization	Canonical, StochasTok ($\alpha_{\text{pre}} \in \{0.1, 0.5\}$)
Training steps	30,000
Batch size	480
Optimizer	AdamW
Learning rate	6×10^{-4} (min. 6×10^{-5})
Adam parameters	$\beta_1 = 0.9$, $\beta_2 = 0.95$
Weight decay	0.1
Learning-rate schedule	Linear warmup (5k steps) + linear decay
Gradient clipping	1.0
Dropout	0.1

Table 4: Supervised fine-tuning setup for **Tiny-LLM** using stochastic tokenization.

Supervised fine-tuning (Tiny-LLM)	
Training objective	Supervised fine-tuning
tokenization mode	Canonical, StochasTok, StochasTok-uni, Uniform-k, Uniform, BPE-Dropout
Training steps	3,000
Batch size	480
Optimizer	AdamW
Learning rate	1×10^{-4} (min. 1×10^{-5})
Adam parameters	$\beta_1 = 0.9$, $\beta_2 = 0.95$
Weight decay	0.05
Warmup steps	500
Dropout	0.1

Table 5: Supervised fine-tuning configuration for Hugging Face models.

Supervised fine-tuning (GPT2-xl, Llama-1b, Llama-3b, Llama-8b, Qwen-0.6b, Gemma-1b)	
Training objective	Supervised fine-tuning
tokenization mode	Canonical, StochasTok, StochasTok-uni, Uniform-k, Uniform, BPE-Dropout
Training steps	1,500
Batch size	16
Optimizer	AdamW
Learning rate	1×10^{-4} (min. 1×10^{-5})
Adam parameters	$\beta_1 = 0.9$, $\beta_2 = 0.95$
Weight decay	0.01
Warmup steps	150
LoRA rank (r)	8
LoRA scaling (α)	32
LoRA dropout	0.05
Task type	Causal language modeling

D EXPERIMENTAL DETAILS

In this section, we describe our evaluation setup in detail.

D.1 NORMALIZED NUMBER OF SPLITS

Assume a string sequence $\boldsymbol{x} = (x_1, \ldots, x_s)$ and its canonical token sequence $\boldsymbol{v}^c = (v_1^c, \ldots, v_m^c)$. Let $\boldsymbol{v} \sim P(\cdot \mid \boldsymbol{x})$ be a non-canonical tokenization of $\boldsymbol{x}$, where $\boldsymbol{v} = (v_1, \ldots, v_p) \in \mathcal{T}_{\mathcal{V}}(\boldsymbol{x})$. We define the per-token split-count vector $\boldsymbol{S} = (S_1, \ldots, S_m)$ that describes the number of splits induced in each canonical token.

We define the normalized number of splits for a token sequence as

$$\alpha(\boldsymbol{v}, \boldsymbol{v}^c) := \frac{\sum_i S_i}{|\boldsymbol{v}^c|}, \tag{3}$$

which corresponds to the average number of splits per canonical token.

D.2 AVERAGE TOKENIZATION ROBUSTNESS

We quantify average accuracy given stochastic tokenizations in a multiple-choice setting. For each input question string $\boldsymbol{x}$, we have answer options $\{\boldsymbol{a}_c\}_{c=1}^C$ and correct index $y \in \{1, \ldots, C\}$. Assume the stochastic tokenization of the query $\boldsymbol{v} \sim P(\cdot \mid \boldsymbol{x}; \alpha_{\text{eval}})$, and canonical tokenization of the answer options $\{\boldsymbol{v}_{\boldsymbol{a}_c}\}_{c=1}^C$. The model predicts the answer with the highest conditional probability

$$\hat{y}\left(\boldsymbol{v}, \{\boldsymbol{v}_{\boldsymbol{a}_c}\}_{c=1}^C\right) = \underset{c \in \{1,\ldots,C\}}{\arg\max}\ p(\boldsymbol{v}_{\boldsymbol{a}_c} \mid \boldsymbol{v}). \tag{4}$$

We approximate the expected accuracy under stochastic tokenization by Monte Carlo sampling. We draw $M = 10$ tokenizations $\{\boldsymbol{v}^{(m)}\}_{m=1}^M$ and compute

$$\text{Acc} = \frac{1}{M} \sum_{m=1}^{M} \mathbf{1}\left(\hat{y}\left(\boldsymbol{v}^{(m)}, \{\boldsymbol{v}_{\boldsymbol{a}_c}\}_{c=1}^C\right) = y\right). \tag{5}$$

We report the average accuracy over $N = 500$ data samples,

$$\overline{\text{Acc}} = \frac{1}{N} \sum_{n=1}^{N} \frac{1}{M} \sum_{m=1}^{M} \mathbf{1}\left(\hat{y}\left(\boldsymbol{v}_n^{(m)}, \{\boldsymbol{v}_{\boldsymbol{a}_c,n}\}_{c=1}^C\right) = y_n\right). \tag{6}$$

D.3 ADVERSARIAL TOKENIZATION ROBUSTNESS

To evaluate adversarial tokenization robustness, we use the greedy search algorithm by Geh et al. (2025), summarized in Alg. 1. For a given input string $\boldsymbol{x}$, the attacker selects an initial tokenization. Unless stated otherwise, we use the canonical tokenization as a default initialization and use a maximum of 50 attack iterations with early stopping when no improving neighbour is found. In ablation experiments, we report results for randomly drawn initial tokenizations. Let $\boldsymbol{v}_n^{\text{adv}}$ denote the adversarial tokenization for input string $\boldsymbol{x}_n$.

We compare the clean average accuracy on the canonical tokenization

$$\overline{\text{Acc}}_{\text{clean}} = \frac{1}{N} \sum_{n=1}^{N} \mathbb{1}\left(\hat{y}\left(\tau(\boldsymbol{x}_n), \{\boldsymbol{v}_{\boldsymbol{a}_c,n}\}_{c=1}^C\right) = y_n\right) \tag{7}$$

and the adversarial accuracy

$$\overline{\text{Acc}}_{\text{adv}} = \frac{1}{N} \sum_{n=1}^{N} \mathbb{1}\left(\hat{y}\left(\boldsymbol{v}_n^{\text{adv}}, \{\boldsymbol{v}_{\boldsymbol{a}_c,n}\}_{c=1}^C\right) = y_n\right). \tag{8}$$

D.4 In-context learning

Let $(\boldsymbol{v}_i, \boldsymbol{v}_{\boldsymbol{a}_i})_{i=1}^K$ be K tokenized context examples and let $\boldsymbol{v}^*$ be a test query with answer options $\{\boldsymbol{v}^*_{\boldsymbol{a}_c}\}_{c=1}^C$. To evaluate in-context learning robustness, we sample stochastic tokenizations for each context example $\boldsymbol{v}_i \sim P(\boldsymbol{v}_i \mid \boldsymbol{x}_i; \alpha_{\text{ICL}})$, and test whether this improves robustness to non-canonical query tokenizations $\boldsymbol{v}^* \sim P(\boldsymbol{v}^* \mid \boldsymbol{x}^*; \alpha_{\text{eval}})$. We denote the number of solved examples presented in the context by K. We report the average accuracy over $N = 500$ data samples

$$\overline{\text{Acc}} = \frac{1}{N} \sum_{n=1}^{N} \mathbb{1}\left(\hat{y}\left((\boldsymbol{v}_1, \boldsymbol{v}_{\boldsymbol{a}_1}, \ldots, \boldsymbol{v}_K, \boldsymbol{v}_{\boldsymbol{a}_K}, \boldsymbol{v}^*_n), \{\boldsymbol{v}^*_{\boldsymbol{a}_c,n}\}_{c=1}^C\right) = y^*_n\right) \tag{9}$$

where y^*_n denotes the ground-truth index and $\hat{y}(\cdot)$ the predicted answer index.

For the adversarial attack, we define the adversarial objective as maximizing the margin of the conditional probability assigned to each answer option c

$$\bar{z}_c(\boldsymbol{v}^*) = p(\boldsymbol{v}_{\boldsymbol{a}_c} \mid \boldsymbol{v}_1, \boldsymbol{v}_{\boldsymbol{a}_1}, \ldots, \boldsymbol{v}_K, \boldsymbol{v}_{\boldsymbol{a}_K}, \boldsymbol{v}^*)\,. \tag{10}$$

D.5 Distances of representations

We analyse how training with different stochastic tokenization schemes affects the internal representations of the same word with different non-canonical tokenizations by measuring the distance between its hidden states. We consider the 1,000 most common English words. For each word $\boldsymbol{x}$, we draw 50 tokenizations uniformly from $\mathcal{T}_{\mathcal{V}}(\boldsymbol{x})$.

Let $h_\ell(\boldsymbol{v}) \in \mathbb{R}^d$ denote the hidden representation at model layer ℓ for tokenization $\boldsymbol{v}$, obtained via last-token pooling. For each layer ℓ, we report the squared L^2 distance between the canonical tokenization v^c and a sampled tokenization $\boldsymbol{v}$, where the length of both vectors has been scaled to have unit norm:

$$d_\ell(\boldsymbol{v}, \boldsymbol{v}^c) = \left\| \frac{h_\ell(\boldsymbol{v})}{\|h_\ell(\boldsymbol{v})\|_2} - \frac{h_\ell(\boldsymbol{v}^c)}{\|h_\ell(\boldsymbol{v}^c)\|_2} \right\|_2 \tag{11}$$

We report the average of $d_\ell(\boldsymbol{v}, \boldsymbol{v}^c)$ across words and sampled tokenizations.

E Datasets

E.1 Language games

Language Game (Sims et al., 2026) consists of six question types that require subword understanding (count letter, contains letter, starts with string, ends with string, longest word, shortest word). For the training set we randomly sample from the top 1k English words.

Below, we illustrate the effect of our stochastic tokenization schemes for questions of the Language Game data set tokenizer with a Llama 3 tokenizer. Hyphens ("-") indicate token boundaries.

Canonical:

- Which-choice-has-the-most-letter-'-b-'s-?-These-are-the-possible-option-words-:-[-up-,-baby-,-rain-,-glad-].-Answer-:--baby-
- Which-option-ends-with-'o'-?-The-possible-option-strings-:-[-no-,-hard-,-by-,-happen-].-Answer-:--no-
- What-word-is-the-shortest-?-The-available-option-strings-are-:-[-row-,-suffix-,-gather-,-energy-].-Answer-:--row-

StochasTok ($\alpha = 1$):

- Wh-i-ch-choice-has-the-most-le-t-t-er-'-b-'s-?-These-are-the-possible-option-words-:-[-up-,-baby-,-rain-,-g-lad-].-A-n-sw-er-:--baby-
- Which-option-end-s-with-'o'-?-The-possible-option-s-tring-s-:-[-no-,-hard-,-b-y-,-happen-].-Answer-:--no-

- Wh-at-word-i-s-the-shortest-?-The-available-option-string-s-are-:-[-r-o-w-,-suffix-,-g-ather-,-e-n-ergy-].-Ans-wer-:--row-

STOCHASTOK-UNI ($\alpha = 1$):

- W-hic-h-choice-has-the-mo-s-t-letter-'-b-'-s-?-The-se-are-the-possible-opt-ion-words-:-[-u-p-,-baby-,-rain-,-g-la-d-].-Answer-:--baby-
- Which-op-ti-on-en-ds-w-i-t-h-'o'-?-The-possible-op-tion-strings-:-[-no-,-h-a-r-d-,-by-,-happen-].-Answer-:--no-
- What-wo-rd-is-th-e-shortest-?-The-available-o-p-tion-strings-a-r-e-:-[-row-,-su-ff-ix-,-gather-,-ener-gy-].-Answer-:--row-

UNIFORM-K ($\alpha = 1$):

- Which-c-h-o-ic-e-has-th-e-mos-t-letter-'-b-'s-?-The-se-are-the-poss-ible-op-tion-wo-r-ds-:-[-up-,-ba-by-,-rain-,-glad-].-An-s-we-r-:--baby-
- W-h-ic-h-option-end-s-w-ith-'o'-?-T-he-possible-option-str-ing-s-:-[-no-,-h-ard-,-by-,-happen-].-An-sw-er-:--no-
- What-w-o-rd-is-the-s-hort-est-?-The-a-vailable-option-s-t-r-i-n-gs-are-:-[-row-,-suffix-,-ga-ther-,-energy-].-Ans-w-er-:--row-

UNIFORM:

- Wh-i-c-h-cho-i-c-e-h-as-t-h-e-m-o-st-le-tte-r-'-b-'s-?-Th-es-e-a-r-e-t-he-po-ssi-b-l-e-op-t-ion-words-:-[-u-p-,-b-a-b-y-,-r-ain-,-g-lad-].-A-ns-w-e-r-:--baby-
- W-h-i-ch-op-ti-on-ends-wi-t-h-'o'-?-T-h-e-po-s-s-ible-option-s-tr-i-n-g-s-:-[-no-,-ha-rd-,-b-y-,-ha-p-pe-n-].-Ans-w-er-:--no-
- W-hat-word-i-s-th-e-s-ho-r-t-e-st-?-Th-e-a-v-a-i-l-abl-e-op-t-i-o-n-s-t-ring-s-a-re-:-[-row-,-s-u-ff-i-x-,-g-a-th-er-,-e-n-er-g-y-].-An-s-wer-:--row-

CUTE (Edman et al., 2024) tests the Character-level Understanding of Tokens of models. The original data set was intended for zero-shot evaluation, providing only a test split. Thus, we follow Sims et al. (2026) in generating a custom CUTE data set for training and evaluating the smaller scale LLMs. We generate questions for seven subword tasks (contains letter, delete letter, insert letter, orthography, spelling, inverse spelling, substitute letter). For the training set we randomly sample from the top 1k English words.

