

Language Generation with Infinite Contamination

Anay Mehrotra
Yale University

Grigoris Velegkas
Google Research

Xifan Yu
Yale University

Felix Zhou
Yale University

Abstract

The remarkable success of large language models has led to a growing body of theoretical research aimed at understanding them. Here, a recent line of work studies language generation in the limit, a formal model of language learning where an algorithm observes an adversarially generated enumeration of strings from an unknown target language K and must eventually generate new, unseen strings from K . [KM24, NeurIPS] proved that generation is achievable in surprisingly general settings; whenever K belongs to a known countable collection of languages (even the collection of all Turing-enumerable languages). The generator of [KM24] suffers from “mode collapse:” it generates from an increasingly ever-smaller subset of the target. To address this, the recent work of [KW25, FOCS] formalized stronger notions of generation that require the generator’s output to be “dense” in the target language; informally, requiring it to asymptotically cover a positive fraction of the target language. They showed that generation with density, surprisingly, remains achievable for all countable collections.

However, both of these works rely on the crucial assumption of *perfect* data: in their model, the adversary can neither insert strings from outside the target language (*i.e.*, noise) nor omit strings from it (*i.e.*, omissions). In practice, training data for language models is notoriously noisy, raising the fundamental question:

How much contamination (in the form of omissions or insertions) can language generation tolerate?

Recent works made partial progress on this question by studying (non-dense) generation with either finite amounts of noise (but no omissions) [RR25, ICML] or omissions (but no noise) [BPZ26, SODA]. In this work, we characterize the contamination tolerance of both types of generation by proving the following results:

- ▷ **Generation under Contamination:** Language generation in the limit is achievable for all countable collections if and only if the fraction of contaminated examples converges to zero. When this condition fails, we characterize the collections which remain generable.
- ▷ **Dense Generation under Contamination:** We show that dense generation is strictly less robust to contamination than standard generation, requiring stronger conditions on the contamination rate for the density notions introduced in [KW25].

As a byproduct, we resolve an open question of [RR25] by showing that generation is possible with only membership oracle access to languages with finitely many contaminated examples.

Finally, we provide hope for practical noise tolerance while maintaining density via a beyond-worst-case analysis: We introduce a model where the adversary’s enumeration must be “close” to a canonical ordering of the language, capturing settings where simpler examples appear before complex ones. Here, we prove that dense generation is achievable even with infinite contamination provided the fraction of contaminated examples converges to zero. This result suggests that curriculum learning (the practice of presenting easier examples early in training that is widely used in pretraining) may be crucial for enabling models to learn despite the high contamination rates in datasets scraped from the web.

Contents

1	Introduction	1
1.1	Informal Results	5
1.1.1	Definitions	5
1.1.2	Our Results	6
1.2	Technical Overview	8
1.3	Open Problems	11
2	Related Work	12
2.1	Related Work on Language Identification in the Limit	12
2.2	Related Work on Language Generation in the Limit	12
2.3	Other Theoretical Efforts at Understanding Language Generation	14
3	Preliminaries and Model	14
3.1	Notation	15
3.2	Language Generation (without Noise)	15
3.3	Language Generation under Noise	17
3.4	Language Generation under Omission	18
3.5	Language Generation under Contamination	18
3.6	Density of Generators	20
3.7	What Type of Access Does the Generator Have To $\mathcal{L}$?	21
4	Algorithmic Templates	21
4.1	Priority-Based Intersection Meta Algorithm	21
4.2	Finite Expansion Sub-Routine	24
5	Generation in the Limit under Contamination	26
5.1	Generation with Vanishing Noise Rate and Arbitrary Omissions	27
5.2	Characterization of Generation with Constant Noise Rate and Arbitrary Omissions	29
6	Generation with Density under Contamination	30
6.1	Generation with Set-Based Density under Contamination	30
6.1.1	Set-Based Upper Density under Finite Contamination	31
6.1.2	Characterization of Set-Based Lower Density under Finite Contamination	33
6.1.3	Characterization of Set-Based Lower & Upper Density under Vanishing Noise Rate and Arbitrary Omissions	38
6.1.4	Characterization of Set-Based Lower & Upper Density under Constant Noise Rate and Arbitrary Omissions	40
6.2	Generation with Element-Based Density under Contamination	43
6.2.1	From Set-Based Lower Density to Element-Based Lower Density	43
6.2.2	Element-Based Density with Finite Contamination	47
6.2.3	Impossibility of Element-Based Density Under Infinite Contamination	48

7	Generation with Density Beyond the Worst-Case	49
7.1	Properties of Bounded Displacement Adversaries	51
7.2	Results for Set-Based Density	53
7.2.1	Generation with Set-Based Lower Density	53
7.2.2	Lower Bound for Set-Based Density	55
7.3	Results for Element-Based Density	56
7.3.1	Generation with Element-Based Lower Density	56
7.3.2	Lower Bound for Element-Based Upper Density	57
A	Generation with Repetitions in the Input Enumeration	63
B	Additional Results	63
B.1	Generation with Vanishing and Unknown Noise Rate: Sorting by Index Only	63
B.2	Generating from Finite Collections with Noise	65
C	Additional Preliminaries	68

1 Introduction

Large language models (LLMs) have transformed text generation and are already seeing wide-ranging applications from information retrieval to theorem-proving (e.g., [NRT25]). This progress has sparked a growing body of theoretical work seeking to understand the foundations of language generation. These works range from fine-grained analyses of specific architectures [WCM22; AS23; SHT23; CPW24; PNP24] to abstract investigations asking: when is coherent generation possible at all? Both views have led to useful insights. The fine-grained view has led to, e.g., speculative decoding [Che+23; LKM23] and faster algorithms for implementing attention [Wan+20; Zah+20; AS23; AS24; Han+24]. On the other hand, the abstract view has uncovered fundamental trade-offs between (lack of) hallucination and other desirable properties [HKKT18; KV24; KMV24; CP25a; KNVZ25; KW25] and provided principled approaches for aggregating multiple model outputs (e.g., [Hua+25]). At the same time, bringing these perspectives closer to practical applications is a natural goal: theoretical models that incorporate more realistic aspects of language generation can yield sharper insights. Our work resides in the abstract viewpoint on language generation and contributes to bringing it closer to the fine-grained view.

The study of language generation in computer science is not new; it predates LLMs. Indeed, Turing’s imitation game [Tur50] proposed language as a probe of cognition; Shannon [Sha51] analyzed the compressibility of English via its entropy; Chomsky [Cho56] introduced a hierarchy that has played a central role in understanding the computational power of different automata models [Sip12]; and Gold [Gol67] initiated one of the first works on studying when learning from examples is possible, which is arguably the essence of much of today’s learning theory.

Most relevant for us is Gold’s model of language identification in the limit, which formulates learning as an online two-player game between an adversary and a learner. In this game, the adversary selects a target language $K = L_z$ from a collection $\mathcal{L} = \{L_1, L_2, \dots\}$, and starts enumerating K one element at a time. After seeing each element, the learner’s goal is to guess the index z of the K ; and the learner is said to succeed if it stabilizes to the right guess z after some finite time (no matter what elements the adversary shows). A line of work, culminating in the works of Angluin [Ang79; Ang80], completely characterized the collections which are identifiable in the limit: this characterization showed that identification is largely intractable; even simple collections as regular languages are not identifiable in the limit.

This raises a puzzle: on the one hand, LLMs acquire non-trivial generation ability from just observing examples, on the other hand, the above model of language learning rules out learnability even for simple collections like regular languages. Kleinberg and Mullainathan [KM24] offered an elegant resolution to this puzzle by observing that the requirement of generation is weaker than the requirement in Gold’s learning model: generation only asks to produce new strings from the language, not to identify the entire language. To formalize this, they introduce a model of language *generation* in the limit:

Informal Definition 1 (Language Generation in the Limit; see Section 3). *The game is specified by a countable collection of languages $\mathcal{L} = \{L_1, L_2, \dots\}$ (such as all regular languages or all context-free grammars). First, the adversary fixes a target language $K \in \mathcal{L}$ and an enumeration of K .¹ Then, at*

¹Formally, an enumeration of K is an infinite sequence $x_1, x_2, \dots$ (possibly with duplicates) such that each $x_i \in K$ and every $x \in K$ appears at some position.

iteration $n \geq 1$, the adversary reveals x_n ; given the history $S_n = \{x_1, \dots, x_n\}$, the generator outputs a new string $w_n \notin S_n$, which is its guess for an unseen element of K .

A generator G is said to succeed in generating from $\mathcal{L}$ in the limit if for all $K \in \mathcal{L}$ and all enumerations of K , there exists a time n^* such that for every $n \geq n^*$, we have $w_n \in K \setminus S_n$. $\mathcal{L}$ is said to be generable in the limit if there is a generator G that succeeds for each $K \in \mathcal{L}$.

In sharp contrast to identification, [KM24] showed this goal is remarkably feasible: there is a generator G that succeeds in generating from any countable collection $\mathcal{L}$ (even the collection of all Turing-enumerable languages). This surprising positive result sparked a flourishing line of research exploring various aspects of language generation [KMV24; CP25a; CP25b; HKMV25; KMV25; KW25; LRT25; PRR25; RR25; BPZ26]; see Section 2 for a detailed discussion.

Language Generation with Density. A downside of [KM24]’s model is that it only requires the generator to generate new elements of the target language K without obtaining a meaningful coverage of it. For instance, their model accepts a generator that outputs “I generated 1,” “I generated 2,” and so on as a successful generator for English. While this generator indeed demonstrates an interesting ability, it can count, it hardly captures the richness of English. It turns out that the generator of Kleinberg and Mullainathan [KM24] exhibits a similar behavior as it operates by generating from progressively “thinner” subsets of K , effectively suffering from mode collapse. To address this limitation, Kleinberg and Wei [KW25] introduced *language generation with density*, requiring the generator to asymptotically cover a positive “fraction” of the target language. They proposed several notions of density that, at a high level, are grouped into two categories according to their view of a generator:

- ▷ **Element-based density** focuses on the output sequence generated during the infinite game, without taking into account the “internal” representation of the generator, *e.g.*, the weights of an LLM. This resembles the point of view of a user that is accessing the model through an API; the internal representation of the model might be evolving over time, but the user only sees its sequence of outputs. While we have not defined element-based density formally, we can still gain some intuition. Suppose that over its course of interaction with the adversary, G outputs the sequence $4\mathbb{N} = \{4, 8, 12, \dots\}$ and $K = \mathbb{N}$, then G achieves element-based density $1/4$. If on the other hand, K is the set of even numbers, then this density is $1/2$.
- ▷ **Set-based density** measures the density of all elements *producible* by the generator G at any given time. For this to make sense, G must be equipped with an option where after each round, we can pause and ask it to generate more and more elements. This is, of course, natural for real-world language models, which once trained are used to generate text as often as necessary. Compared to element-based density, set-based density tries to capture expressivity of one model, rather than of single elements generated by an infinite sequence of models.