CANONICAL:

- Spell-out-the-word-"-stood-".-Answer-:-s-t-o-o-d-
- Is-there-a-"-e-"-in-"-salt-"?-Answer-:-No-
- Delete-every-instance-of-"-c-"-in-"-molecule-".-Answer-:-m-ole-ule-

E.2 GENERAL BENCHMARKS

ARC-E (Clark et al., 2018) consists of natural, grade-school science questions with four answer options.

CANONICAL:

- Which-factor-will-most-likely-cause-a-person-to-develop-a-fever-?-a-bacterial-population-in-the-bloodstream-
- L-ich-ens-are-symb-iotic-organisms-made-of-green-algae-and-fungi-.-What-do-the-green-algae-supply-to-the-fungi-in-this-symb-iotic-relationship-?-food-

- When-a-switch-is-used-in-an-electrical-circuit-,-the-switch-can-stop-and-start-the-flow-of-current-.-

COPA (Roemmele et al., 2011) contains multiple-choice questions that test causal reasoning with two answer options.

CANONICAL:

- My-body-cast-a-shadow-over-the-grass-.-cause-?-The-sun-was-rising-.-
- The-woman-tolerated-her-friend-'s-difficult-behavior-.-cause-?-The-woman-knew-her-friend-was-going-through-a-hard-time-.-
- The-women-met-for-coffee-.-cause-?-They-wanted-to-catch-up-with-each-other-.-

CSQA (Talmor et al., 2019) contains multiple-choice questions that test commonsense knowledge with five answer options.

CANONICAL:

- The-sanctions-against-the-school-were-a-punishing-blow-,-and-they-seemed-to-what-the-efforts-the-school-had-made-to-change-?-ignore-
- Sam-my-wanted-to-go-to-where-the-people-were-.-Where-might-he-go-?-populated-areas-
- To-locate-a-ch-oker-not-located-in-a-jewelry-box-or-boutique-where-would-you-go-?-jewelry-store-

HELLASWAG (Zellers et al., 2019) contains multiple-choice questions that test the most natural continuation given four answer options.

CANONICAL:

- [-header-]-How-to-dry-pant-yh-ose-in-a-hurry-[-title-]-Lay-the-pant-yh-ose-flat-on-your-towel-.-[-step-]-Use-a-dry-bath-towel-for-best-results-.-The-thicker-the-towel-is-,-the-more-effective-this-method-will-be-.-If-you-do-not-have-a-thick-towel-,-repeat-the-method-using-two-separate-towels-.-[-title-]-Roll-pant-yh-ose-into-your-thick-towel-.-
- [-header-]-How-to-make-german-style-meat-lo-af-[-title-]-Combine-beef-,-pork-,-onions-,-beaten-eggs-,-bread-crumbs-,-3-tablespoons-water-,-parsley-,-pap-rika-,-prepared-mustard-and-salt-.-[-step-]-Mix-well-.-[-title-]-Press-the-mixture-into-the-8-"-baking-pan-.-[-step-]-Place-the-3-hard-cooked-eggs-in-a-row-down-the-middle-of-the-meat-mixture-,-then-fold-up-both-sides-,-so-you-have-a-loaf-.-[-title-]-Pre-heat-the-oven-to-375-.-
- [-header-]-How-to-feel-full-without-eating-[-title-]-Chew-a-piece-of-gum-.-[-step-]-Ch-ewing-a-piece-of-gum-will-trigger-your-brain-and-stomach-to-believe-you-are-about-to-eat-or-feeling-full-.-This-will-not-only-stimulate-your-mind-to-feel-full-,-but-will-also-make-sure-that-your-mouth-is-too-busy-to-eat-.-[-sub-steps-]-Make-sure-to-chew-sugar-less-gum-so-that-you-don-'t-get-unnecessary-calories-.-Ch-ewing-gum-can-even-burn-11-calories-an-hour-.-

PIQA (Bisk et al., 2020) contains multiple-choice questions that test knowledge about the physical world with two answer options.

CANONICAL:

- When-boiling-butter-,-when-it-'s-ready-,-you-can-Pour-it-into-a-jar-
- To-permanently-attach-metal-legs-to-a-chair-,-you-can-Weld-the-metal-together-to-get-it-to-stay-firmly-in-place-
- how-do-you-indent-something-?-leave-a-space-before-starting-the-writing-

SOCIALIQA contains multiple-choice questions that test social commonsense knowledge with three answer options.

CANONICAL:

- Choose-the-most-likely-answer-.-
  Context-:-After-weeks-apart-from-one-another-Austin-invited-their-boyfriend-to-dinner-.-
  Question-:-Why-did-Austin-do-this-?-
  reconnect-
- Choose-the-most-likely-answer-.-
  Context-:-Lee-spent-extra-time-to-explain-clearly-the-topic-purpose-to-the-students-.-
  Question-:-How-would-you-describe-Lee-?-
  careful-
- Choose-the-most-likely-answer-.-
  Context-:-The-man-and-woman-were-too-tired-to-cook-,-so-Bailey-made-their-breakfast-.-
  Question-:-How-would-Bailey-feel-afterwards-?-
  helpful-

F PSEUDOCODES

Algorithm 1 Greedy adversarial tokenization (radius-2 neighbourhood)

Require: string $\boldsymbol{x}$, correct option y, scoring function $z_c(\boldsymbol{v})$, start tokenization $\boldsymbol{v}_0 \in \mathcal{T}_{\mathcal{V}}(\boldsymbol{x})$, max steps S

$\boldsymbol{v} \leftarrow \boldsymbol{v}_0$
for $s = 1, \ldots, S$ **do**
  $\mathcal{N} \leftarrow \mathrm{Ne}(\boldsymbol{v}) = \mathcal{T}^2_{\mathcal{V}}(\boldsymbol{x}, \boldsymbol{v})$
  $\boldsymbol{v}^\star \leftarrow \arg\max_{\boldsymbol{u} \in \mathcal{N}} \left(\max_{c \neq y} z_c(\boldsymbol{u}) - z_y(\boldsymbol{u})\right)$
  if $\max_{c \neq y} z_c(\boldsymbol{v}^\star) - z_y(\boldsymbol{v}^\star) \leq \max_{c \neq y} z_c(\boldsymbol{v}) - z_y(\boldsymbol{v})$ **then**
    break ▷ local optimum
  end if
  $\boldsymbol{v} \leftarrow \boldsymbol{v}^\star$
end for
return $\boldsymbol{v}$

The expected split rate α controls the split budget relative to the canonical sequence length $m = |\boldsymbol{v}^c|$. In our experiments, we use `bernoulli` sampling which satisfies $\mathbb{E}[K] = \alpha m$.

Algorithm 2 STOCHASTOK: stochastic tokenization via random pairwise splits

Require: String $\boldsymbol{x} \in \mathcal{X}^*$; canonical tokenizer τ; split map $\mathcal{S}$; expansion parameter $\alpha \geq 0$; split mode $b \in \{\texttt{ceil}, \texttt{bernoulli}\}$

$\boldsymbol{v}^c \leftarrow \tau(\boldsymbol{x})$ ▷ canonical tokenization
$\boldsymbol{v} \leftarrow \boldsymbol{v}^c$
$K \leftarrow \textsc{SampleBudget}(\alpha, |\boldsymbol{v}^c|, b)$
$k \leftarrow 0$
while $k < K$ **do**
  $\mathcal{I} \leftarrow \{i \in \{1, \ldots, |\boldsymbol{v}|\} : \mathcal{S}(\boldsymbol{v}_i) \neq \emptyset\}$
  if $\mathcal{I} = \emptyset$ **then**
    break
  end if
  $i \leftarrow \texttt{randomChoice}(\mathcal{I})$
  $t \leftarrow \boldsymbol{v}_i$
  $(t_1, t_2) \leftarrow \texttt{randomChoice}(\mathcal{S}(t))$
  $\boldsymbol{v} \leftarrow (v_1, \ldots, v_{i-1}, t_1, t_2, v_{i+1}, \ldots, v_{|\boldsymbol{v}|})$
  $k \leftarrow k + 1$
end while
return $\boldsymbol{v}$

Algorithm 3 SAMPLEBUDGET: split-budget selection

Require: Expansion parameter $\alpha \geq 0$; canonical length m; split mode b

if $b = \texttt{ceil}$ **then**
  $K \leftarrow \lceil \alpha m \rceil$
else if $b = \texttt{bernoulli}$ **then**
  $r \leftarrow \alpha$
  for $i = 1$ to m **do**
    $B_i \sim \mathrm{Bernoulli}(r)$
  end for
  $K \leftarrow \sum_{i=1}^{m} B_i$
end if
return K

Algorithm 4 UNIFORM-K: added-token MRMDD compilation

Require: String $\boldsymbol{x}$; upper split budget k; canonical tokenization $\boldsymbol{v}^c$
Ensure: MRMDD $\mathcal{M}_{0:k}$
Compile MDD $\mathcal{M}$ from $\boldsymbol{x}$
Compute canonical token boundaries from $\boldsymbol{v}^c$
Create $k+1$ copies $\mathcal{M}_0, \mathcal{M}_1, \ldots, \mathcal{M}_k$
for each edge $e = (i, j)$ in $\mathcal{M}$ **do**
    if e completes a canonical token **then**
        Add edges $(\mathcal{M}_\ell^{(i)}, \mathcal{M}_\ell^{(j)}), \forall \ell \in [0:k]$
    else
        Add edges $(\mathcal{M}_\ell^{(i)}, \mathcal{M}_{\ell-1}^{(j)}), \forall \ell \in [1:k]$
    end if
end for
Prune all paths that do not end at a terminal node in $\mathcal{M}_0$
return $\mathcal{M}_{0:k}$

Algorithm 5 STOCHASTOK-UNI: stochastic tokenization with uniform fixed-count token segmentation

Require: String $\boldsymbol{x}$; canonical tokenizer τ; vocabulary $\mathcal{V}$; expansion parameter α; split mode b
$\boldsymbol{v}^c \leftarrow \tau(\boldsymbol{x})$
$m \leftarrow |\boldsymbol{v}^c|$
$K \leftarrow \textsc{SampleBudget}(\alpha, m, b)$
$\boldsymbol{p} \sim \mathrm{Dirichlet}(1, \ldots, 1)$
$(S_1, \ldots, S_m) \sim \mathrm{Multinomial}(K, \boldsymbol{p})$
for $i = 1$ to m **do**
    $w_i \leftarrow \textsc{TokenString}(v_i^c)$
    $M_i \leftarrow \textsc{MaxFeasibleAddedTokens}(w_i, \mathcal{V})$
end for
$(S_1, \ldots, S_m) \leftarrow \textsc{RedistributeOverflow}((S_i)_i, (M_i)_i, \boldsymbol{p})$ ▷ Cap infeasible counts and reassign leftover split budget to tokens with remaining capacity, sampled proportionally to $\boldsymbol{p}$.
for $i = 1$ to m **do**
    if $S_i = 0$ **then**
        $\tilde{\boldsymbol{v}}_i \leftarrow (v_i^c)$
    else
        $\tilde{\boldsymbol{v}}_i \leftarrow \textsc{UniformTokenSegmentation}(w_i, \mathcal{V}, S_i + 1)$
        if $\tilde{\boldsymbol{v}}_i = \emptyset$ **then**
            $\tilde{\boldsymbol{v}}_i \leftarrow (v_i^c)$
        end if
    end if
end for
return $(\tilde{\boldsymbol{v}}_1, \ldots, \tilde{\boldsymbol{v}}_m)$

Algorithm 6 UNIFORMTOKENSEGMENTATION: uniform token segmentation with a fixed segment count

Require: Token string w; vocabulary $\mathcal{V}$; number of segments r

function COUNTSEGMENTS($start, r$)
  if $start = |w|$ **then**
    return $\mathbb{1}[r = 0]$
  end if
  if $r = 0$ **then**
    return 0
  end if
  $total \leftarrow 0$
  for $end = start + 1$ to $|w|$ **do**
    $u \leftarrow w[start : end]$
    if $u \in \mathcal{V}$ **then**
      $total \leftarrow total +$ COUNTSEGMENTS($end, r - 1$)
    end if
  end for
  return $total$
end function

function BUILDSEGMENTS($start, r$)
  if $start = |w|$ **then**
    return $()$
  end if
  $\mathcal{U} \leftarrow []$, $\mathbf{q} \leftarrow []$
  for $end = start + 1$ to $|w|$ **do**
    $u \leftarrow w[start : end]$
    if $u \in \mathcal{V}$ **then**
      $q \leftarrow$ COUNTSEGMENTS($end, r - 1$)
      if $q > 0$ **then**
        append (u, end) to $\mathcal{U}$ and q to $\mathbf{q}$
      end if
    end if
  end for
  if $\mathcal{U} = \emptyset$ **then**
    return $\emptyset$
  end if
  $(u, end) \leftarrow$ `weightedRandomChoice`$(\mathcal{U}, \mathbf{q})$
  $\boldsymbol{z} \leftarrow$ BUILDSEGMENTS($end, r - 1$)
  return the sequence formed by prepending u to $\boldsymbol{z}$
end function

if COUNTSEGMENTS$(0, r) = 0$ **then**
  return $\emptyset$
else
  return BUILDSEGMENTS$(0, r)$
end if

G PROOF OF THE MAIN ROBUSTNESS THEOREM

Beyond the empirical results outlined in the previous section, we aim to theoretically characterize how tokenization affects the robustness of the resulting model. In this perspective, we focus on evasion attacks, *i.e.*, adversarial perturbations applied at test time. In this direction, let's consider a trained classifier $f\colon \mathcal{V}^* \to \mathcal{Y}$ and let $v \in \mathcal{V}^*$ be an input with its associated output $y \in \mathcal{Y}$, such that $f(v) = y$. The goal of an attacker is to craft a small additional perturbation to the input, such as to generate a point $\tilde{v}$ whose prediction $f(\tilde{v})$ is different from the original one. We note that in our setting, the function f represents the LLM *after* the tokenizer, comprising the embedding function and subsequent transformer blocks with self-attention.

As the generated adversarial perturbation should be "similar" to the original input, we consider a similarity budget ε (the maximum possible changes to the input), comprising the set of valid adversarial perturbations defined by a neighbourhood $\mathcal{B}(x, \varepsilon)$. In our setting $\mathcal{B}(x, \varepsilon) = \mathcal{T}_{\mathcal{V}}^{k}(x)$ with edit distance at most k. Now, the adversarial aim is to find, within $\mathcal{B}(x, \varepsilon)$, points that not only satisfy the adversarial aim of flipping the classification but also result in the worst prediction. In this direction, given our model f, the model's *adversarial risk* can be formulated as:

$$\mathcal{R}_{\varepsilon}[f] = \mathop{\mathbb{E}}_{x \in \mathcal{D}_{\mathcal{X}}} \left[\sup_{\tilde{v} \in \mathcal{B}(x,\varepsilon)} d_{\mathcal{Y}} \left(f\left(\tilde{v}\right), f\left(\tau(x)\right)\right)\right]. \tag{12}$$

From a defence perspective, the objective is to ensure that the risk remains small, thereby keeping predictions within the same output region and making it harder to find a valid perturbation. This goal can be formulated as:

Definition 2 (Adversarial Robustness)**.** *The classifier f is said to be* (ε, γ)-robust *if its adversarial risk with respect to the classifier f satisfies: $\mathcal{R}_{\varepsilon}[f] \leq \gamma$.*

Definition 2 uses an upper-bound perspective (γ) as computing the exact risk is generally intractable. A smaller value of γ indicates stronger robustness guarantees. Consequently, comparing the resulting bounds provides a principled way to assess and contrast the model's robustness. We consider a 1-layer Transformer-based model in which all activation functions are assumed to be 1-Lipschitz, which is the case for many common activations, *e.g.*, `tanh` and `sigmoid` (Virmaux & Scaman, 2018). We additionally assume the output of the embedding function to be bounded in terms of its norm (*e.g.*, $[0, 1]^{n \times d}$, with n being the number of tokens and d the hidden dimension). In this setting, the model's underlying robustness, γ, can be derived and is given by the following.

Theorem G.1. *Let $f\colon \mathcal{V}^* \to \mathcal{Y}$ be a classifier following our considered problem setup. Then f is* (ε, γ)-robust*, with:*

$$\gamma = 2\|W\| \left(\frac{d}{d-1}\right)^2 C_1 C_2 \sqrt{k},$$

$$\textit{with } C_1 = 1 + \|W_O\|\sqrt{H} \max_h \left[\|W^{V,h}\| \left[4\frac{n}{\sqrt{d/H}}\|W^{Q,h}\|\|W^{K,h}\|+1\right]\right],$$
$$C_2 = \left(1 + \|W_{FFN}\|\right)\|W_{\textit{out}}\|.$$

This bound itself does not distinguish between canonical training and the different stochastic tokenization schemes considered in this work. Instead, it characterizes how the perturbation neighbourhood and model sensitivity affect adversarial risk. A stochastic tokenization scheme with incomplete support does not expose the model to all tokenizations within this neighbourhood during training, potentially leaving it vulnerable to uncovered perturbations. Theorem G.1 characterizes how the perturbation neighbourhood and model sensitivity enter the resulting adversarial-risk bound. In Sec. I.2.7, we empirically connect stochastic tokenization to reduced local sensitivity.

Proof. We recall our considered problem setup, in which f is a classifier (a large-language model). In the following analysis, we will ignore the reliance on the tokenizer τ and focus on the remaining function, *i.e.*,

$$f = A \circ E, \tag{13}$$

where:

- $E : \mathcal{V}^* \to \mathbb{R}^{d \times n}$ an embedding function, mapping from tokens to a continuous space.
- $A : \mathbb{R}^{d \times n} \to \mathbb{R}^m$ an attention-block, mapping from a set of tokens to a classification, for instance. Note that we consider the pooling operation within this block.

Now recall that $\mathcal{X}$ denotes the space of raw inputs (*e.g.*, strings or words), $\mathcal{V}$ is the token vocabulary, and let $\mathcal{V}^*$ be the set of finite token sequences where n is the length of the sequence, which depends on the input's length and the considered tokenization. Further, let d be the embedding dimension.

We now consider that each space within our components is equipped with an appropriate metric or distance, *i.e.*, $d_{\mathcal{V}^*}$ (edit distance) on token sequences, and the L^2 norm in Euclidean spaces.

Given a character sequence $\boldsymbol{x}$, we denote:

$$s_1 \in \mathcal{V}^*, \quad s_2 \in \mathcal{T}^k_{\mathcal{V}^*}(\boldsymbol{x}, s_1), \tag{14}$$

where s_1 is, for example, the canonical tokenization of a string, and s_2 denotes an adversarially perturbed counterpart within the k-neighbourhood of s_1. Consequently, the adversarial perturbation induces at most k token-level changes, which can be written as:

$$d_{\mathcal{V}^*}(s_1, s_2) \leq k. \tag{15}$$

The effect of the perturbation can first be shown within the embedding level, and we therefore start by writing the effect of this perturbation. Since tokenization perturbations may change the sequence length, we compare s_1 and s_2 under an edit-aligned padding. Let $H(s) \in \{0,1\}^{|\mathcal{V}| \times n}$ denote the padded one-hot representation of a token sequence $s = (v_1, \ldots, v_n)$, so that the embedding function can be written as

$$E(s) = WH(s), \tag{16}$$

where $W \in \mathbb{R}^{d \times |\mathcal{V}|}$ is the learned embedding matrix.

If s_2 is obtained from s_1 by at most k binary split operations, then at most $2k$ aligned columns differ. Each differing column replaces one one-hot vector by another one-hot vector, possibly involving the padding symbol, and therefore has ℓ_2-norm at most $\sqrt{2}$. We can consequently formulate the following:

$$\|H(s_1) - H(s_2)\|_F \leq 2\sqrt{k}. \tag{17}$$

And consequently, we can also derive the following:

$$\|E(s_1) - E(s_2)\| = \|W(H(s_1) - H(s_2))\| \leq \|W\| \, \|H(s_1) - H(s_2)\| \leq 2 \, \|W\| \, \sqrt{k}. \tag{18}$$

Now, let's consider the second part, which relates to the self-attention block. Let $z_e = E(s_1)$ and $\tilde{z}_e = E(s_2)$ be the embedded representations of the two tokenizations.

For a multi-head attention block with H heads, let $h \in \{1, \ldots, H\}$ index the heads. For each head h, $W^{Q,h}$, $W^{K,h}$, and $W^{V,h}$ denote the query, key, and value projection matrices, respectively, and W_O denotes the multi-head attention output projection. Further, W_{FFN} denotes the linear part, or an upper bound on the Lipschitz constant, of the feed-forward network, and W_{out} denotes the final linear prediction head.

Additionally, as specified in Section 6.1, we consider the output of the embedding function to be bounded in terms of its norm (*e.g.*, $[0,1]^{n \times d}$, with n being the number of tokens and d the hidden dimension). Consequently, based on the previous assumptions and following previous work (Ennadir et al., 2025), we have the following inequality:

$$\|A(z_e) - A(\tilde{z_e})\| \leq \|z_e - \tilde{z_e}\| \left(\frac{d}{d-1}\right)^2 C_1 C_2 \tag{19}$$

with

$$C_1 = 1 + \|W_O\| \sqrt{H} \max_h \left[\left\| W^{V,h} \right\| \left(4 \frac{n}{\sqrt{d/H}} \left\| W^{Q,h} \right\| \left\| W^{K,h} \right\| + 1 \right) \right] \tag{20}$$

$$C_2 = (1 + \left\| W_{FFN} \right\|) \|W_{\text{out}}\|. \tag{21}$$

Hence, after simple substitution we have:

$$\|f(s_1) - f(s_2)\| \leq \|A(E(s_1)) - A(E(s_2))\| \tag{22}$$

$$\leq 2\|W\| \left(\frac{d}{d-1}\right)^2 C_1 C_2 \sqrt{k}\,. \tag{23}$$

□

H DISTRIBUTIONAL PROPERTIES OF STOCHASTIC TOKENIZATIONS

Let $\boldsymbol{x} = (x_1, \ldots, x_s)$ be a string sequence and $\boldsymbol{v}^c = (v_1^c, \ldots, v_m^c)$ its canonical tokenization. We sample a non-canonical tokenization $\boldsymbol{v} \sim P(\cdot \mid \boldsymbol{x})$ from a stochastic tokenization scheme, where $\boldsymbol{v} = (v_1, \ldots, v_p) \in \mathcal{T}_{\mathcal{V}}(\boldsymbol{x})$. We define a per-token split-count vector $\boldsymbol{S} = (S_1, \ldots, S_m)$ that describes the number of splits induced in each canonical token. The split vector $\boldsymbol{S}$ is a random variable. In the following, we discuss the probability distribution of $\boldsymbol{S}$ for each of our considered stochastic tokenization schemes.