Both densities rely on limiting behavior. Element-based density considers an infinite sequence of outputs produced by G . Set-based density, considers the limiting behavior of d_t as $t \rightarrow \infty$ where d_t is the density of the generator at the t -th step.² Depending on whether one considers $\liminf$ or $\limsup$ as the notion of a limit, we get four notions of density: lower/upper element-based

²Set-based density involves two limits: first, a limit over the sequence of elements producible by G in around t ; which is used to define d_t . Then the limit of d_t as $t \rightarrow \infty$. It turns out that changing the former limit from $\liminf$ to $\limsup$ does not qualitatively change the notion of density; hence, [KW25] fixed the first limit as $\liminf$.

density and lower/upper set-based density. Since $\liminf d_t \leq \limsup d_t$, lower density is harder to achieve than upper density.

Dense language generation (for any of the aforementioned notions of density) is a significantly stronger notion than generation in the limit because density prohibits the generator from retreating to generating from ever-shrinking corners of K . Yet, remarkably, [KW25] showed that many notions of dense generation are achievable for all countable collections:

- ▷ **(Element-based Density):** There is a generator that, for any countable collection, generates in the limit and has element-based lower density $\geq 1/8$. For the easier notion of element-based upper density, there is a generator that achieves density $\geq 1/2$ for all countable collections.
- ▷ **(Set-based Density):** Set-based lower density is not achievable for all countable collections. The weaker notion of set-based upper density, however, is: there is a generator achieving a set-based upper density of 1 for all countable collections.

In this work, we continue this study of dense language generation, bringing it closer to practice.

Language Generation under Contamination. A critical assumption in [KM24] and most subsequent works is that every element x presented to the generator is valid (*i.e.*, $x \in K$), and, conversely, every (fixed) element $x \in K$ is eventually presented to the generator. This stands in stark contrast to the reality of LLM training. Pre-training data is notoriously noisy, comprising an amalgam of text from sources of varying quality. Despite extensive resources devoted to data cleaning [Raf+20; Dod+21] and the emergence of entire industries focused on curating high-quality training data, the web-scale corpora used for pre-training LLMs remain riddled with errors, repetitions, low-quality content, and syntactically invalid code [Kre+22; Lee+22]. Moreover, while the amount of pre-training data keeps increasing, it is unrealistic to hope that it can, eventually, contain every factually correct statement. Yet modern LLMs consistently generate coherent text despite training on such imperfect data, raising the following question which is the starting point of our work: *Is language generation feasible from training data with large amounts of contamination?*

Two works take an initial step for capturing contamination [RR25; BPZ26]. Raman and Raman [RR25] allow finitely many invalid examples (but no omissions) and show generation remains possible from all countable collections. The finiteness of noisy examples is crucial for their analysis; it is not hard to show that if the noisy examples are unbounded their approach fails. The assumption of finite noise is particularly strong in this model where the amount of training data is unlimited. Bai, Panigrahi, and Zhang [BPZ26] allow omission of arbitrarily many elements from K (without adding any invalid ones) and, again, show generation is possible for all countable collections provided infinitely many survive. This restriction of no invalid elements, however, is inherent to their approach and their generator fails even with *two* invalid examples (Remark 3.4).³

In practice, both invalid elements and omissions persist simultaneously, and in infinite quantity. Understanding contamination-tolerance with both types of contamination in potentially unbounded quantity, is an important conceptual extension that both brings [KM24]’s model closer to reality and motivates new techniques to handle resulting challenges. This raises our first question:

³Apart from these results, both [RR25; BPZ26] also have additional results which are less related to our focus here, and we survey them in more detail in Section 2.

Question 1. How much *contamination* (both in the form of invalid examples and omissions) can generation tolerate while still being possible for all countable collections?

Moreover, as shown in [KW25], achieving density requires more sophisticated learning mechanisms, potentially making it more vulnerable to contamination. This raises our second question:

Question 2. Is dense language generation as contamination-tolerant as generation?

Our first set of results is a thorough answer to both of the above questions; we present informal statements of these results in the next section, but first present a quick summary below.

Summary of Results for Questions 1 and 2. We consider different regimes of the amount of contamination in the adversarial stream, and completely characterize when language generation in the limit is achievable under contamination across these regimes. Language generation shows surprising tolerance: under natural limits on the frequencies of contamination, generation remains possible for all countable collections even with infinite contamination of both types. In comparison, dense generation is much less tolerant: it becomes impossible for all countable collections with any infinite contamination, no matter how sparse.

This paints a bleak picture for contamination tolerance of dense generation, which is the more interesting type of generation as it ensures $\mathbb{G}$ actually learns a meaningfully rich subset of K . In practice, of course, data has much more regularity, which can be leveraged by language models for generation. This was already noted by [KM24]. Given the above results, a natural question is whether additional assumptions on the data-generating process can enable generation in the presence of infinite contamination. Or phrased in another way:

Question 3. Is there a natural beyond-worst-case model where dense language generation is achievable under infinite contamination?

A natural idea is to consider a stochastic model, where the samples are i.i.d. from some distribution. This was studied by [KMV25], who showed that stochasticity does not make achieving breadth (a stronger notion of density) any easier. Here, we introduce a new beyond-worst case model that limits the adversary’s freedom in choosing the enumeration of K . Roughly speaking, there is a canonical enumeration of K which can be thought of as having easier examples before harder examples, and the *bounded* adversary is not allowed to present hard examples “too early.” This is inspired by curriculum learning, which is a machine learning strategy that orders training data from easy to difficult, and is widely used to train LLMs [BLCW09].

Summary of Results Beyond Worst-Case. Briefly, we show that, with bounded adversaries, dense language generation (for all above notions of density) is achievable for all countable collections with (sufficiently sparse) infinite contamination; requiring the same amount of sparsity required as generating without density. We believe this model is of independent interest in the rapidly growing area of language generation.

Roadmap. In Section 1.1 we present the informal statements of the results described above. Then, in Section 1.2, we provide an overview of the proofs of the results. Finally, in Section 1.3 we further discuss our results and present open problems.

1.1 Informal Results

This section presents informal statements of our results. First, we present necessary definitions.

1.1.1 Definitions

We use $\mathbb{G} = (\mathbb{G}_n)_{n \in \mathbb{N}}$ to denote generators. For each n , $S_n = \{x_1, x_2, \dots, x_n\}$ denotes the set of elements revealed by the adversary in the first n iterations and $\mathbb{G}_n(x_1, \dots, x_n)$ denotes the corresponding output of $\mathbb{G}$. We consider two types of generators: element-based ones which output an element $\mathbb{G}_n(x_1, \dots, x_n) = w_n$, and set-based ones that output a set $\mathbb{G}_n(x_1, \dots, x_n) = G_n$.⁴

We already informally defined language generation in the limit for element-based generators in [Informal Definition 1](#). The set-based definition is very similar: A set-based generator $\mathbb{G}$ is said to generate from K in the limit if there is a finite n^* such that for all $n > n^*$, $G_n \subseteq K \setminus S_n$ ([Definition 5](#)).

Next, we define different notions of density from [\[KW25\]](#). For this, we need to fix a canonical ordering or enumeration of elements in each language L . We do so by fixing a canonical enumeration of the universe (of all possible strings) and letting L_n denote the first n elements in the induced canonical enumeration of L . The two element-based notions of density are as follows:

Informal Definition 2 (Element-based density; [Definition 9](#)). *Let $W = \{w_1, w_2, \dots\}$ be the sequence of outputs of an element-based generator $\mathbb{G}$ with target K . The element-based lower density and element-based upper density of $\mathbb{G}$ are $\mu_{\text{low}}(W; K) := \liminf_n \frac{|W \cap K_n|}{n}$ and $\mu_{\text{up}}(W; K) := \limsup_n \frac{|W \cap K_n|}{n}$ respectively.*

See [Appendix C](#) for a refresher on $\liminf$ and $\limsup$. Since $\liminf \leq \limsup$ for any sequence, the element-based lower density is harder to satisfy than element-based upper density. The two can be very different, e.g., if $W = \cup_{\ell \in \mathbb{N}} [(2\ell)!, (2\ell + 1)!]$, then $\mu_{\text{low}}(W, \mathbb{N}) = 0$ and $\mu_{\text{up}}(W, \mathbb{N}) = 1$. Next we introduce set-based notions of density.

Informal Definition 3 (Set-based density; [Definition 10](#)). *Let $G = \{G_1, G_2, \dots\}$ be the sequence of sets output by a set-based generator $\mathbb{G}$ with target K . The set-based lower density and set-based upper density of $\mathbb{G}$ are $\liminf_{n \rightarrow \infty} \mu_{\text{low}}(G_n; K)$ and $\limsup_{n \rightarrow \infty} \mu_{\text{low}}(G_n; K)$, respectively.*

As before, upper density is a weaker requirement than lower density. To gain some intuition, suppose $G_n = K$ if n is even and, otherwise, $G_n = \{w_i\}$ where w_i is some unseen string in K . Then $\mathbb{G}$ has set-based upper density of 1 and a set-based lower density of 0 (as $\mu_{\text{low}}(\{w_i\}, K) = 0$).

The final ingredient we need to state our results is the levels of contamination considered.

Informal Definition 4 (Contamination Regimes; [Definitions 7 and 8](#)). *Fix a language K and an enumeration $E = (x_1, x_2, \dots)$ not necessarily of K . We focus on the following four cases:*

1. **(Finite Contamination)** *E is an enumeration of K with finite contamination if $|E \triangle K| < \infty$.*
2. **($o(1)$ -Noise)** *E is an enumeration of K with $o(1)$ -Noise if $|\{x_1, x_2, \dots, x_n\} \setminus K|/n = o(1)$.*
3. **(Constant Noise)** *E is an enumeration of K with c -noise if $|\{x_1, x_2, \dots, x_n\} \setminus K|/n \leq c$, for large n .*
4. **(Arbitrary Omission)** *E is an enumeration of K with arbitrary omissions if $|K \setminus E| = \infty$.*

Finite contamination allows both invalid examples and omissions, already generalizing the case of just finite noise in [\[RR25\]](#). The next two regimes quantify the fraction of noisy examples among the first n : in $o(1)$ -noise this fraction tends to 0, and in c -noise, it becomes $\leq c$ for large enough n . With omissions, a notion of “the fraction of omissions” is more nuanced because we never know when an omission happened (a missing element could always appear later in the enumeration).

⁴This notion of set-based generation can be thought of as an “improper” version of index-based generation in [\[KW25\]](#) who restricted the outputs $G_n \in \mathcal{L}$; this improper version was also studied by [\[KMOV25\]](#).

With omissions, quantifying “the fraction of omissions” is more nuanced since missing elements could always appear later. Hence, we primarily focus on two regimes: finite omissions (a special case of finite contamination) and arbitrary omission (studied by [BPZ26] without noise).