STOCHASTOK STOCHASTOK applies N token-level splits in an iterative sampling process (see Alg. 2). At each step, a token v_i is selected uniformly at random and replaced by a pair of subtokens $(t_1, t_2) \in \mathcal{V}^2$ with $t_1 \circ t_2 = v_1$. In the next step, a token from this extended token sequence is selected uniformly. Now, there are two possibilities to select a token that corresponds to the original token v_i. This procedure corresponds exactly to a Polya-urn model. Thus, the number of times each original token is split follows a *Dirichlet–multinomial distribution* with symmetrical concentration parameter

$$\boldsymbol{S} \sim \mathsf{DirMult}(N, \mathbf{1}). \tag{24}$$

STOCHASTOK-UNI We construct STOCHASTOK-UNI to follow the per-token split-count distribution $\boldsymbol{S}$ of STOCHASTOK. We draw $\boldsymbol{S} \sim \mathsf{DirMult}(N, \mathbf{1})$, and sample the tokenizations for each canonical token v_i with the specified split count S_i. Recall that each token $v_i^c \in \mathcal{V}$ corresponds to a character sequence, which we denote by $\tilde{\boldsymbol{x}}_i \coloneqq \kappa(v_i^c) \in \mathcal{X}^*$. Thus, we sample $\boldsymbol{v}_i \sim \mathsf{Unif}\left(\mathcal{T}_{\mathcal{V}}^{S_i+1}(v_i^c)\right)$.

UNIFORM-K UNIFORM-K selects a non-canonical tokenization uniformly at random from the set of all possible valid segmentations with exactly k added tokens (k induced splits), *i.e.*, $\boldsymbol{v} \sim \mathsf{Unif}(\{\boldsymbol{v} \in \mathcal{T}_{\mathcal{V}}(\boldsymbol{x}) : |\boldsymbol{v}| = |\boldsymbol{v}^c| + k,\ \boldsymbol{v} \neq \boldsymbol{v}^c\})$.

Next, we want to describe the distribution over the per-token split-count $\boldsymbol{S} = (S_1, \ldots, S_m)$ this scheme induces. To simplify, we consider only splits within canonical token boundaries. For a canonical token v_i^c, let $A_i(s) = |\mathcal{T}_{\mathcal{V}}^{s+1}(v_i^c)|$ denote the number of non-canonical tokenizations of v_i^c with exactly s splits. Using these counts, we construct a polynomial

$$G_i(x) \;=\; \sum_{s \geq 0} A_i(s)\, x^{\,s}. \tag{25}$$

The total number of tokenizations of $\boldsymbol{v}^c$ with exactly k splits can be computed via the product

$$G(x) = \prod_{i=1}^{m} G_i(x) \tag{26}$$

via the k-th coefficient $[x^k]G(x)$. To obtain the probability of canonical token v_i^c having exactly $S_i = s$ splits, we define

$$H_i(x) \;=\; \prod_{j \neq i} G_j(x), \tag{27}$$

and obtain

$$\mathbb{P}\left(S_i = s \mid \sum_j S_j = k\right) = \frac{A_i(s)\,[x^{k-s}]\,H_i(x)}{[x^k]\,G(x)}\,. \tag{28}$$

UNIFORM UNIFORM samples tokenizations uniformly from all valid non-canonical tokenizations, $\boldsymbol{v} \sim \mathsf{Unif}(\mathcal{T}_{\mathcal{V}}(\boldsymbol{x}))$. For simplification, assume an infinite vocabulary $\mathcal{V}$. Let S_i be the number of splits within v_i^c. Then, S_i follows a *binomial* distribution

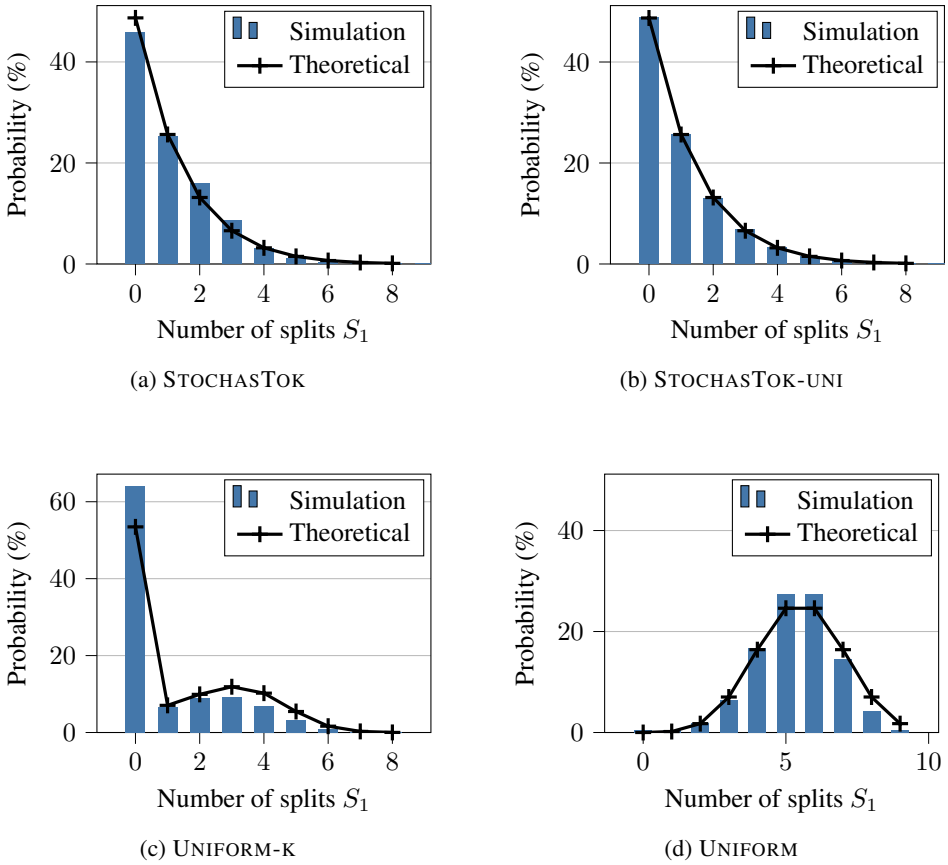
$$S_i \sim \mathsf{Binom}(|\kappa(v_i^c)| - 1, 1/2) \tag{29}$$

as UNIFORM corresponds to flipping a fair coin at every internal boundary of the string $\kappa(v_i^c)$.

H.1 EMPIRICAL DISTRIBUTION OVER PER-TOKEN SPLIT-COUNTS

We test whether our theoretical distributions for the per-token split-count distributions match the empirical distribution for the different stochastic tokenization schemes.

We take an example input sequence $\boldsymbol{x}$ =(`revolution is a rapid, fundamental transformation of a society's class, state, ethnic or religious structures`). We sample $D = 50,000$ draws of tokenizations $\boldsymbol{v} \sim P(\cdot \mid \boldsymbol{x})$ and report the empirical distribution over the number of splits S_1 for the first token $v_1 =$ `revolution` in Fig. 7.



(a) STOCHASTOK (b) STOCHASTOK-UNI

(c) UNIFORM-K (d) UNIFORM

Figure 7: Empirical and theoretical per-token split-count distributions for STOCHASTOK, STOCHASTOK-UNI, UNIFORM-K, and UNIFORM.

H.2 Different tokenizers and their effect on stochastic tokenization schemes

GPT2's BPE tokenizer (Sennrich et al., 2016) has a vocabulary of 50,257 tokens, whereas Llama3's tokenizer has a vocabulary size of 128k. The Llama3 tokenizer is BPE based with the difference that it ignores merge rules if a string is part of the vocabulary.

The histogram plots in Figure 8 and 9 demonstrate that the underlying tokenizer affects the size and structure of the non-canonical tokenization space. The larger vocabulary of Llama provides more valid alternative segmentations for the example considered. However, the tokenizers also differ in their vocabularies and merge rules, which in turn affect the resulting non-canonical tokenizations. Note that the vocabulary size can influence the useful range of α. In all our experiments, α controls the number of additional tokens relative to the canonical sequence length. A larger-vocabulary tokenizer may produce a shorter canonical sequence containing longer tokens. Such a tokenizer may benefit from a larger α to make a comparable portion of the tokenization space reachable. However, the ideal choice of α not only depends on vocabulary size, but also on token lengths and their available decompositions. In addition, the number of stochastic tokenization samples per input determines how well we cover the support of possible tokenizations, at the cost of increasing the effective dataset size.

GPT2

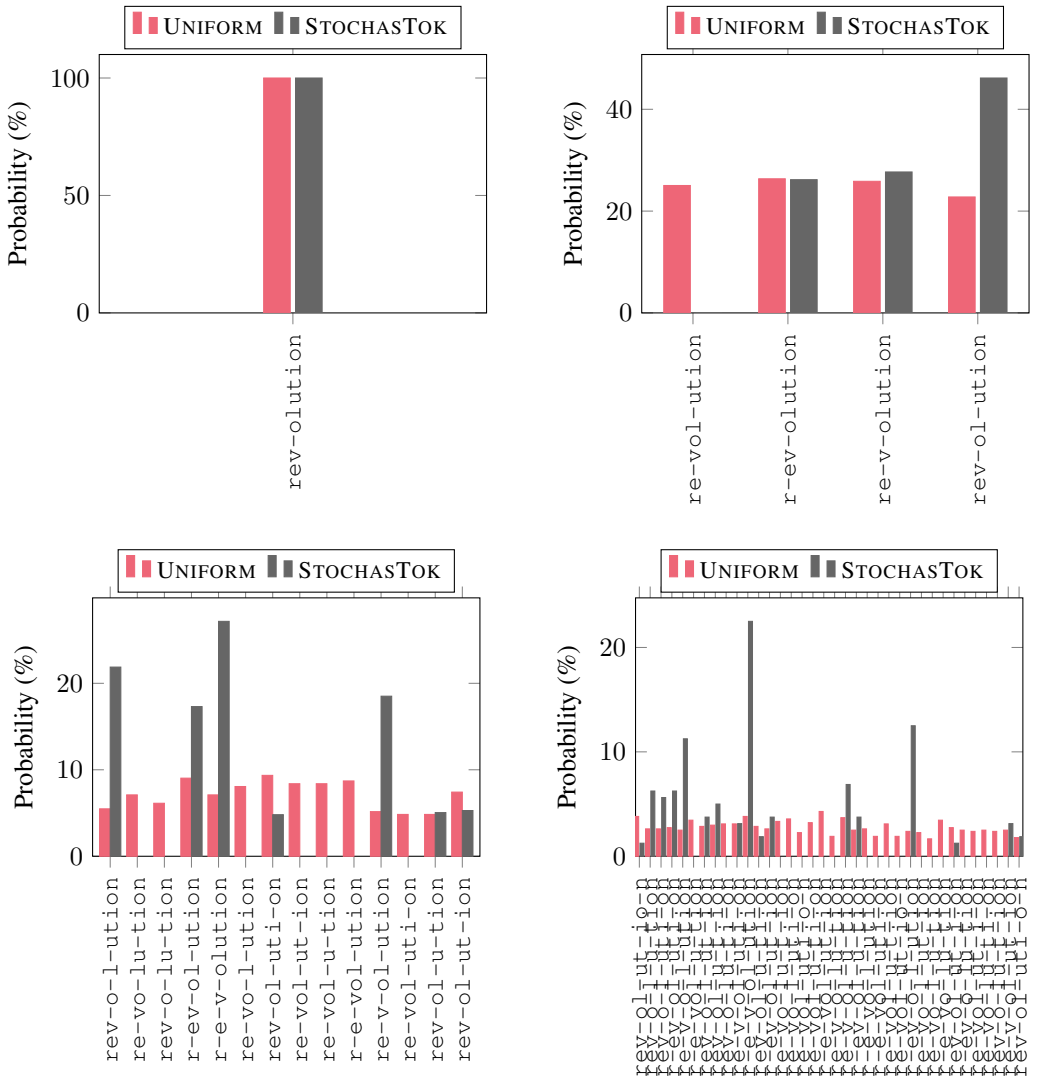


Figure 8: Histogram comparison of segmentation probabilities for two sampling schemes (UNIFORM vs. STOCHASTOK).

Llama 3

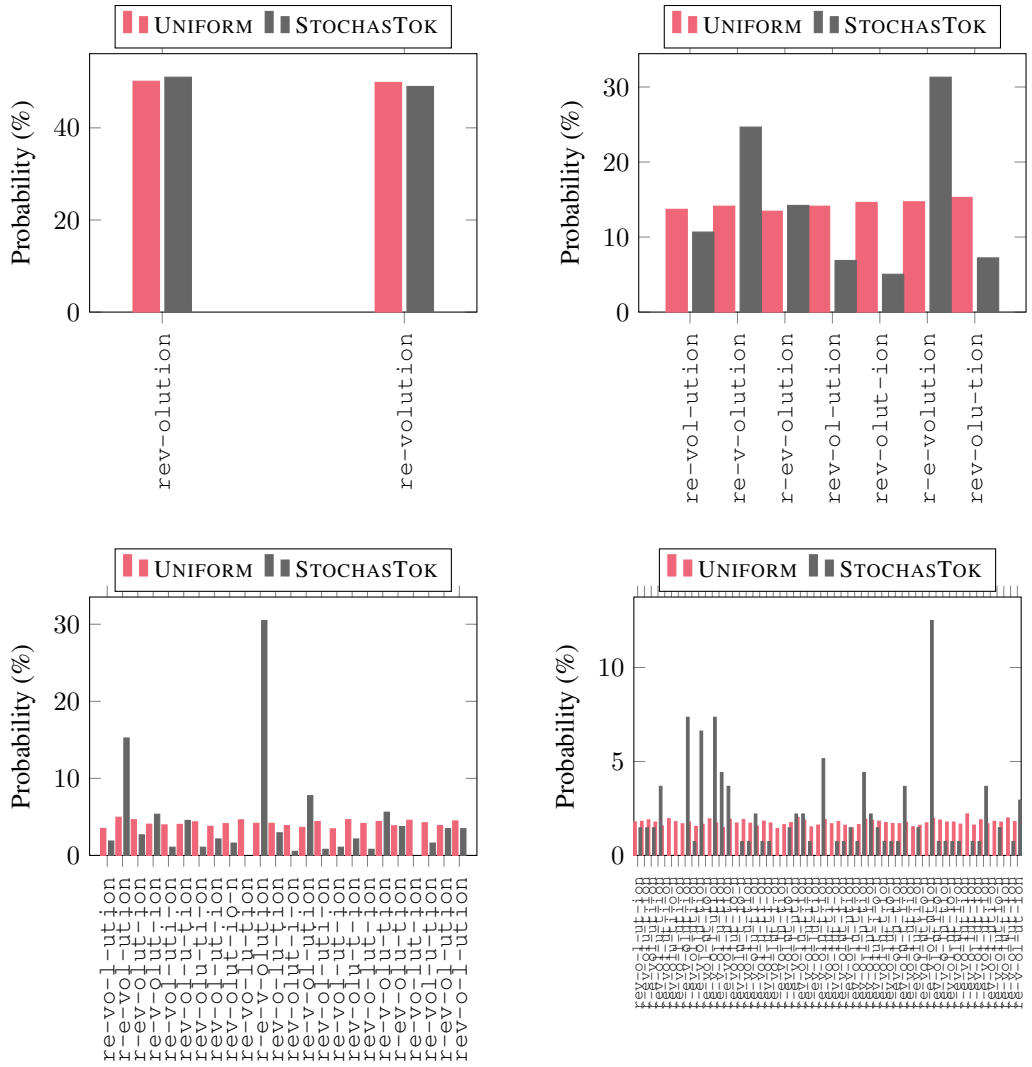


Figure 9: Histogram comparison of segmentation probabilities for two sampling schemes (UNIFORM vs. STOCHASTOK).

I ADDITIONAL EXPERIMENTS AND ABLATIONS

I.1 ADDITIONAL RESULTS FOR STOCHASTIC PRE-TRAINING

We report perplexity $\mathrm{PPL}(x_{1:T}) = \exp\left(-\frac{1}{T}\sum_{t=1}^{T}\log p(x_t \mid x_{<t})\right)$ on a validation subset of OpenWebText for our pre-trained **Tiny-LLM** models in Table 6. Stochastic pre-training leads to higher perplexity on *canonical* validation data. This is consistent with the training objective: during stochastic pre-training, the model is trained to also predict *non-canonical* token sequences. At the same time, perplexity is substantially reduced for non-canonically tokenized validation data compared to the canonical pre-trained model.

Table 6: Perplexity on an OpenWebText validation subset for canonically and non-canonically tokenized inputs.

α_{pre}	canonical	$\alpha_{\text{eval}} = 0.1$	$\alpha_{\text{eval}} = 0.5$
0.0	29.77	98.08	474.72
0.5	53.06	43.89	30.28

In addition, we report the multiple-choice accuracy of the **Tiny-LLM** models on standard benchmark datasets in Table 7. Increasing α_{pre} slightly worsens the fit to the canonical distribution, but improves robustness to non-canonical tokenizations. Importantly, during stochastic pre-training, the model is trained to predict non-canonical token sequences. This can reduce accuracy in standard MCQ evaluations, where we compare the log-likelihood of canonically tokenized answer continuations. In subsequent fine-tuning stages, however, we can perturb the input while keeping the prediction targets canonical. This encourages robustness to perturbed input tokenizations without requiring the model to generate non-canonical outputs.

Table 7: MCQ accuracy of pre-trained **Tiny-LLM** models on standard benchmarks under canonical and non-canonical tokenization at evaluation time.

	ARC-E		BLIMP		COPA		HELLASWAG	
α_{pre}	canon.	$\alpha_{\text{eval}} = 0.5$	canon.	$\alpha_{\text{eval}} = 0.5$	canon.	$\alpha_{\text{eval}} = 0.5$	canon.	$\alpha_{\text{eval}} = 0.5$
0.0	0.3780	0.2720	0.8480	0.6900	0.6600	0.4900	0.3160	0.2740
0.5	0.3300	0.3120	0.8300	0.8040	0.6100	0.5900	0.2960	0.3060

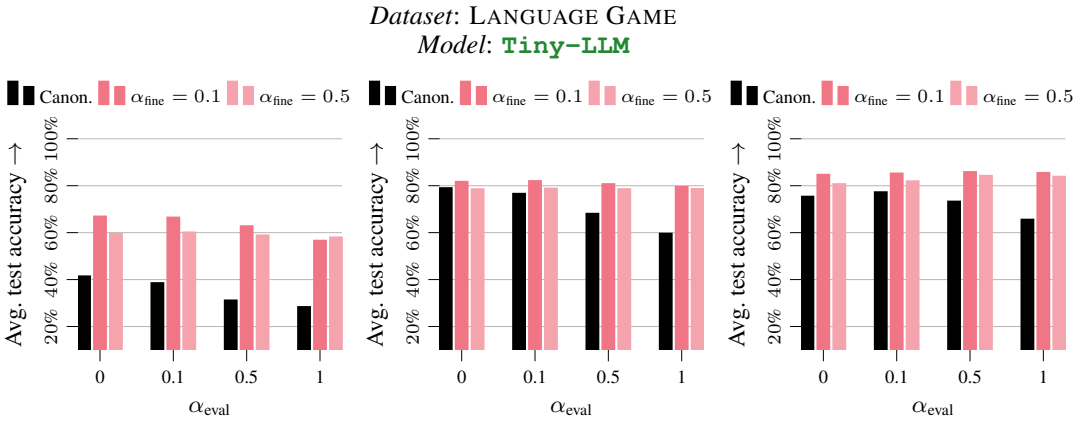
Lastly, we evaluate the **Tiny-LLM** models on TOKSUITE (Altıntaş et al., 2025), which probes a variety of tokenization-sensitive tasks. Under *canonical* evaluation, stochastic pre-training with $\alpha_{\text{pre}} = 0.5$ leads to a small drop in overall accuracy compared to canonical pre-training, from 0.3746 to 0.3623. For individual subsets, the largest decreases are observed for CONTRACTIONS (from 0.5854 to 0.4146), ABBREVIATIONS (from 0.3784 to 0.2703), WEB SEARCH QUERIES (from 0.3514 to 0.2432), SIMILAR WORDS (from 0.5405 to 0.4595), and COLLOQUIAL VARIANTS (from 0.4250 to 0.3500). However, we also observe improvements on several subsets, including SPELLED-OUT TEXT (from 0.2667 to 0.4667), HISTORICAL SPELLING (from 0.3500 to 0.5000), KEYBOARD PROXIMITY ERRORS (from 0.2927 to 0.4146), and DATE FORMATS (from 0.1667 to 0.2778).

Under *stochastic* evaluation with STOCHASTOK perturbations at $\alpha_{\text{eval}} = 0.5$, however, stochastic pre-training improves overall accuracy from 0.2974 to 0.3535. The largest gains are observed for CHARACTER SUBSTITUTION (from 0.2500 to 0.4375), HISTORICAL SPELLING (from 0.3000 to 0.4500), OCR ERRORS (from 0.2381 to 0.3810), ORTHOGRAPHIC ERRORS (from 0.3250 to 0.4375), SPELLED-OUT TEXT (from 0.4000 to 0.5333), and SPACED STYLING (from 0.1000 to 0.2250). We still observe decreases on a few subsets, such as COLLOQUIAL VARIANTS (from 0.4250 to 0.3250), ABBREVIATIONS (from 0.3243 to 0.2973), and SCRIPTED TEXT (from 0.2750 to 0.2250).

Overall, this is consistent with our findings on the other benchmarks: stochastic pre-training slightly worsens the fit to the canonical distribution, but improves robustness to non-canonical tokenizations.

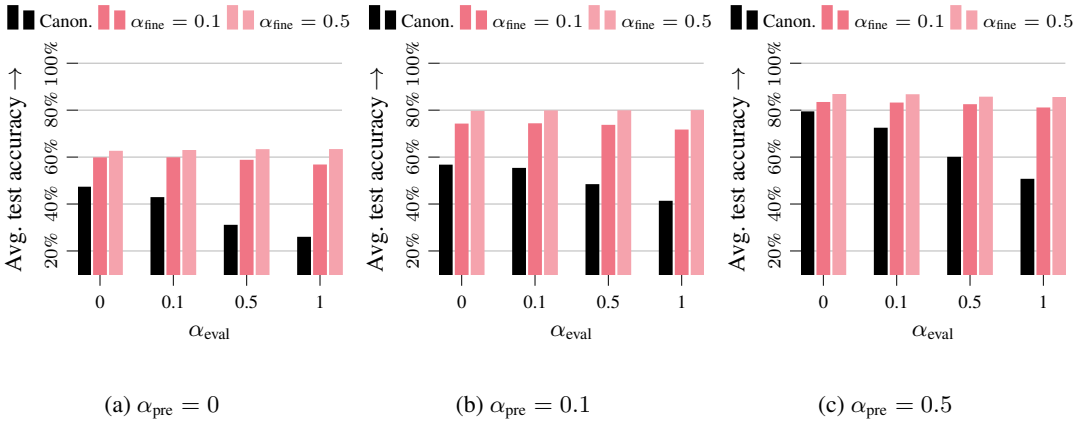
I.2 Additional results for stochastic fine-tuning

I.2.1 Additional results for Tiny-LLM



(a) $\alpha_{\text{pre}} = 0$ (b) $\alpha_{\text{pre}} = 0.1$ (c) $\alpha_{\text{pre}} = 0.5$

Figure 10: Average test accuracy of Tiny-LLM on Language Game under stochastic tokenization during fine-tuning, for different levels of stochasticity during pre-training. Stochastic fine-tuning improves robustness to random perturbations during testing across pre-training settings.

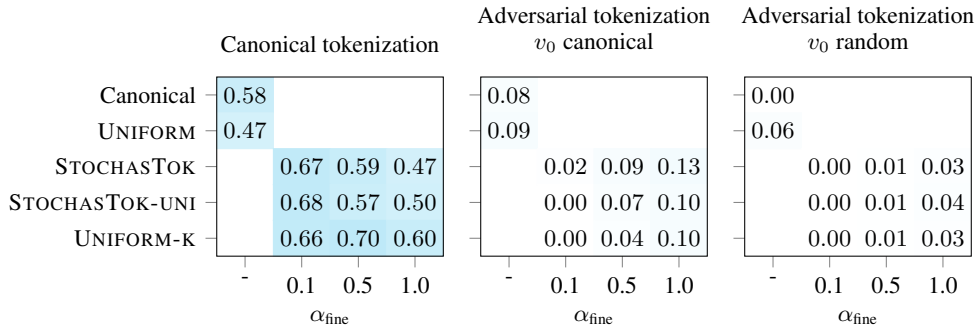


(a) $\alpha_{\text{pre}} = 0$ (b) $\alpha_{\text{pre}} = 0.1$ (c) $\alpha_{\text{pre}} = 0.5$

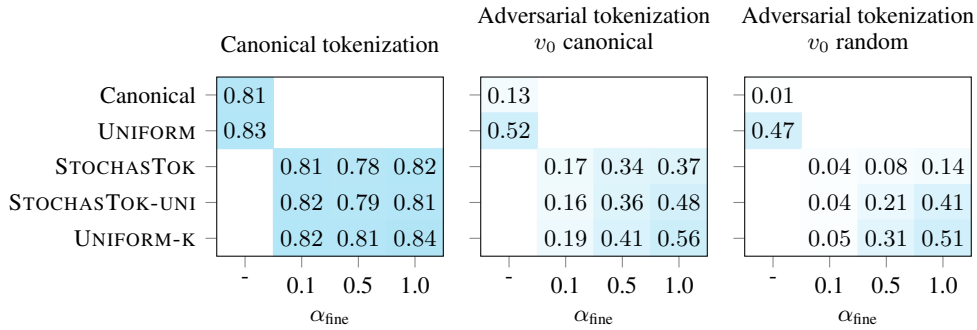
Figure 11: Average test accuracy of Tiny-LLM on Cute under stochastic tokenization during fine-tuning, for different levels of stochasticity during pre-training. Stochastic fine-tuning improves robustness to random perturbations during testing across pre-training settings.