1.1.2 Our Results

Our main results are comprehensive characterizations of when language generation with and without density is achievable under contamination. Interestingly, all our algorithmic results follow from two algorithmic templates (Section 4), explained in the technical overview (Section 1.2). Our first result studies generation under contamination; its formal version is in Section 5.

Informal Theorem 1 (Generation under Contamination). *The following results hold:*

1. Under $o(1)$ -noise and arbitrary omissions, all countable collections are generable in the limit.
2. Under c -noise (for any fixed $c \in (0, 1)$) and arbitrary omissions, a countable collection $\mathcal{L}$ is generable in the limit if and only if $\mathcal{L}$ satisfies the condition in Theorem 5.4 with parameter c . Moreover, for each $c \in (0, 1)$, there is a finite collection $\mathcal{L}$ that violates this condition.

The first result shows that generation is quite tolerant to contamination: even with infinite contamination, generation remains possible for all countable collections when the noise-fraction tends to 0. This significantly generalizes both [RR25; BPZ26]. It might seem surprising that we do not require any constraint on omissions; this stems from generation’s asymmetric objective: it requires generating new unseen elements, thus penalizing false positives much more than false negatives.

The second result explores the harder regime where noise fraction does not vanish, revealing the limits of generation’s contamination tolerance. While we omit the technical details of the condition, we note that all finite collections of size at most $1/c$ satisfy the condition with parameter c .

The algorithms for both regimes utilize a novel idea: they carefully re-order languages based on the adversary’s actions (Section 4), contrasting with existing algorithms that use static orderings.

Our next results study dense generation under contamination, starting with set-based density:

Informal Definition 5 (Set-based Dense Generation under Contamination). *The following results hold:*

1. Under finite contamination, all countable collections are generable with set-based upper density 1.
2. Under finite contamination, a countable collection $\mathcal{L}$ is generable in the limit with ρ set-based lower density if and only if $\mathcal{L}$ satisfies the condition in Theorem 6.5 with parameter ρ . Moreover, for each $\rho \in (0, 1]$, there is a countable collection that violates this condition.
3. Under $o(1)$ -noise rate and arbitrary omissions, a countable collection $\mathcal{L}$ is generable in the limit with (lower or upper) set-density ρ if and only if $\mathcal{L}$ satisfies the condition in Theorem 6.11 with parameter ρ . Moreover, for each $\rho \in (0, 1]$, there is a countable collection that violates these conditions.
4. Under c -noise rate (for $c \in (0, 1]$) and arbitrary omissions, a countable collection $\mathcal{L}$ is generable in the limit with (lower or upper) set-density ρ if and only if $\mathcal{L}$ satisfies the condition in Theorem 6.14 with parameters (c, ρ) . Moreover, for each $c, \rho \in (0, 1]$, there is a countable collection that violates these conditions.

Formal statements of these results appear in Section 6.1. The key takeaway is that dense generation is much less contamination-tolerant than non-dense generation. This contrasts sharply with

[KW25]’s positive results, which suggested density was achievable whenever generation was. Interestingly, while set-based lower and upper densities have different characterizations without contamination, they collapse to one under contamination (parts 3-4). Finally, set-based lower density becomes particularly challenging, even under finite contamination: it is unachievable whenever $\mathcal{L}$ has two languages, such as, $L = \mathbb{N}$ and $P = \{n \text{ is prime}\}_{n \in \mathbb{N}}$, that satisfy $P \subseteq L$ and $\mu_{\text{low}}(P, L) = 0$ (part 2). This is a much more stringent requirement than the requirement without contamination [KW25], who showed that impossibility of set-based generation with lower density requires the existence of a pathological sub-collection of $\mathcal{L}$ with infinite cardinality.

Remark 1.1. The result for set-based upper density is stronger: we are able to achieve it under finite noise and an *infinite* amount of omissions provided the omissions are not “too dense;” we formalize this in Section 3 as *c*-omissions. The formal results appear as Theorems 6.1 and 6.4.

All algorithms above rely on our two algorithmic templates: finite contamination uses one template, while the remaining cases use the other. We believe these templates can be of independent interest in the study of language generation. As an illustration, we use them to answer an open question from [RR25]:

Corollary 1.2 (Membership-Query–Based Algorithm with Finite Contamination; Corollary 4.5). *There is a computable $\mathbb{G}$ that, for any countable collection $\mathcal{L} = \{L_1, L_2, \dots\}$, given access to an oracle which given w and i answers “Is $w \in L_i$?,” generates $\mathcal{L}$ in the limit under finite contamination.*

In contrast, [RR25] required additional oracles to achieve generation under finite noise.

Next, we study element-based density under contamination. Surprisingly, unlike set-based density, element-based density does not uniformly become harder in each contamination regime.

Informal Theorem 2 (Element-based Dense Generation under Contamination). *The following hold:*

1. *Under finite contamination, every countable collection $\mathcal{L}$ is generable in the limit with element-based upper density $\rho = 1/2$.*
2. *Under finite contamination, every countable collection $\mathcal{L}$ is generable in the limit with element-based lower density $\rho = 1/8$.*
3. *Under $o(1)$ -noise, there is a collection that isn’t generable in the limit with >0 element-based upper density.*

In fact, our formal versions of parts 1 and 2 are stronger: We show that if there exists an algorithm that achieves element-based lower (respectively upper) density ρ for all countable collections in the contamination-less case, there exists an algorithm that achieves element-based lower (respectively upper) density ρ under finite contamination (Theorem 6.18). Hence, remarkably, element-based density with finite contamination is exactly as hard as with no contamination, contrasting with set-based lower density which became significantly harder. (Indeed, the upper and lower densities of $\rho = 1/2$ and $\rho = 1/8$ match those obtained by [KW25] with no contamination.) The formal statements of these results appear in Section 6.2.

Language Generation Beyond the Worst-Case (Section 7). Next, we introduce a beyond-worst-case model which restricts the order in which the adversary can present the elements K .

Informal Definition 6 (Bounded Adversary; Definition 11). *Let the first n elements of K be $\{\kappa_1, \kappa_2, \dots\}$. We say that an adversary is M -bounded if it presents an enumeration $x_1, x_2, \dots$ such that, for sufficiently large n , if $x_n \in K$ then $x_n = \kappa_{i_n}$ for some $i_n \leq Mn$.*

Importantly, an M -bounded adversary can still select arbitrary target languages K but cannot present its elements in an arbitrary order. Under the interpretation that the canonical enumeration of K places “easier” elements before “harder” ones, M -bounded adversaries cannot, infinitely often, enumerate very hard examples way before easier ones. This is inspired by practical observations of LLM training; *e.g.*, it is a crucial folklore practice that LLMs are first trained on “easier” tasks before harder ones. More broadly, this phenomenon is known as “curriculum learning” [BLCW09; HW19].

In this beyond-worst-case model, we show that dense generation is much more tractable.

Informal Theorem 3 (Dense Generation with Bounded Adversary). *With an M -bounded adversary:*

1. *Under $o(1)$ -noise and arbitrary omissions, all countable collections are generable in the limit with set-based lower density $\geq \frac{1-\varepsilon}{M}$ (for any fixed $\varepsilon > 0$). Moreover, there is a finite collection for which it is impossible to generate in the limit with set-based upper density $> 1/M$.*
2. *Under $o(1)$ -noise and arbitrary omissions, all countable collections are generable in the limit with element-based lower density $\frac{1-\varepsilon}{2M}$ (for any fixed $\varepsilon > 0$). Moreover, there is a finite collection for which it is impossible to generate in the limit with element-based upper density $> 1/M$.*

Thus, both set-based and element-based lower densities become achievable for all countable collections with M -bounded adversaries. This is a significant improvement over arbitrary adversaries where even finite contamination was problematic; and no notion of density was achievable with finite contamination (no matter how sparse). We remark that, while dense generation is more tractable, the beyond worst-case model remains non-trivially hard: identification largely remains impossible in the model (Theorem 7.6) and generation without density still fails at constant noise rates as in Informal Theorem 1 (Theorem 7.7). The formal statement of this result appears in Section 7.

Proper vs. Improper Learning in the Worst-Case. We conclude this section with the following:

Remark 1.3 (Proper vs. Improper Learning in the Worst-Case). For the vast majority of our set-based results, the learning algorithms are *improper* meaning that the set they output is not part of the collection $\mathcal{L}$. While we allow the outputs to be arbitrary, the outputs of our algorithms are much more structured. For instance, several of our algorithms output *intersections* of finitely many languages from $\mathcal{L}$. The remaining algorithms output a set G such that $|G \Delta L| < \infty$ for some $L \in \mathcal{L}$. In fact, Example 3.3 shows that this is necessary: proper learning in the presence of contamination is much more restrictive than improper learning. This is in sharp contrast to the uncontaminated setting, where the results of [KM24; KW25] show that proper learners are as powerful as improper ones.

1.2 Technical Overview

In this section, we give an overview of our techniques and how they relate to prior works. Our main results are a comprehensive characterizations of when different notions of generation can or cannot be achieved under different levels of contamination. We divide this into two parts: upper bounds (or algorithms) and lower bounds.

Upper Bounds for Dense and Non-Dense Generation

All of our algorithms across the worst-case model and the bounded adversarial model, with and without density, rely on two algorithmic templates. Here, we overview these templates. They are discussed in more detail in [Section 4](#).

Finite-Expansion Sub-Routine ([Section 4.2](#)). Our first algorithmic template is quite simple and enables us to achieve dense and non-dense generation with finite contamination. The algorithm relies on the following elementary observation: If the enumeration E generated by the adversary has finite contamination, then it must be a 0-contamination enumeration of a language K' satisfying with $|K' \triangle K| < \infty$. Now, of course, K' might not be a language in our collection $\mathcal{L}$, but the natural thing to do is to add K' to $\mathcal{L}$. We will need to add all possible choices since we do not know K' in advance. Then, we end up with the following collection

$$\widetilde{\mathcal{L}} := \{L_{A,B} := L \cup A \setminus B : L \in \mathcal{L}, A \subseteq \Sigma^* \setminus L, B \subseteq L, |A| < \infty, |B| < \infty\},$$

where Σ^* is the universe of strings. Since the number of finite subsets of a countable collection are countable, $\widetilde{\mathcal{L}}$ is a countable collection. This immediately gives us an algorithm for generation with finite amounts of contamination since we can use [\[KM24\]](#)'s algorithm that works for all countable collections with $\widetilde{\mathcal{L}}$ instead of $\mathcal{L}$. While this idea is obvious in hindsight and analyzing it is also straightforward, it has many useful properties that make it widely applicable:

- ▷ As we have seen, it preserves countability (if $\mathcal{L}$ is countable, so is $\widetilde{\mathcal{L}}$)
- ▷ Further, membership access to $\mathcal{L}$ is sufficient to get membership access to $\widetilde{\mathcal{L}}$ ([Lemma 4.4](#))⁵
- ▷ If $\mathbb{G}$ generates with density ρ (for any notion of density we study) with respect to $\mathcal{L}$, then $\mathbb{G}$ also generates density ρ with respect to $\widetilde{\mathcal{L}}$.