Dataset: LANGUAGE GAME
Model: `Tiny-LLM`

Pre-training perturbation: $\alpha_{\text{pre}} = 0$



Pre-training perturbation: $\alpha_{\text{pre}} = 0.1$



Pre-training perturbation: $\alpha_{\text{pre}} = 0.5$

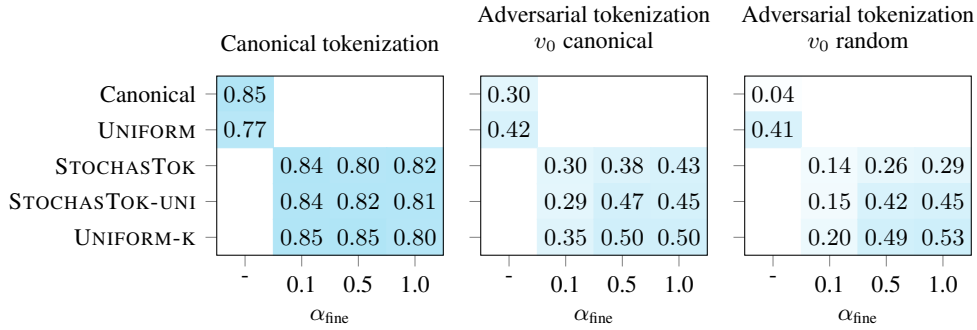
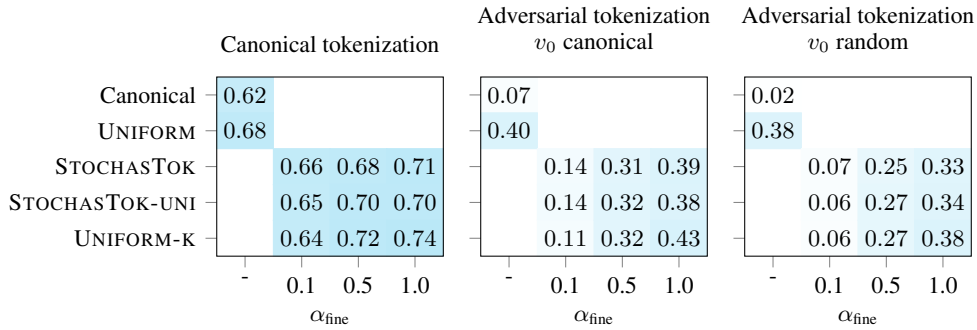


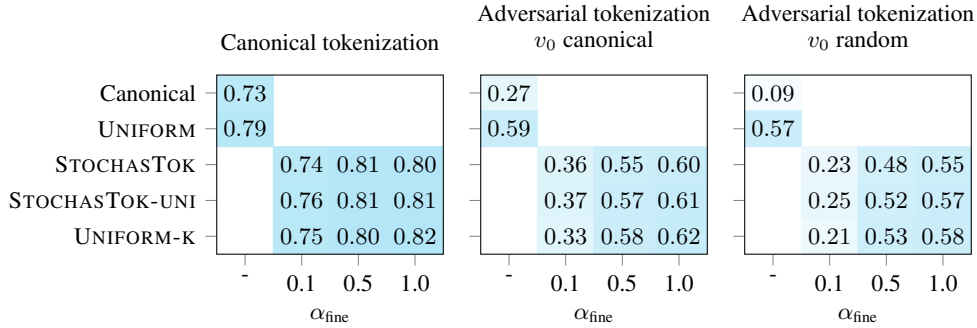
Figure 12: Effect of perturbation strength α_{pre}, α_{fine}, and different training schemes on adversarial robustness for the LANGUAGE GAME data set.

Dataset: CUTE
Model: `Tiny-LLM`

Pre-training perturbation: $\alpha_{\text{pre}} = 0$



Pre-training perturbation: $\alpha_{\text{pre}} = 0.1$



Pre-training perturbation: $\alpha_{\text{pre}} = 0.5$

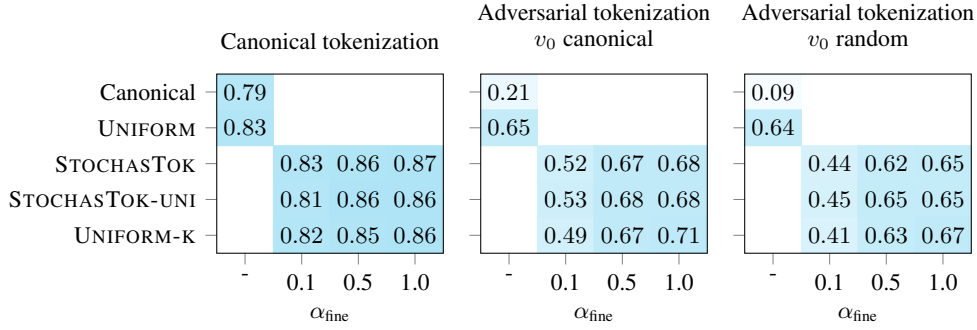
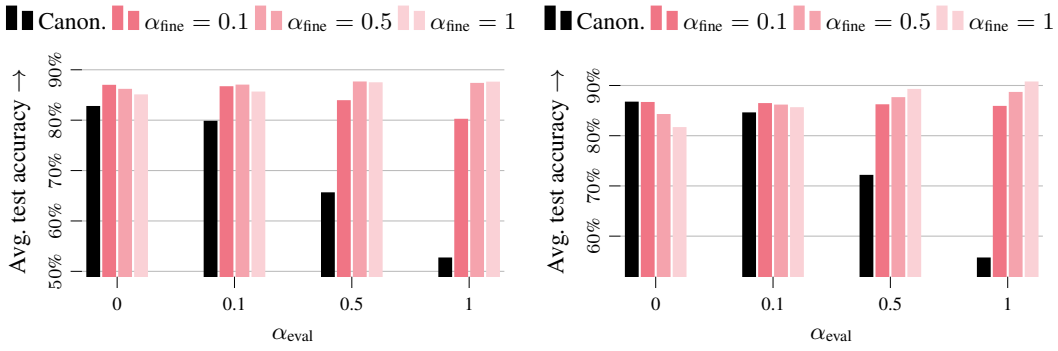


Figure 13: Effect of perturbation strength α_{pre}, α_{fine}, and different training schemes on adversarial robustness for the CUTE data set.

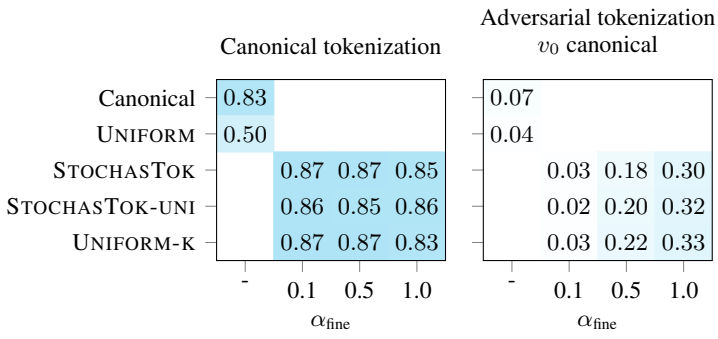
I.2.2 ADDITIONAL RESULTS FOR GPT2-xl



(a) LANGUAGE GAME (b) CUTE

Figure 14: Average accuracy of GPT2-xl for stochastic tokenization during evaluation and fine-tuning on LANGUAGE GAME and CUTE. Stochastic fine-tuning improves robustness to random perturbations during evaluation.

Dataset: LANGUAGE GAME



Dataset: CUTE

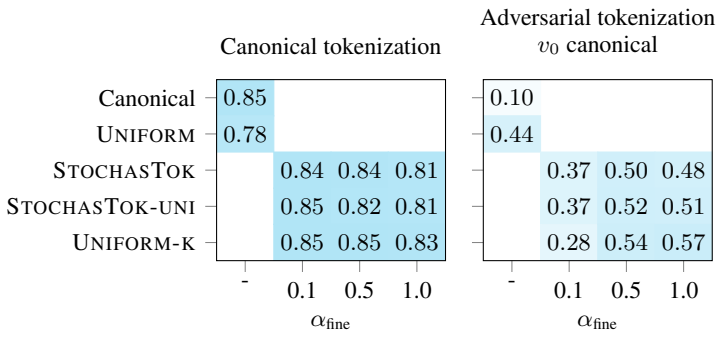
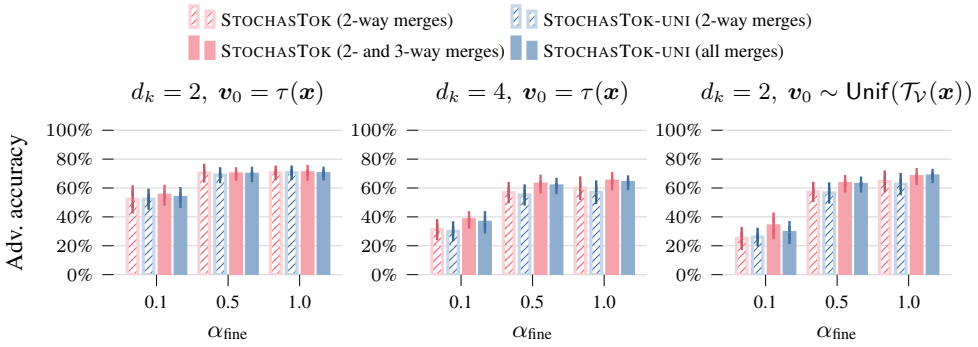


Figure 15: Accuracy of GPT2-xl under canonical and adversarial tokenization on LANGUAGE GAME and CUTE. Stochastic fine-tuning consistently improves robustness to perturbations during evaluation, with the highest gains for methods that address support and sampling bias (STOCHASTOK-UNI, UNIFORM-K).

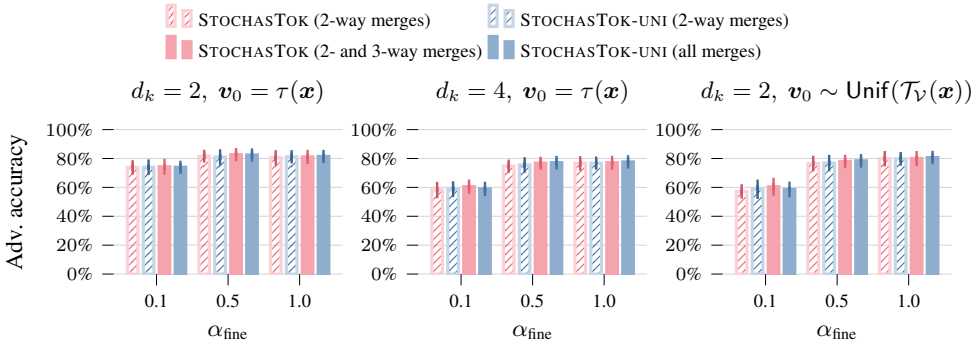
I.2.3 ABLATION ON TOKENIZATION SUPPORT

STOCHASTOK induces a distribution over tokenizations that is biased and with incomplete support. We showed that addressing both issues with STOCHASTOK-UNI results in increased robustness to adversarial tokenizations. In this ablation, we study which of the two extensions (uniformity or support) is the main driver of the improvements. Thus, we construct two additional stochastic tokenization schemes. On the one hand, we use the same STOCHASTOK algorithm as in Alg. 2, but build the dictionary of possible expansions with 2-way and 3-way merges. This still results in a biased distribution, but increases the support of possible tokenizations. On the other hand, we restrict STOCHASTOK-UNI to sample only tokenizations that are reachable with STOCHASTOK, *i.e.*, achievable via 2-way merges.

We test the robustness to adversarial tokenization attacks for three different attack settings. Each attack consists of 10 iterations. First, we vary the initial tokenization $\boldsymbol{v}_0$: canonical $\boldsymbol{v}_0 = \tau(\boldsymbol{x})$, or uniform $\boldsymbol{v}_0 \sim \mathsf{Unif}(\mathcal{T}_{\mathcal{V}}(\boldsymbol{x}))$. Second, we increase the neighbourhood of tokenizations for the greedy search in each iteration step by increasing the edit distance $d_k = \{2, 4\}$. Fig. 16a shows the results for 10 LoRA fine-tuned `Llama-1b` models on LANGUAGE GAME with our stochastic tokenization schemes for different expansion proportions α_{fine}. The first case ($d_k = 2,\ \boldsymbol{v}_0 = \tau(\boldsymbol{x})$) corresponds to the easiest setting. Within the tokenization neighbourhood with edit distance $d_k = 2$, all methods perform equally well. However, when increasing the neighbourhood to $d_k = 4$ *or* starting from a random initial tokenization, methods with increased support exhibit higher robustness. The same trend, albeit to a smaller extent, is visible for the CUTE dataset in Fig. 16b.



(a) LANGUAGE GAME dataset



(b) CUTE dataset

Figure 16: Adversarial accuracy (%) of 10 LoRA fine-tuned `Llama-1b` models versus fine-tuning stochasticity α_{fine} for three attack settings (defined by neighbourhood radius d_k and initial tokenization $\boldsymbol{v}_0$). Methods with larger support over $\mathcal{T}_{\mathcal{V}}(\boldsymbol{x})$ (STOCHASTOK (2- and 3-way merges), STOCHASTOK-UNI (all merges)) are more robust. Increasing uniformity on the same support (STOCHASTOK-UNI (2-way merges)) yields no substantial improvements.

I.2.4 Ablation on model size

To assess whether the observed robustness gains depend on model size, we repeat fine-tuning and adversarial evaluation for Llama-3 models with 1B, 3B, and 8B parameters. We compare canonical fine-tuning against StochasTok and Uniform-k fine-tuning with $\alpha_{\text{fine}} = 0.5$. Results on Language Game are shown in Table 8.

Table 8: Canonical and adversarial accuracy on Language Game across Llama-3 model sizes.

Training	α_{fine}	1B	3B	8B
Canonical	–	0.954 / 0.040	0.982 / 0.390	**0.994** / 0.602
StochasTok	0.5	0.956 / 0.596	0.984 / 0.874	0.990 / 0.926
Uniform-k	0.5	**0.968 / 0.806**	**0.986 / 0.916**	**0.994 / 0.958**

Across the evaluated model sizes, stochastic fine-tuning consistently improves adversarial robustness, with Uniform-k providing the highest adversarial accuracy. For the considered model sizes and the Language Game dataset, canonically fine-tuned models of the larger model sizes are less vulnerable to adversarial tokenization.

I.2.5 Tokenizer with higher vocabulary size

Next, we test whether stochastic tokenization also improves robustness for tokenizers with a larger vocabulary size. Specifically, we consider **`Gemma-1b`** with a vocabulary size of 262k entries. Figs. 17 and 18 summarize average and adversarial robustness, respectively, and show the same improvements as in our main experiments. The greedy search for the adversarial tokenization is performed for 10 iterations, $d_k = 2$ neighbours, starting from canonical initial tokenizations v_0. Stochastic tokenization improves robustness, with stochastic methods with broader support providing greater adversarial robustness for the same split budget.

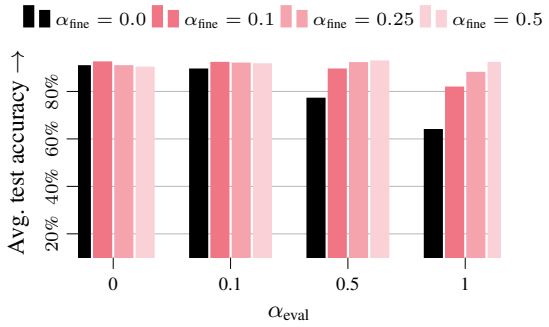


Figure 17: Average perturbation robustness of **`Gemma-1b`** fine-tuned and evaluated with StochasTok on Language Game.

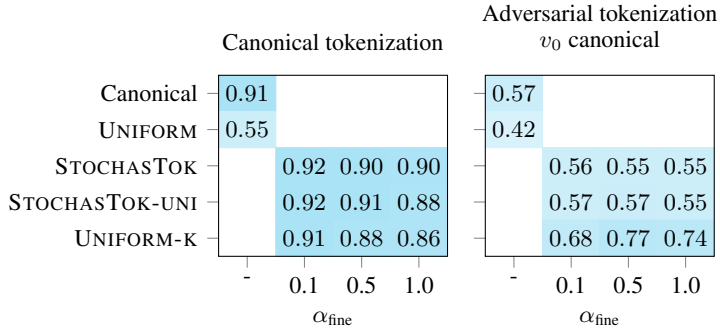


Figure 18: Canonical and adversarial accuracy of **`Gemma-1b`** on Language Game.

I.2.6 COMPARISON TO BPE-DROPOUT

BPE-dropout (Provilkov et al., 2020) is a popular stochastic tokenization method for BPE tokenizers. It introduces stochasticity in the tokenization process by randomly skipping merge rules with a specified dropout probability p_{drop}.

However, a direct comparison to our methods is not straightforward: The perturbation strength is controlled indirectly via the dropout rate p_{drop}, and the resulting number of induced splits further depends on the input string and tokenizer's merge structure. In contrast, the methods considered in the main text allow explicit control over the number of induced splits.

To illustrate this, we apply BPE-dropout to the example string from Sec. H.1, namely *"revolution is a rapid, fundamental transformation of a society's class, state, ethnic or religious structures"*. We sample 50,000 stochastic tokenizations with BPE-dropout using the Llama 3 tokenizer and track the resulting non-canonical segmentations. First, Table 9 reports the statistics of the total number of induced splits for different values of p_{drop}. The effective perturbation strength varies substantially even for a fixed value of p_{drop}. Second, Fig. 19 summarizes the distribution over the non-canonical tokenizations of the first token (*"revolution"*). This reflects the findings of (Cognetta et al., 2024) that BPE-dropout has a strong sampling bias, which may lead to certain non-canonical tokenizations being underrepresented.

Table 9: Induced number of splits under BPE-dropout for different dropout rates p_{drop}.

p_{drop}	mean	std	min	max
0.10	0.75	0.90	0	6
0.25	3.01	1.89	0	14
0.50	22.41	4.62	6	44
0.75	53.06	6.13	29	78

Next, we empirically evaluate the performance of BPE-dropout compared to our methods for **`Tiny-LLM`**, **`Llama-1b`**, and **`Qwen-0.6b`** fine-tuned on LANGUAGE GAME. Fig. 20 shows average accuracy under canonical and stochastic evaluation with STOCHASTOK tokenization. We observe that BPE-DROPOUT improves robustness to non-canonical tokenizations compared to canonical fine-tuning.

Fig. 21 reports robustness under adversarial tokenization. As noted above, a direct comparison between p_{drop} and α_{fine} is not possible. Thus, we determine the effective average number of splits for the methods, α_{eff}. Note that for our methods α_{eff} closely matches α_{fine} by construction. For BPE-DROPOUT, the effective number of splits varies significantly across the **`Tiny-LLM`**, **`Llama-1b`**, and **`Qwen-0.6b`** tokenizers for the same p_{drop}, and additionally depends on the string sequence.

The greedy search for the adversarial tokenization is performed for 10 iterations, $d_k = 2$ neighbours, starting from canonical initial tokenizations v_0. Here, BPE-dropout again improves robustness over canonical fine-tuning, while more uniform stochastic tokenization schemes remain competitive or stronger at comparable perturbation strengths.

Llama 3

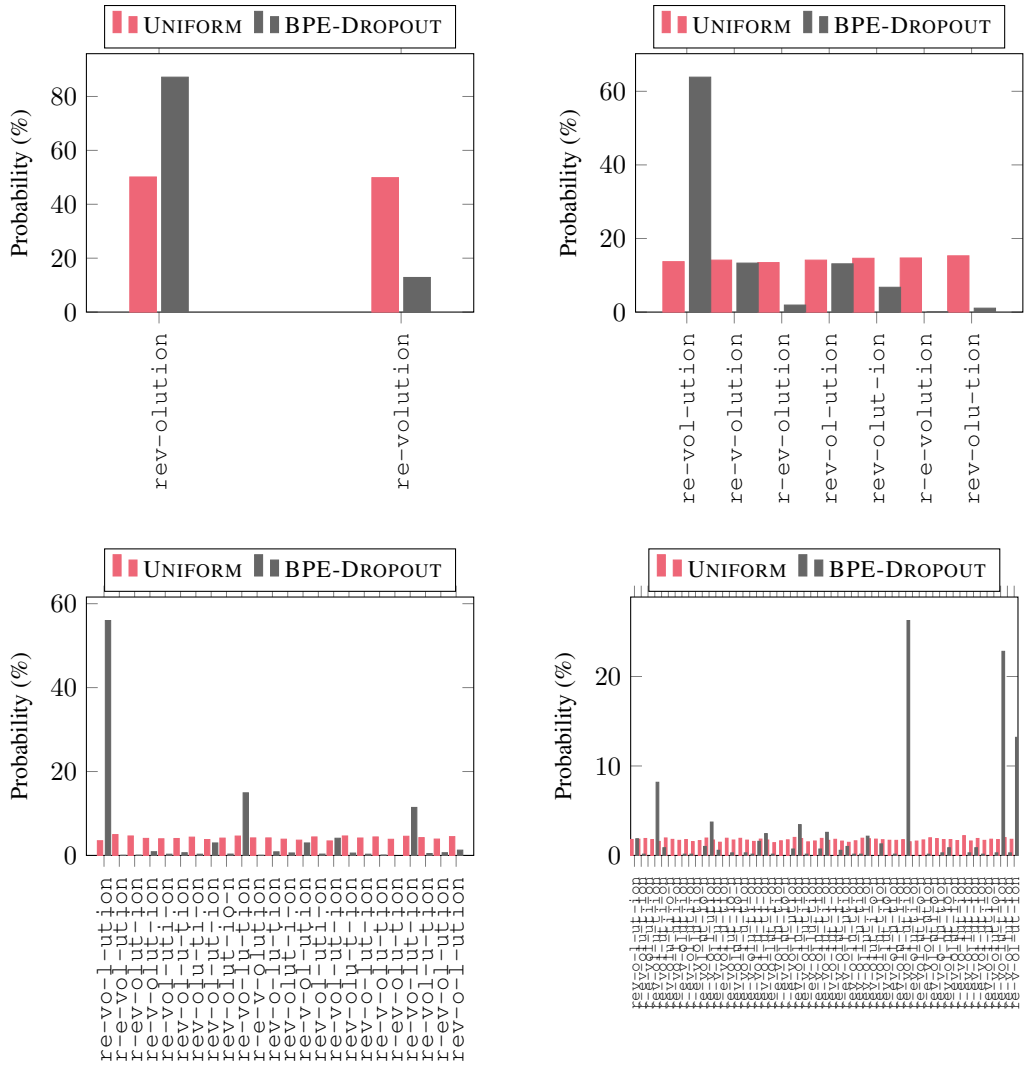
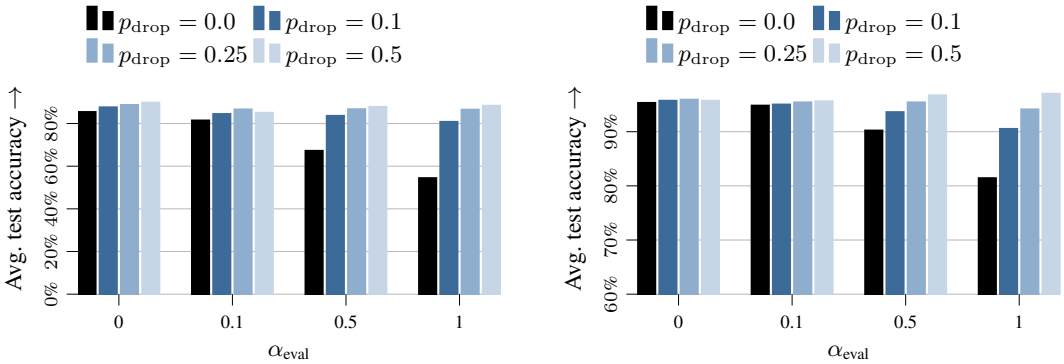
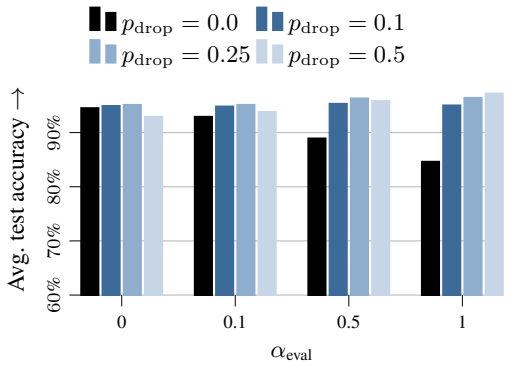


Figure 19: Histogram comparison of segmentation probabilities for two sampling schemes (UNIFORM vs. BPE-DROPOUT with $p_{drop} = 0.5$).