This is not an exhaustive list. For instance, the transformation also preserves generation with approximate breadth, which is a stronger notion than density introduced in [\[KMV25\]](#). These properties are what make this transformation interesting, and it leads to a number of results. In particular, points 1 and 2 above together allow us to get a membership oracle-based algorithm for generation with finite contamination that already resolves an open question in [\[RR25\]](#); which seems hard to get using prior techniques.

Priority-Based Intersection Algorithm ([Section 4.1](#)). Next, we discuss our algorithms for the much more involved case where there are amount of contamination is infinite. We begin by explaining why earlier approaches fail:

1. The previous simple approach that constructs the “expanded” collection $\widetilde{\mathcal{L}}$ fails because the collection $\widetilde{\mathcal{L}}$ becomes uncountable (if the amount of noise is not finite) and not all uncountable collections are generable in the limit, even without the requirement of density. The situation with density is even more complicated as the transformation is no longer density preserving.
2. Another idea is to use approaches from [\[RR25; BPZ26\]](#). [\[RR25\]](#) use a nice observation for generating in the limit: if the amount of noise is finite then, for sufficiently large n , the

⁵A membership oracle to $\mathcal{L} = \{L_1, L_2, \dots\}$ is a primitive that, given $w \in \Sigma^*$ and i , answers “Is $w \in L_i$?”

second half of the training examples (namely, $\{x_{n/2}, x_{n/2+1}, \dots, x_n\}$) eventually contains no noisy examples. Hence, roughly speaking, feeding the second half of the examples to an appropriate generation algorithm suffices. This, of course, fails when there is an infinite amount of noisy examples in the stream. [BPZ26] make another nice observation: if $\mathbb{G}$ has the property that it generates from $\mathcal{L}$ after $n^* = n^*(K)$ iterations, where n^* depends on the target K , but not on its enumeration, then $\mathbb{G}$ generates from $\mathcal{L}$ in the limit under arbitrary omissions (provided there is no noise). This suffices as [CP25a] constructed a generator with this property for all countable collections. However, this approach fails even when there are two noisy examples in the enumeration without any omission (see Remark 3.4).

To understand our (meta) algorithm, it is instructive to first understand the algorithm of [CP25a]. Roughly speaking, in the n -th iteration, their algorithm considers the n languages $\{L_1, L_2, \dots, L_n\}$ and, from these, it removes any language inconsistent with the training data seen so far S_n , i.e., any $L \not\supseteq S_n$. Let the resulting languages be $L_{\sigma(1)}, L_{\sigma(2)}, \dots, L_{\sigma(m)}$ (for $m \leq n$), then they output $L_{\sigma(1)} \cap L_{\sigma(2)} \cap \dots \cap L_{\sigma(\ell)}$ for the largest $1 \leq \ell \leq m$ such that the resulting intersection is infinite.⁶

Of course, this algorithm does not directly work when there is contamination in the sample stream because, for instance, the consistency check of $L \supseteq S_n$ is no longer meaningful. Consider the case of c -noise; let the enumeration be $E = \{x_1, x_2, \dots\}$. The natural counter part is to check whether E is an c -noisy enumeration of L . This, however, requires having access to the entire enumeration E , which we have never seen at any iteration $n < \infty$. One could instead check if the fraction of elements from $\{x_1, x_2, \dots, x_n\}$ not in L , which we term the *empirical noise rate*, is at most $c + \varepsilon$ for small fixed $\varepsilon > 0$. There are examples where this fails because this approach can include “bad” languages L that do not meet the actual requirement that E is a c -noisy enumeration for L . One could tighten this check by setting $\varepsilon = 0$, but then we run into the issue that the empirical noise rate for certain bad languages can fluctuate above and below c infinitely often. To gain some intuition suppose $K = \{2, 4, 6, \dots\}$ and $L = \{n : n \in \mathbb{N}, n \bmod 4 \neq 0\} \cap A$ where $A = \bigcup_{\ell \in \mathbb{N}} [(2\ell)!, (2\ell + 1)!]$. Then, the enumeration $E = \{1, 2, 3, \dots\}$, is a $1/2$ -noisy enumeration for K and $3/4$ -noisy enumeration of L (so L should be excluded in our check). However, the empirical noise rate of L fluctuates between 0 and $3/4$ infinitely often (and, hence, L would end up being included in our check infinitely often).

This is not merely a superficial problem in the above approaches, but rather an inherent problem in verifying c -noise or $o(1)$ -noise, which are necessarily limiting phenomena, at finite times. Hence, to overcome this we need a new approach that is able to (i) remove “bad” languages from our “active set” of languages, and (ii) ensure that the target language K always remains in this set after some finite time. To design our approach, we take inspiration from the failure of the above approaches, where certain “bad” languages fluctuate between passing our check and failing our check infinitely often. Our algorithm assigns each language a priority. Initially, the priority of language L_i is simply $-i$ (so the languages in order of priority are $L_1, L_2, \dots$) and, each time, L_i fails our check we penalize the language by decreasing its priority. The key observation is that for both c -noise and $o(1)$ -noise, the target language K will only be penalized for a finite amount of time; where as every “bad” language will be penalized infinitely often. This idea is sufficient for us to design algorithms for generation with c -noise and $o(1)$ -noise.

However, like our previous algorithmic template, this also has quite general applicability: to

⁶To get an element-based generator, they output the smallest unseen element from this set.

all notions of density and also for generation with bounded adversaries. Indeed, this core algorithmic template of assigning priorities to languages shows up in all of our algorithms beyond the finite contamination regime, each time, with a slightly different notion of priority. For instance, with M -bounded adversaries, we need to check whether the provided enumeration E is indeed M -bounded for a language L ; this is again a limiting phenomenon which cannot be checked at any finite time. The situation for dense generation is even more complicated. Indeed, [KW25] already demonstrated that achieving lower density (both element-based set-based) for all countable collections is incredibly difficult and their algorithms carefully need to balance the trade-off between generating only valid elements and covering enough fraction of K . Now, with infinite amount of contamination this becomes even harder due to the fluctuating empirical noise rates we discussed above.

Lower Bounds for Dense and Non-Dense Generation

A common theme in learning theory is that, once we have the “right” algorithm, obtaining tight lower bounds is not that hard. This is also the case with our lower bounds. An interesting property of our lower bounds is that many of them have a finite witness. For instance, consider dense generation with finite contamination. All our characterizations here have the following form: a collection $\mathcal{L}$ violates the condition if there are two pathological languages $L_1, L_2 \in \mathcal{L}$ with a certain property. (For instance that $L_1 \subseteq L_2$ and $\mu_{\text{low}}(L_1, L_2) = 0$.) This contrasts existing lower bounds in generation and identification which required infinite witnesses. For instance, [KW25] showed that the existence of an infinite perfect tower characterizes lower set-based density. However, as its name suggests, an infinite perfect tower is necessarily witnessed by an infinite sub-collection of languages. Similarly, the characterizations of identification in the limit [Ang80] and language generation with breath [KMV24; CP25a; KMV25] also require infinite witnesses.

1.3 Open Problems

There are several interesting future directions related to noisy generation, and more broadly, the line of work on generation in the limit. Regarding noisy generation, an interesting direction is to fully characterize element-based generation for all types of contamination we consider in our work. Moreover, it would be nice to obtain tight bounds in the beyond-worst-case setting we introduced in our work. While we have developed an algorithm that uses only membership oracle access to $\mathcal{L}$ in the setting of finite contamination, we have developed several algorithms that require more complicated oracles. It is an interesting open direction to understand what can be done using simpler oracles in this setting. Last but not least, perhaps a way to circumvent some of the lower bounds we have shown, other than restricting the adversary, is to relax the requirement of the learner: how does the landscape of (noisy) generation look like we allow the generator to output a vanishing amount of hallucinations? We remark that a similar question was asked by [KW25] in the context of improving the density guarantees of their algorithms.

2 Related Work

In this section, we present further related work, including other works on language identification in the limit and language generation in the limit. In an attempt to give through overviews of the presented works, this section is quite long, perhaps unavoidably. That said, reading it is not necessary to understand our results and we encourage readers to skip it in the first reading and revisit as necessary.

2.1 Related Work on Language Identification in the Limit

Starting from Gold [Gol67] there has been a rich line of work including both linguistics and computer science on the model of language generation in the limit. Of particular relevance to our work are the line of works on language identification in the limit in settings where the adversary can include invalid examples in the enumeration (*i.e.*, noise) or omit elements [Sch85; FJ89; BJS92; CJS97; Ste97; MS03a]. In particular, Jain [Jai94] extends Gold’s model to streams with infinitely many inaccuracies, controlling corruption via density (covering finite, vanishing, and constant-rate noise). Mukouchi and Sato [MS03b] use a neighbor-system (metric) model that permits insertions and deletions provided each corrupted item lies within a fixed distance of some true string. Tantini, Higuera, and Janodet [THJ06] analyze the same metric-based setting and prove identification results under such systematic noise.

2.2 Related Work on Language Generation in the Limit

Next, we discuss works on language generation in the limit. We have already discussed some of these works [KM24; KW25; RR25; BPZ26] briefly in the introduction. We expand upon the discussion of these and other works below.

Language Generation with Uncountable Collections. As we mentioned in Section 1, [KM24] introduced the model of language generation in the limit. While they and many subsequent works, including ours, focus on the setting where the collection of languages is countable, Li, Raman, and Tewari [LRT25] extended the study to uncountable collections of languages. [LRT25] introduced two more fine-grained notions of language generation: uniform and non-uniform generation. Uniform generation requires the number of strings n^* the generator needs to see before starting to generate (see Definition 5) to be independent of the choice of the target language K . Non-uniform generation allows n^* to depend on K , but requires it to be independent of the enumeration of K chosen by the adversary. (Note that generation in the limit allows n^* to be dependent of both K and the enumeration of K .) [LRT25] characterized the collections that are uniformly generable as well as collections that are non-uniformly generable. The latter characterization, in particular, demonstrates that all countable collections can be non-uniformly generated although with a different algorithm than that of [KM24]; this result was also concurrently and independently obtained by [CP25a]. Interestingly, Charikar and Pabbaraju [CP25a] showed that this type of generation is *impossible* if the learner only has membership oracle access to $\mathcal{L}$. [LRT25] gave several sufficient conditions for uncountable collections to be generable in the limit and left a complete characterization as an open problem. En route to obtaining a complete characterization, they left other open problems, notably checking closedness under finite unions: if $\mathcal{L}_1, \mathcal{L}_2, \dots, \mathcal{L}_k$ are generable in the

limit, then is $\cup_{i=1}^k \mathcal{L}_i$ also generable in the limit? [HKMV25; BPZ26] concurrently and independently resolved this problem by showing that the statement is false even with $k = 2$. Their proofs are near-identical and rely on a diagonalization argument which has also been featured in other works on language generation as discussed below.