(a) `Tiny-LLM`

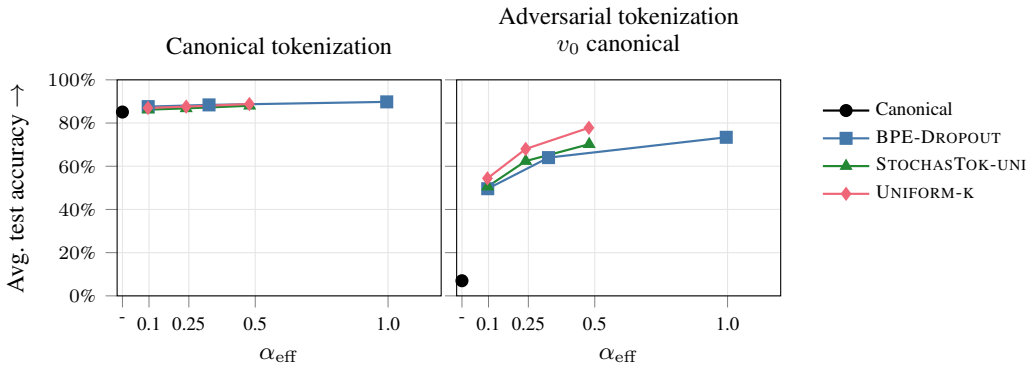
(b) `Llama-1b`



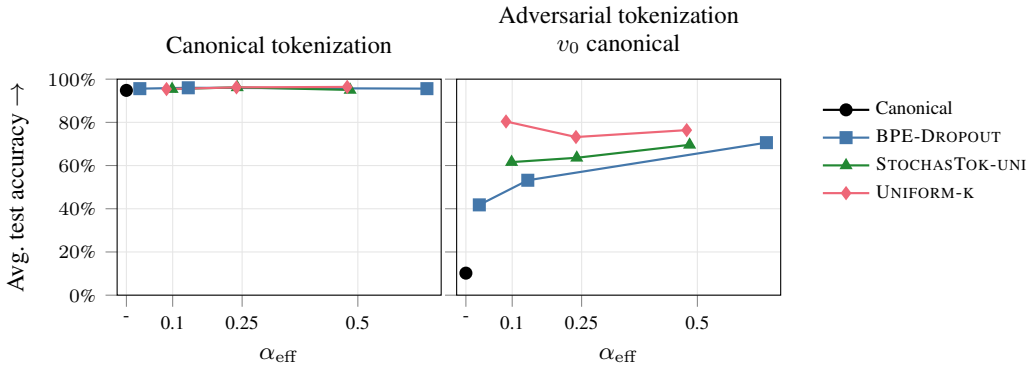
(c) `Qwen-0.6b`

Figure 20: Average accuracy of `Tiny-LLM`, `Llama-1b`, and `Qwen-0.6b` fine-tuned with BPE-DROPOUT and evaluated on LANGUAGE GAME under canonical and STOCHASTOK tokenization. BPE-DROPOUT improves robustness to non-canonical tokenizations. However, choosing an appropriate dropout rate p_{drop} is less intuitive, since the resulting number of splits depends on the underlying input text and its merge structure.

Model: **Tiny-LLM**



Model: **Llama-1b**



Model: **Qwen-0.6b**

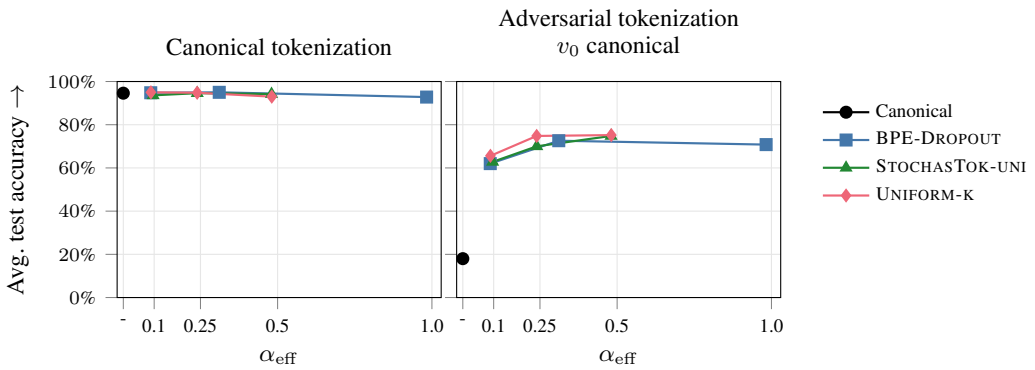


Figure 21: Accuracy of **Tiny-LLM**, **Llama-1b**, and **Qwen-0.6b** under canonical and adversarial tokenization on LANGUAGE GAME for stochastic fine-tuning with BPE-DROPOUT, STOCHASTOK-UNI, and UNIFORM-K. All stochastic fine-tuning methods improve robustness compared to canonical fine-tuning. Note that α_{fine} and p_{drop} cannot be compared directly as they induce different numbers of splits.

I.2.7 ABLATION ON EMBEDDING DISTANCES

We measure the mean normalized distance between the hidden representations of canonical and non-canonical tokenizations of the same word across layers. We use the 1,000 most frequent English words and uniformly sample 50 non-canonical tokenizations for each word. Fig. 22 shows the results for `Llama-1b` after canonical and stochastic fine-tuning on LANGUAGE GAME. While canonical fine-tuning does not reduce these distances compared to the zero-shot model, stochastic fine-tuning consistently yields smaller distances, especially in deeper layers. This indicates reduced local sensitivity to tokenization changes and is consistent with the improved robustness to adversarial tokenization observed in the main paper.

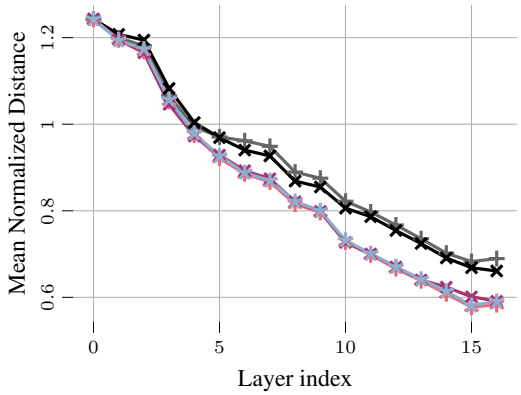


Figure 22: Canonical fine-tuning (—×—) does not affect the distances of alternative tokenizations compared to the zero-shot `Llama-1b` (—|—). Stochastic tokenization (STOCHASTOK —×—, STOCHASTOK-UNI —×—, UNIFORM-K —×—) reduces distances in deeper layers.

I.3 Additional results for in-context learning

I.3.1 Example prompt for Language Game

Canonical context + perturbed question ($\alpha_{query} = 1$).

```
Below- are- solved- examples-.- For- the- final-
question-,- predict- the- correct- option.-

Which- choice- has- the- most- letter- '-b-'s-?- These-
are- the- possible- option- words-:- [- up-,- baby-,-
rain-,- glad-].- Answer-:- - baby-

Which-option-ends-with-'o'-?-The-possible-option-
strings-:-[-no-,-hard-,-by-,-happen-].-Answer-:--
no-

What-word-is-the-shortest-?-The-available-option-
strings-are-:-[-row-,-suffix-,-gather-,-energy-].-
Answer-:--row-

T-h-e-possible-option-words-a-re-:-[-straight-,-ca-
r-r-y-,-di-ffer-,-h-alf-].-Which-of-the-p-ossible-
option-words-has-the-most-let-te-r-'-t-'s-?-Ans-we-
r-:-
```

Perturbed context ($\alpha_{context} = 1$) + perturbed question ($\alpha_{query} = 1$).

```
Below-are-solved-examples-.-For-the-final-question-,-
predict-the-correct-option.-

Wh-i-ch-choice-has-the-most-le-t-t-er-'-b-'s-?-
These-are-the-possible-option-words-:-[-up-,-baby-
,-rain-,-g-lad-].-A-n-sw-er-:--baby-

Which-option-end-s-with-'o'-?-The-possible-option-s-
tring-s-:-[-no-,-hard-,-b-y-,-happen-].-Answer-:--
no-

Wh-at-word-i-s-the-shortest-?-The-available-option-
string-s-are-:-[-r-o-w-,-suffix-,-g-ather-,-e-n-
ergy-].-Ans-wer-:--row-

T-h-e-possible-option-words-a-re-:-[-straight-,-ca-
r-r-y-,-di-ffer-,-h-alf-].-Which-of-the-p-ossible-
option-words-has-the-most-let-te-r-'-t-'s-?-Ans-we-
r-:-
```

Table 10: Accuracy (%) of **Llama-8b** under canonical versus uniformly sampled query tokenization across datasets, with $K = 10$ few-shot examples.

Context tokenization	α_{ICL}	LANGUAGE GAME		ARC-E		COPA		CSQA		HELLASWAG		PIQA		SOCIALIQA	
		canon.	**unif.**	**canon.**	**unif.**	**canon.**	**unif.**	**canon.**	**unif.**	**canon.**	**unif.**	**canon.**	**unif.**	**canon.**	**unif.**
Canonical	-	83.9	70.3	78.1	71.3	92.0	84.6	68.3	63.0	54.8	50.2	81.4	79.5	52.4	45.1
STOCHASTOK	0.1	84.6	75.6	76.7	71.7	92.0	86.0	68.8	63.4	55.0	50.9	81.7	79.6	52.8	46.3
STOCHASTOK	0.5	85.9	80.2	77.5	73.2	91.1	86.8	67.8	64.1	54.3	50.9	81.9	79.7	53.7	47.4
STOCHASTOK	1.0	87.7	81.8	77.4	74.0	91.3	87.0	67.1	63.9	54.4	50.9	81.9	80.0	53.8	47.5
STOCHASTOK-UNI	0.1	85.1	75.6	76.4	71.5	92.0	85.8	68.6	63.5	55.1	50.8	81.6	79.5	53.0	46.2
STOCHASTOK-UNI	0.5	86.8	80.4	77.1	73.1	91.1	87.0	67.4	63.7	54.2	50.8	82.0	79.8	53.6	47.3
STOCHASTOK-UNI	1.0	87.1	81.3	77.3	73.8	91.6	87.3	66.7	63.9	54.2	51.0	81.9	80.1	53.4	47.7
UNIFORM-K	0.1	85.1	74.5	77.3	71.5	92.0	85.2	68.7	63.4	54.8	50.5	81.4	79.5	52.7	46.2
UNIFORM-K	0.5	86.2	79.0	77.1	72.5	91.4	87.1	68.0	63.7	54.4	51.0	81.8	79.8	53.0	47.8
UNIFORM-K	1.0	87.2	80.8	77.1	73.7	91.1	87.2	67.6	64.0	54.3	50.9	81.9	80.0	52.8	48.9
UNIFORM	-	85.0	81.7	77.6	75.2	90.2	87.0	62.3	63.8	54.1	51.0	81.9	80.4	51.4	49.8

Table 11: Accuracy (%) on LANGUAGE GAME using **Llama-8b** under canonical and adversarial tokenizations, with 5 context examples.

	CANONICAL	STOCHASTOK	STOCHASTOK-UNI	UNIFORM-K	UNIFORM
Canonical	83.8	89.2	89.6	88.6	84.0
Adversarial	18.8	50.4	51.0	49.2	45.4