Language Generation in a Statistical Model. [KM24]’s model has also been extended along another access: adversarial enumeration of examples. While [KM24], study generation when the examples are provided by a worst case adversary, [KMV25] introduce a statistical model where the samples are generated i.i.d. from a *fixed* distribution $\mathcal{P}$ whose support matches the target language K . In this model, they obtain the first and near-tight sample complexities of identification in the limit, generation in the limit, and related tasks (which are discussed later in this section). This is precisely the notion of sample complexity studied in the universal-rates framework of [Bou+21]. Apart from [Bou+21], [KMV25]’s model was also described by [Ang88] in the context of language identification; [Ang88] showed that the characterization of which collections are identifiable does not change between the stochastic and in-the-limit models. This model can also be thought of as a model beyond the worst-case, as the enumeration shown to the generator is sampled i.i.d. from a distribution instead of being chosen by an adversary. However, it is quite different from the beyond-worst-case model we study in this work. Neither model is a special case of the other. On the one hand, in the stochastic model the adversary can construct a distribution with a heavy tail ensuring that elements with large indices in the canonical ordering can appear early in the samples so does not meet the boundedness requirement (Definition 11). On the other hand, the bounded adversary model in this work allows the enumeration selected to be adversarial (and, in particular, non-stochastic) provided it meets the boundedness requirements. It is an interesting question to understand if the stronger results we obtain for dense generation can also be obtained in the statistical model.

Language Generation in the Limit with Noise. Having discussed the variants of language generation in the literature, we now turn to discussing works on language generation with noisy examples. As already mentioned in Section 1, there are only two relevant works in the literature: [RR25; BPZ26]. [RR25] initiated the study of language generation in the limit with a *finite* amount of additive noise n^* (unknown to the generator); in this setting, they characterized classes that are uniformly generable. Their results, in particular, imply that all countable collections remain generable with a finite amount of additive noise. In the context of generation from noisy samples, [BPZ26] paper shows that infinite omissions do not change which collections are uniformly or non-uniformly generable. Combining this with the result in [CP25a] implies that all countable collection remains non-uniformly generable with infinite omissions. [BPZ26] also study the setting in [RR25]; they show several results including a characterizations of non-uniform generation with a finite amount of additive noise. The main additional challenges in our work is that we deal with (i) the setting with an *infinite* amount of additive noise (where the algorithms from these works fail) and (ii) also simultaneously deal with omissions.

Breadth and Density in Language Generation. Prior to the density-based definitions of “breadth” proposed by Kleinberg and Wei [KW25], the parallel works of Kalavasis, Mehrotra, and Velegkas [KMV24], Charikar and Pabbaraju [CP25a], and Kalavasis, Mehrotra, and Velegkas [KMV25] introduced more stringent notions of breadth, asking (roughly) that the (infinite) set of elements the generator can produce in every timestep is eventually a subset of the target and misses only

finitely many elements from K . They characterized when this family of notions of breadth can be achieved. Moreover, [KMV25] introduced and studied a statistical model of language generation, and [CP25a] explored other aspects of generation including feedback, non-uniform⁷ generation with various types of oracle-access to the collection, and the role of feedback in generation. On a related note, Peale, Raman, and Reingold [PRR25] introduced a notion of “representative generation” where the strings of the universe are divided into different groups and the generator’s outputs are required to represent all groups. This can also be thought of as a weakening of the notions of breadth studied in aforementioned works.

Further Works on Language Generation. Apart from the aforementioned works on language generation, there are two other recent papers studying this framework. Charikar and Pabbaraju [CP25b] study a notion of Pareto-optimal non-uniform generation, which seeks to obtain algorithms that achieve the Pareto frontier for non-uniform generation. On a different front, Karbasi, Montasser, Sous, and Velegkas [KMSV25] propose and study a theoretical model of hallucination detection that is inspired by the language identification and generation settings. Lastly, Vafa et al. [Vaf+24] propose a theoretical model that measures the proximity between the language represented by an LLM and the language it was trained on..

2.3 Other Theoretical Efforts at Understanding Language Generation

For completeness, we discuss some other theoretical efforts on understanding LLMs.

Representational Power of Transformer Architecture. There has been a long line of work trying to understand the class of functions that transformers can represent; see, *e.g.*, [SHT23; CPW24; PNP24] and references therein.

Effectiveness of Chain-of-Thought. Another recent theoretical effort has to do with building theoretical evidence about the usefulness of chain-of-thought in language models [Mal23; AML25; HWL25; Jos+25].

Models Hallucinations. There have several theoretical works giving evidence that LLMs that provide some non-trivial utility *must* hallucinate [KV24; XJK24; KNVZ25].

Model Stealing. The process of trying to infer the parameters of a proprietary LLM through calls to its API is known as *model stealing*. Liu and Moitra [LM25], building on the work of Mahajan, Kakade, Krishnamurthy, and Zhang [MKKZ23], showed that, under a theoretical model, this is possible for any LLM that has low rank.

3 Preliminaries and Model

In this section, we formalize language generation with adversarial noise in the example stream. We follow the learning-theoretic presentation of generation in the limit used in recent work, and then parameterize the adversary by the amount of incorrect or *noisy* examples it can inject and the number of elements from K it can skip or *omit* from the enumeration.

⁷This is a technical term which studies how quickly the learner can start outputting valid, unseen elements of K .

3.1 Notation

Let Σ be a finite alphabet. We use Σ^* to denote the set of all finite strings over Σ . We often call Σ^* the *universe* and use U to denote it. We will often take $U = \Sigma^*$ to be the set of natural numbers $\mathbb{N}$. This is without loss of generality as Σ^* is a countable set because Σ is finite. Given a sequence $x = (x_1, x_2, x_3, \dots)$ of elements of U , we use $x_{1:n}$ to denote the prefix of elements $x_1, x_2, \dots, x_n$.

For us, a *language* is any infinite subset $L \subseteq U$. An enumeration of a language L is an ordered list $\ell_1, \ell_2, \ell_3, \dots$ of elements of L , so that every element of L appears in the list exactly once.⁸ Given a collection $\mathcal{L}$ of languages (*i.e.*, a collection of infinite sets in U), the closure of $\mathcal{L}$ is the common intersection of all languages in $\mathcal{L}$ and is denoted as $\text{Cl}(\mathcal{L}) := \bigcap_{L \in \mathcal{L}} L$. When talking about densities, we will also endow the universe U with a canonical ordering $u_1, u_2, \dots$. When $U = \mathbb{N}$, the canonical enumeration will always be $1, 2, 3, \dots$. This canonical ordering induces a canonical ordering of each language $L \subseteq U$ and, for language L , we denote it by $\ell_1, \ell_2, \dots$. Intuitively, one can think of this ordering as ordering the elements of L from “easy” ones to “harder” ones; we will return to this intuition in [Section 7](#) where we go beyond the worst-case adversarial model.

Throughout the paper, we use standard asymptotic notations $o(\cdot)$, $O(\cdot)$, $\Omega(\cdot)$, and $\Theta(\cdot)$. Finally, to define different notions of densities of a generator, we need the following standard notions of densities of subsets of natural numbers:

Definition 1 (Set densities in $\mathbb{N}$). *Let $A = \{a_1, a_2, a_3, \dots\}$ and $B = \{b_1, b_2, b_3, \dots\}$ be subsets of $\mathbb{N}$, with their elements listed in the natural ordering of $\mathbb{N}$.*

- ▷ The upper density of A in B is $\mu_{\text{up}}(A, B) = \limsup_{n \rightarrow \infty} \frac{1}{n} |\{A \cap \{b_1, \dots, b_n\}\}|$.
- ▷ The lower density of A in B is $\mu_{\text{low}}(A, B) = \liminf_{n \rightarrow \infty} \frac{1}{n} |\{A \cap \{b_1, \dots, b_n\}\}|$.

We refer the reader to [Appendix C](#) for a quick refresher on the definition of the limit superior and limit inferior. Now, to gain some intuition about $\mu_{\text{up}}(\cdot)$ and $\mu_{\text{low}}(\cdot)$, consider the following sets

$$E = 2\mathbb{N} := \{2n\}_{n \in \mathbb{N}}, \quad P = \{n \text{ is prime}\}_{n \in \mathbb{N}}, \quad \text{and} \quad F = \bigcup_{i \in \mathbb{N}} [(2i)!, (2i+1)!].$$

These languages have the following densities in the universe $U = \mathbb{N}$:

$$\mu_{\text{up}}(E, U) = \mu_{\text{low}}(E, U) = \frac{1}{2}, \quad \mu_{\text{up}}(P, U) = \mu_{\text{low}}(P, U) = 0, \quad \mu_{\text{up}}(F, U) = 1 \quad \text{and} \quad \mu_{\text{low}}(F, U) = 0.$$

Hence, as we can see, lower and upper densities of L in $\mathbb{N}$ can be different for L and the lower and upper density for one language in $\mathbb{N}$ can be the same or different. Due to the definition of limit inferior and limit superior, it always holds that $\mu_{\text{low}}(L, U) \leq \mu_{\text{up}}(L, U)$.

3.2 Language Generation (without Noise)

Let $\mathcal{L} := \{L_1, L_2, L_3, \dots\}$ denote a countable collection of languages in U . In the language generation problem formalized by [\[KM24\]](#), an adversary fixes a target language $K \in \mathcal{L}$ and an enumeration of K . At round $n \in \mathbb{N}$, the adversary reveals an element $x_n \in K$ according to the chosen enumeration, and the goal of generator is to be able to generate from the target language K

⁸We note that our definition of enumeration differs from some of the previous works in that we do not allow repeated elements. As we detail in [Appendix A](#), this assumption is without loss of generality.

eventually. We call this setting the noiseless setting because the adversary commits to a complete enumeration of all the elements from the target language K , neither including noisy examples nor omitting examples in its enumeration.

Remark 3.1 (Infinite Cardinality of Languages). Following [KM24] and other prior works, we will always assume that each language $L \in \mathcal{L}$ has infinite cardinality, as otherwise, when $K = L$, the set of unseen valid elements a generator can output, namely, $K \setminus S_n$, can become empty.

In this work, we consider three natural notions of generation introduced by [KMV24; CP25b; KW25]: index-based generation, element-based generation, and set-based generation. Towards defining this, we first need to define three types of generators: index-based, element-based, and set-based.

Definition 2 (Index-based Generators; [KW25]). *An index-based generator $\mathbb{G} = (\mathbb{G}_n)_{n \in \mathbb{N}}$ maps the observed history to an index $i \in \mathbb{N}$, i.e., given examples $x_1, \dots, x_n$ revealed by the adversary till round n , the generator outputs $\mathbb{G}_n(x_1, \dots, x_n) = i_n \in \mathbb{N}$.*

Definition 3 (Element-based Generators; [KW25]). *An element-based generator $\mathbb{G} = (\mathbb{G}_n)_{n \in \mathbb{N}}$ maps the observed history to an element $w \in U$ different from the elements revealed by the adversary so far and the previously generated elements, i.e., given examples $x_1, \dots, x_n$ revealed by the adversary till round n , the generator outputs $\mathbb{G}_n(x_1, \dots, x_n) = w_n \in U \setminus (\{x_1, \dots, x_n\} \cup \{w_1, \dots, w_{n-1}\})$.*

Definition 4 (Set-based Generators; [CP25a; KMV25; KW25]). *A set-based generator $\mathbb{G} = (\mathbb{G}_n)_{n \in \mathbb{N}}$ maps the observed history to a set of element $S \subseteq U$ not containing any element revealed by the adversary so far, i.e., given examples $x_1, \dots, x_n$ revealed by the adversary till round n , the generator outputs $\mathbb{G}_n(x_1, \dots, x_n) = A_n \subseteq U \setminus \{x_1, \dots, x_n\}$.*

Having defined the three types of generators, we are now ready to define language generation in the limit.

Definition 5 (Language Generation in the Limit; [KM24; KW25]). *A generator $\mathbb{G} = (\mathbb{G}_n)_{n \in \mathbb{N}}$ is said to generate in the limit from a language collection $\mathcal{L}$ if for any $K \in \mathcal{L}$ and any enumeration of K , there exists a finite time n^* such that for all $n \geq n^*$, the following holds:*

- ▷ If $\mathbb{G}$ is an index-based generator, then $L_{i_n} \subseteq K$;
- ▷ If $\mathbb{G}$ is an element-based generator, then $w_i \in K$; and
- ▷ If $\mathbb{G}$ is a set-based generator, then $A_n \subseteq K$.

To draw comparisons between the three notions of generation above, clearly index-based generation is the most restrictive notion, as it implies set-based generation, and set-based generation implies element-based generation. However, simply getting set-based generation in the limit is not interesting, as once element-based generation in the limit is possible, one can simply output a singleton set according to the element-based generator. We introduce the set-based generation to study the breadth achieved by such a generator, which we will discuss in Section 3.6. Here, element-based generation is identical to the notion of language generation in the limit introduced by [KM24].

Recent results show that, in the noiseless setting, generation in the limit is possible for every countable collection $\mathcal{L}$ [KM24]. In fact, the algorithm of [KM24] achieves index-based generation, the most restrictive notion of generation. However, as we will show, index-based generation is no longer sufficient once we consider the noisy setting where the input stream of examples contains noisy examples.

3.3 Language Generation under Noise

It is easy to see that when infinitely many noisy examples are allowed, we need some restriction on the adversary to enable generation. For instance, without any restrictions the adversary can simply enumerate K and another language K' disjoint from K in alternate steps, and we never know whether to generate from K or K' . Towards this, we define the noise rate as follows.

Definition 6 (Empirical Noise Rate). *For a language L and an infinite sequence $x_{1:\infty} \in U^{\mathbb{N}}$, define the (empirical) noise rate up to time n by*

$$R(L; x_{1:n}) = \frac{1}{n} |\{t \leq n : x_t \notin L\}|.$$

Now we can define noisy enumeration as follows.

Definition 7 (Regimes of Noisy Enumeration). *Fix a language $L \subseteq U$:*

- ▷ **$o(1)$ -Noise Enumeration:** *An enumeration of L with $o(1)$ -noise is an ordered list $x_1, x_2, x_3, \dots$ of elements so that every element of L appears in the list exactly once, and the empirical noise rate satisfies $R(L; x_{1:n}) = o(1)$.*
- ▷ **c -Noise Enumeration:** *An enumeration of L with c -noise is an ordered list $x_1, x_2, x_3, \dots$ of elements so that every element of L appears in the list exactly once, and there exists n^* such that for all $n \geq n^*$, the empirical noise rate satisfies $R(L; x_{1:n}) \leq c$.*
- ▷ **Finite Noise Enumeration:** *An enumeration of L with finite noise is an ordered list $x_1, x_2, x_3, \dots$ of elements so that every element of L appears in the list exactly once, and there are at most a finite number of noisy examples in the enumeration, i.e., $|\{x_1, x_2, \dots\} \setminus L| < \infty$.*

In this work, we focus on three regimes:

1. **Vanishing noise rate:** The adversary chooses an enumeration of K with $o(1)$ -noise.
2. **Constant noise rate:** The adversary chooses an enumeration of K with c -noise (for $c \in [0, 1]$).
3. **Finite noise:** The adversary chooses an enumeration of K with finite noise.

Both the first and the second regimes strictly generalize the third regime, “finite-noise” model with only finitely many noisy examples, which has been initiated and studied for generation by [RR25] and later by [BPZ26].

Similar to the noiseless setting, we say a generator generates from a given language collection $\mathcal{L}$ in the limit under vanishing noise rate/constant noise rate c /finite noise, if for any enumeration of the target language K with $o(1)$ -noise/ c -noise/finite noise, the generator eventually generates from the target language.

Separation of Proper and Improper Learning with a Single Noisy Element

Index-based generation can be thought as “properly” learning to generate from language K , while both element-based and set-based generations correspond to “improper” learning as they do not require the generator’s outputs to be languages in $\mathcal{L}$. Our next example shows that index-based generation is too restrictive even for a two-language collection when the adversary’s enumeration contains 1 noisy example.

Example 3.2. Take $U := \mathbb{N} = \{1, 2, 3, \dots\}$, $L_1 := \mathbb{N} \setminus \{1\}$, $L_2 := \mathbb{N} \setminus \{2\}$, and $\mathcal{L} = \{L_1, L_2\}$. If the adversary chooses a complete enumeration of $\mathbb{N}$, then clearly this enumeration is an enumeration with 1 noisy example for both L_1 and L_2 with one noisy example and thus the adversary is free to declare the target language $K = L_1$ or $K = L_2$. However, outputting any index $i_n \in \{1, 2\}$ does not guarantee $L_{i_n} \subseteq K$.

Hence, we mainly focus on element-based and set-based generation in the noisy setting.

3.4 Language Generation under Omission

We also consider the setting where the adversaries omit certain elements of the target K in its enumeration, which further complicates things. We remark that the algorithm of [KM24] needs that the adversary enumerates all the elements of K , and any index-based generator provably fails even when the adversary is allowed to omit one element of K (see [Example 3.3](#)). Thus, we again focus on element-based generation and set-based generation in the setting with omission. We consider the following regimes of omissions in adversary’s enumeration.

Definition 8 (Regimes of Enumeration with Omission). *Fix a language $L \subseteq U$.*

- ▷ *An enumeration of L with finite omissions is an enumeration of $\hat{L} \subseteq L$, s.t., $|L \setminus \hat{L}| < \infty$.*
- ▷ *An enumeration of L with c -omissions is an enumeration of $\hat{L} \subseteq L$, s.t., $\mu_{\text{low}}(\hat{L}, L) \geq 1 - c$.*
- ▷ *An enumeration of L with arbitrary (or infinite) omissions is an enumeration of $\hat{L} \subseteq L$, s.t., $|\hat{L}| = \infty$.*

3.5 Language Generation under Contamination

Now that we have defined both enumeration with noise in [Definition 7](#) and enumeration with omission in [Definition 8](#), we may pair up these definitions and talk about enumeration under contamination, including both noise and omissions. Consider the following examples.

- ▷ *An enumeration of L with finite noise and finite omissions is an enumeration with finite noise of $\hat{L} \subseteq L$ such that $|L \setminus \hat{L}| < \infty$.*
- ▷ *An enumeration of L with $o(1)$ -noise and c -omissions is an enumeration with $o(1)$ -noise of $\hat{L} \subseteq L$ such that $\mu_{\text{low}}(\hat{L}, L) \geq 1 - c$.*
- ▷ *An enumeration of L with c -noise and arbitrary omissions is an enumeration with c -noise of $\hat{L} \subseteq L$ such that $|\hat{L}| = \infty$.*

Generation in the limit when the adversary chooses an enumeration with both noise and omissions means that the generator still needs to be eventually consistent with K under any such enumeration, as defined in [Definition 5](#).

- ▷ An enumeration of L with finite noise and finite omissions is an enumeration with finite noise of $\hat{L} \subseteq L$ such that $|L \setminus \hat{L}| < \infty$.
- ▷ An enumeration of L with $o(1)$ -noise and c -omissions is an enumeration with $o(1)$ -noise of $\hat{L} \subseteq L$ such that $\mu_{\text{low}}(\hat{L}, L) \geq 1 - c$.
- ▷ An enumeration of L with c -noise and arbitrary omissions is an enumeration with c -noise of $\hat{L} \subseteq L$ such that $|\hat{L}| = \infty$.

Separation of Proper and Improper Learning with a Single Omission

Our next result shows that even for simple collections consisting of two languages, and very restricted adversaries that omit only a single element from K , index-based generation is provably impossible.

Example 3.3. Consider the same example as in [Example 3.2](#): $L_1 := \mathbb{N} \setminus \{1\}$, $L_2 := \mathbb{N} \setminus \{2\}$, and $\mathcal{L} = \{L_1, L_2\}$. If the adversary enumerates $3, 4, 5, \dots$, this enumeration is a valid enumeration with 1 element omitted for both L_1 and L_2 . However, neither index $i_n \in \{1, 2\}$ guarantees $L_{i_n} \subseteq K$.

Remark 3.4. We note that [\[BPZ26\]](#)'s approach for generating in the limit with omissions (with no noise) is fragile to noise. Their approach is simple: they show that if a generator generates $\mathcal{L}$ in the limit with the property that $\mathbb{G}$ starts generating after n^* iterations under an enumeration without noise or omissions, where n^* can depend on the choice of the target language $K \in \mathcal{L}$ but does *not* depend on the enumeration of K chosen by the adversary, then $\mathbb{G}$ (without any changes) already generates in the limit with arbitrary omissions. A generator with this property all countable collections was provided by [\[CP25a\]](#). This approach, however fails when there is noise in the enumeration. We demonstrate the failure of this approach using just two noisy examples. Consider the collection $\mathcal{L} = \{K, L_1, L_2, \dots\}$ where

$$\begin{aligned} K &= \mathbb{N}, \\ L_i &= \{-1, -2\} \cup \{1, 2, \dots, i-1\} \cup T, \quad \text{for each } i \in \mathbb{N}, \\ T &= \{\dots, -102, -101, -100\}. \end{aligned}$$

Now suppose target language is $K = \mathbb{N}$ and the adversary's enumeration: $E = (-2, -1, 1, 2, 3, \dots)$. Note that E contains exactly two noisy examples (-2 and -1) with no omissions.

Why their algorithm fails? At iteration $t \geq 2$, the algorithm from [\[CP25a\]](#) incorrectly generates elements from set T rather than from K . This occurs because: The algorithm examines the first n languages consistent with the observed data $\{-1, -2, 1, 2, \dots, n-2\}$. Among these first n languages, only L_{n-1} remains consistent with the observed data. The algorithm then generates the smallest unseen element from L_{n-1} , which is the smallest unseen element from T . Since $T \cap K = \emptyset$, the algorithm never generates elements of the target language K . This demonstrates that noise-free generation strategies cannot be directly applied to noisy settings, even with minimal noise.

3.6 Density of Generators

Besides requiring that a generator to generate in the limit so that hallucinations eventually stop, it is important to ensure that the generator has learnt a non-trivial fraction of the target language and can generate novel elements and does not limit the generation process to a restrictive subset of the target language. This motivates the study of coverage or breadth of a language generation algorithms. Towards that end, [KMV24; CP25a; KMV25] defined some notions of “breadth” which treat the generating algorithm as a set-based generator, and require that its output misses only finitely many elements of the target.⁹

For convenience, we will assume the universe of all possible strings $U = \mathbb{N}$, since for any countable set, we may identify its elements with natural numbers using a fixed canonical ordering of its elements.

[KW25] introduced the following idea for describing the breadth of a generation algorithm. For any two sets $A, B \subseteq \mathbb{N}$, there are two natural ways to describe what fraction of elements of B is covered by A , namely the upper density $\mu_{\text{up}}(A, B)$ and the lower density $\mu_{\text{low}}(A, B)$ as defined in Definition 1. From now on, whenever we talk about densities, we assume the ground set $U = \mathbb{N}$. We may then define the following densities achieved by an element-based generation algorithm.

Definition 9 (Element-based Density). *Let $\mathcal{L}$ be a countable collection of languages in $\mathbb{N}$. Let $\mathbb{G} = (\mathbb{G}_n)_{n \in \mathbb{N}}$ be an element-based generator. Let K be the target language, and $W = \{w_1, w_2, w_3, \dots\}$ be the infinite set of elements that the generator ever outputs in response to the adversary’s enumeration.*

The element-based generator $\mathbb{G}$ achieves element-based upper density ρ if $\mu_{\text{up}}(W, K) \geq \rho$. It achieves element-based lower density ρ if $\mu_{\text{low}}(W, K) \geq \rho$.

While [KW25] also defined a density measure for index-based generators, since we argued that index-based generation is too restrictive for the noisy generation model, we instead consider the following density for set-based generators.

Definition 10 (Set-based Density). *Let $\mathcal{L}$ be a countable collection of languages in $\mathbb{N}$. Let $\mathbb{G} = (\mathbb{G}_n)_{n \in \mathbb{N}}$ be a set-based generator. Let K be the target language, and A_n be the set generated in round n .*

The set-based generator $\mathbb{G}$ achieves set-based upper density ρ if $\limsup_{n \rightarrow \infty} \mu_{\text{low}}(A_n, K) \geq \rho$. It achieves set-based lower density ρ if $\liminf_{n \rightarrow \infty} \mu_{\text{low}}(A_n, K) \geq \rho$.

While we used $\limsup$ and $\liminf$ of the lower set-density to define set-based upper density and set-based lower density, one can also use $\limsup$ and $\liminf$ of the upper set-density to define similar notions. The arguments in the paper can be easily adapted to handle these other notions, but we omit the discussion of them for conciseness of the presentation.

Finally, we remark that there is an even stronger notion of breadth for set-based generation, called approximate breadth by [KMV25]. A set-based generator is said to achieve approximate breadth if there exists a time n^* , such that for all $n \geq n^*$, the output set S_n satisfies $|K \setminus (S_n \cup \{x_1, \dots, x_n\})| < \infty$. Note that any set-based generator that achieves approximate breadth also achieves set-based lower density 1.

⁹To be precise, these works considered more notions of breadth, but they all share a similar viewpoint.

3.7 What Type of Access Does the Generator Have To $\mathcal{L}$?

Before we move on to discuss our main results, let us take a moment and clarify what the learner can do and cannot do, *i.e.*, what is within fair-game of the language generation model.

What the learner cannot do: The learner cannot get access to or make any query about the unknown target language K . For example, it cannot ask “what is the next smallest element of K not enumerated by the adversary so far.”

Moreover, the learner cannot query what the adversary will do in the future. For example, at round n , it cannot ask “what will the adversary enumerate at time n' ” for any $n' > n$. At any round, it only gets to see the elements revealed by the adversary so far.

What the learner can do: We assume the learner is allowed to make any query about the language collection $\mathcal{L}$, as long as it does not involve accessing the unknown K . For example, the following are some well-studied oracles considered in the previous works:

- ▷ (Membership Oracle) Given index i of some language L_i and element $w \in U$, the oracle returns $\mathbb{1}\{w \in L_i\}$.
- ▷ (Subset Oracle) Given indices i, j of two languages L_i and L_j , the oracle returns $\mathbb{1}\{L_i \subseteq L_j\}$.
- ▷ (Intersection Oracle) Given a finite collection of indices I , the oracle returns $\mathbb{1}\{|\cap_{i \in I} L_i| = \infty\}$.

In this work, we will also consider a few more oracles

- ▷ (Density Oracle) Given languages L, K , compute $\mu_{low}(L, K)$.
- ▷ (Density Rate Oracle) Given languages L, K and $\varepsilon > 0$, compute the number of elements m^* until the empirical density $\frac{|L \cap \{x_1, \dots, x_m\}|}{m} \geq \frac{\mu_{low}(L, K)}{1 + \varepsilon}$ is a good approximation of the true density for all $m \geq m^*$.

When we show certain generators exist, we will specify the oracles they need.

4 Algorithmic Templates

In this section we describe two algorithmic templates that we will use extensively in our work.

4.1 Priority-Based Intersection Meta Algorithm

Before presenting our first algorithmic template, it is instructive to first consider an algorithm due to Charikar and Pabbaraju [CP25a] for generation in the limit without contamination.

The [CP25a] Algorithm. At the n -th step, this algorithm considers the input $x_{1:n}$ and the first n languages $L_1, \dots, L_n$ of a given collection $\mathcal{L}$. We first filter for *consistent* languages, *i.e.*, languages $L_{i_n(1)}, L_{i_n(2)}, \dots$ containing all strings $x_1, \dots, x_n$ for $i_n(j) < i_n(j+1)$. Then, the algorithm computes the largest intersection in the order of indices $i_n(1), i_n(2), \dots$ such that the intersection is infinite, *i.e.*, take the intersection of the first J_n filtered languages where J_n is the largest integer such that $|\cap_{j=1}^{J_n} L_{i_n(j)}| = \infty$. We then output an unseen element from this intersection.

Suppose the target language L_{i^*} has index i^* and consider some L_i for $i < i^*$. Either $L_i \supseteq L_{i^*}$, or there is some string $x \in L_{i^*} \setminus L_i$ that is enumerated at some point, so that L_i is no longer consistent. Since L_{i^*} is always consistent, and the consistent languages ordered before L_{i^*} are eventually supersets, the intersection is always taken over a collection containing L_{i^*} so that the algorithm is guaranteed to generate from some infinite subset of L_{i^*} .

Our template algorithm ([Algorithm 1](#)) makes two main changes to the [CP25a] algorithm: (1) we change the order when taking intersections based on the priority of a language and (2) we introduce different stopping rules when taking the intersection.

Re-ordering by Priorities. In the presence of contamination, we cannot completely eliminate candidate languages L_i from consideration if we observe a string $x_n \notin L_i$, since it is unclear if x_n is actually a member of the target language or just noise. Instead, at step n of our algorithm, we assign a priority $P_i^{(n)} \in \mathbb{N}$ to the language L_i . Although we state present priorities abstractly, we should think of incrementing $P_i^{(n)}$ as penalizing a candidate language L_i whenever we observe $x_n \notin L_i$. Then, we take the intersection in increasing order of priorities. Ideally, only supersets of the target language K can be ordered before K , and the intersection we compute will always be a subset of K . This recovers the [CP25a] algorithm when we take the priorities to be $P_i^{(n)} = i$ if L_i is consistent and $P_i^{(n)} = \infty$ once we observe $x_m \notin L_i$ for some $m \leq n$.

Earlier Stopping. If we would like to generate with density, we must avoid taking the intersection with too many languages, or we can end up generating from an infinite subset of the target language with density 0. We thus define an abstract stopping rule that will be instantiated differently depending on the desired guarantee.

Pseudocode. The pseudocode of our meta-algorithm is presented in [Algorithm 1](#). Here we summarize the key notation for the reader's convenience.

1. The priority function $P(L_i, x_{1:n})$ takes a language L_i and the current enumeration $x_{1:n}$ and outputs a priority $P_i^{(n)} \in \mathbb{N} \cup \{\infty\}$.
2. $i_n(j)$ is the index of the j -th ranked language at step n , i.e., $L_{i_n(1)}$ has the smallest priority value $P_{i_n(1)}^{(n)}$ (and thus “highest priority”) at step n and $L_{i_n(2)}$ has the second smallest priority value, etc.
3. The stopping function $J(L_{i_1}, \dots, L_{i_n}, x_{1:n})$ takes a sequence of languages L_{i_j} and the current enumeration $x_{1:n}$ and outputs a stopping index $J_n \in \mathbb{N}$.

Analysis. One key property of the prefix-based intersection meta algorithm ([Algorithm 1](#)) is that the languages that are ranked higher than the target language will eventually stabilize, i.e., the list of such languages eventually stops changing for sufficiently large n . This is formalized in [Lemma 4.1](#) below. We should think of $p \in \mathbb{N}$ below as an upper bound on the priority of the target language.

Lemma 4.1 (Prefix Priority Stabilization). *Suppose [Algorithm 1](#) is executed with input enumeration $x_{1:\infty}$. Let $P_i^{(n)} \in \mathbb{N} \cup \{\infty\}$ be the priority of language L_i computed in step n . Assume $P_i^{(n)}$ is non-decreasing in n and lower bounded by i , i.e., $i \leq P_i^{(n)} \leq P_i^{(n+1)}$ for all $n \geq 1$. For any $p \in \mathbb{N}$, define*

$$P_i^\infty := \lim_{n \rightarrow \infty} P_i^{(n)}, \quad \mathcal{L}(p) := \{L_i : P_i^\infty \leq p\}.$$

Algorithm 1 Meta-Algorithm for Generation with Noise

Input: Countable collection $\mathcal{L} = \{L_1, L_2, \dots\}$; priority function $P : \mathcal{L} \times U^* \rightarrow \mathbb{N} \cup \{\infty\}$ stopping function $J : \mathcal{L}^* \times \Omega^* \rightarrow \mathbb{N}$; enumeration $x_{1:\infty}$

- 1: Let $S_n \leftarrow \{x_1, \dots, x_n\}$ be the set of examples seen in the first n steps
- 2: Let $W_{n-1} \leftarrow \{w_1, \dots, w_{n-1}\}$ be the set of strings output before the n -th step
- 3: **for** $i = 1, 2, \dots, n$ **do**
- 4: Compute priority $P_i^{(n)}$ at step n for language L_i
- 5: **end for**
- 6: Re-order $\{L_1, \dots, L_n\}$ in increasing priority, tie-breaking by index, as $\{L_{i_n(1)}, \dots, L_{i_n(n)}\}$, i.e., for each $j \in [n-1]$, ensure either $P_{i_n(j)}^{(n)} < P_{i_n(j+1)}^{(n)}$ or $P_{i_n(j)}^{(n)} = P_{i_n(j+1)}^{(n)}$ and $i_n(j) < i_n(j+1)$.
- 7: Compute the stopping index J_n .
- 8: Output $\bigcap_{j \leq J_n} L_{i_n(j)}$. ▷ Set-based output
- 9: Output any $w_n \in \bigcap_{j \leq J_n} L_{i_n(j)} \setminus (S_n \cup W_{n-1})$. ▷ Element-based output

Then there is a step n^* such that for all $n \geq n^*$,

- (a) $P_i^{(n)} \leq p$ for all $L_i \in \mathcal{L}(p)$
- (b) $P_i^{(n)} > p$ for all $L_i \notin \mathcal{L}(p)$
- (c) $P_i^{(n+1)} = P_i^{(n)}$ for all $L_i \in \mathcal{L}(p)$

Before proving [Lemma 4.1](#), we provide some intuition for the quantities involved. By the assumption that $P_i^{(n)}$ is a non-decreasing integer sequence indexed by n , we know that the limit P_i^∞ always exists. Since the sequence takes on discrete values, we can further deduce that either $P_i^{(n)} = P_i^\infty$ after some finite time n , or $P_i^\infty = \infty$ and the priority diverges. Assume for now that the priority $P_{i^*}^\infty < \infty$ of the target language L_{i^*} does not diverge. Then $\mathcal{L}(p)$ for $p := P_{i^*}^\infty$ consists of the languages whose priorities are always bounded above by the limiting priority of the target language. [Lemma 4.1](#) states that languages in $\mathcal{L}(p)$ will eventually be ordered before any language not in $\mathcal{L}(p)$. Thus when we take the intersection in [Algorithm 1](#) in increasing order of priorities, we will always consider languages in $\mathcal{L}(p)$ before all other languages. Note that the definition of $\mathcal{L}(p)$ does not depend on a particular step n . [Lemma 4.1](#) simplifies some of the analyses of our algorithms in that it suffices for us to analyze properties of $\mathcal{L}(p)$ rather than analyzing any particular stage of our algorithm.

We are now ready to prove [Lemma 4.1](#).

Proof of [Lemma 4.1](#). First, we note that $P_i^\infty \geq P_i^{(n)} \geq i$ for any $n \geq i$. Hence $\mathcal{L}(p) \subseteq \{L_1, \dots, L_p\}$ and $P_i^{(n)} > p$ for all $n \geq i > p$. Let $n^* \geq 1$ be any integer such that for every $i \in [p]$, either

- (a) $P_i^{(n)} = P_i^\infty \leq p$ for all $n \geq n^*$, or
- (b) $P_i^{(n)} > p$ for all $n \geq n^*$.

This is guaranteed to exist since there are only finitely many limits, each of which is monotonic. The result follows. □

Our goal will be to design a priority function for which the target language L_{i^*} has bounded priority $p = P_{i^*}^\infty < \infty$ so that $L_{i^*} \in \mathcal{L}(p)$. Recall the notation $\text{Cl}(\mathcal{L}') := \bigcap_{L \in \mathcal{L}'} L$ for a collection $\mathcal{L}'$ of languages. If the stopping rule also ensures that the prefix class $\mathcal{H}_n := \{L_{i_n(j)} : j \leq J_n\}$ eventually contains $\mathcal{L}(p)$ and $|\text{Cl}(\mathcal{H}_n)| = \infty$, then the algorithm generates from $\mathcal{L}(p) \subseteq L_{i^*}$. This observation is summarized below.

Corollary 4.2 (Sufficient Condition for Generation). *Suppose the assumptions of Lemma 4.1 hold. If in addition the prefix class $\mathcal{H}_n := \{L_{i_n(j)} : j \leq J_n\}$ satisfies $\mathcal{H}_n \supseteq \mathcal{L}(p)$ and $|\text{Cl}(\mathcal{H}_n)| = \infty$ for all sufficiently large n , then Algorithm 1 generates from $\text{Cl}(\mathcal{L}(p))$.*

Instantiations of Algorithm 1. In this paper, we make use of the meta algorithm Algorithm 1 to prove Theorem 5.1, Theorem 5.4, Theorem 6.11, Theorem 6.14, and Theorem 7.8. We now illustrate a few examples of instantiations of Algorithm 1.

For our algorithms that generate in the limit under vanishing noise (Section 5.1) and constant noise (Section 5.2), we instantiate the priority $P_i^{(n)}$ of language L_i at time step n so that, roughly speaking, $P_i^{(n)}$ increases every time the empirical noise rate $R(L_i; x_{1:n})$ exceeds some threshold c_i . For a vanishing noise rate, we can set c_i to be any positive sequence with small sum $\sum_i c_i < \varepsilon$. This ensures that we prioritize languages that have very small empirical error and the total error is at most an ε -fraction of the input enumeration. For a constant noise rate c , we set $c_i = c$ for all i , so that we prioritize languages whose noise rate eventually falls below c . The stopping condition in these two cases is identical to the [CP25a] algorithm.

Our algorithms for generation with density under M -bounded enumerations with vanishing noise (Section 7) are similar to the algorithm for generation under vanishing noise with two key distinctions. In addition to incrementing the priority $P_i^{(n)}$ of language L_i at step n when the empirical noise rate exceeds some threshold, we also increment $P_i^{(n)}$ if the current input violates the M -boundedness condition with respect to L_i . This ensures we de-prioritize languages for which the input enumeration cannot be M -bounded. We also require a more involved stopping rule which ensures that the intersection we take is dense with respect to every language over which the intersection is taken over. This ensures that if the target language is included in the intersection, then we will always generate from some dense subset of the target language.

4.2 Finite Expansion Sub-Routine

Next, we present a conceptually simple, but quite useful subroutine that we will use throughout, especially in settings where the adversary is restricted to finite contamination of the input. To show why this sub-routine is useful, we state a preliminary result in this section showing that it can be used to achieve generation in the limit, under finite contamination of the input stream. In subsequent sections we will illustrate more involved use-cases of this idea.

Algorithm 2 Sub-Routine for Finite Expansion

Input: Countable collection $\mathcal{L} = \{L_1, L_2, \dots\}$

1: Output $\mathcal{L} := \{L_{A,B} := L \cup A \setminus B : L \in \mathcal{L}, A \subseteq U \setminus L, B \subseteq L, |A| < \infty, |B| < \infty\}$.

Lemma 4.3. *Suppose G is an element-based (set-based) generator that generates from arbitrary countable collection in the limit under enumeration without noise or omissions. Let $\mathcal{L}$ be a countable collection, and $\widetilde{\mathcal{L}}$ be the expanded countable collection constructed from Algorithm 2. Suppose $x_{1:\infty}$ is an enumeration of some target language $K \in \mathcal{L}$ with finite noise and finite omissions. Then, if we apply G to $\widetilde{\mathcal{L}}$, it will generate from K in the limit under the aforementioned enumeration.*

Proof. We start with element-based generators. Notice that, since $x_{1:\infty}$ contains finitely many noisy examples and omits finitely many elements from K , it corresponds to an enumeration of $K_{A,B} \in \mathcal{L}$ without noise or omissions, where A is the set of noisy elements and B is the set of omitted elements. Thus, G executed on $x_{1:\infty}$ and $\widetilde{\mathcal{L}}$ generates in the limit from $K_{A,B}$. Hence, there exists some $n^* \in \mathbb{N}$ such that for all $n \geq n^*$ its output is a fresh element of $K_{A,B}$. Next, notice that since $|A| < \infty$ it must be the case that there is some n' such that for all $n \geq n'$ the output of the algorithm is a fresh element of $K_{A,B} \setminus A$. Since $(K_{A,B} \setminus A) \subseteq K$ we have shown that G generates K in the limit.

Similarly, if G is a set-based generator that generates in the limit from $K_{A,B}$, there exists some $n^* \in \mathbb{N}$ such that for all $n \geq n^*$, the output set A_n is a subset of $K_{A,B}$. Again, since $|A| < \infty$ and $x_{1:\infty}$ is an enumeration of $K_{A,B} = K \cup A \setminus B$ without noise or omissions, there exists some n' such that for all $n \geq n'$, we have $K_{A,B} \setminus \{x_1, \dots, x_n\} \subseteq K_{A,B} \setminus A \subseteq K$. Since the output set satisfies $A_n \subseteq K_{A,B} \setminus \{x_1, \dots, x_n\} \subseteq K$ for all $n \geq \max\{n^*, n'\}$, G generates K in the limit. $\square$

To illustrate the usefulness of this sub-routine and the above result, we get as an immediate corollary that there exists an algorithm that generates in the limit under finite contamination, using only *membership oracle access* to the underlying language collection $\mathcal{L}$. This resolves in the affirmative the open question of Raman and Raman [RR25], who asked for generators with membership oracle access to $\mathcal{L}$ and finite noise (and without omissions). First, we state a result from Kleinberg and Mullainathan [KM24] that we utilize in our proof.

Lemma 4.4 (Generation with Membership Oracle Access under Uncontaminated Input [KM24]). *There is an algorithm that generates in the limit for every countable collection $\mathcal{L}$ with only membership oracle access to $\mathcal{L}$ when the adversarial stream does not include any contaminated examples.*

We are now ready to state and prove our result.

Corollary 4.5 (Generation with Membership Oracles under Finite Omissions). *There is an algorithm that generates in the limit given only membership oracle access to $\mathcal{L}$ under adversarial streams that contain finite amount of contamination.*

Proof. We will instantiate Algorithm 2 with the algorithm of Kleinberg and Mullainathan [KM24], stated in Lemma 4.4. Notice that if we have membership oracle access to $\mathcal{L}$, we can implement membership oracle access to $\widetilde{\mathcal{L}}$, by keeping track of the way we enumerate languages in $\widetilde{\mathcal{L}}$. In other words, if $\widetilde{L}_i$ is the i -th language of $\widetilde{\mathcal{L}}$ we can maintain a mapping to $L_{j,A,B} \in \mathcal{L}$, for some j, A, B . Then, the query “is $u_\ell \in \widetilde{L}_i$ ” is answered by checking if $u_\ell \in L_j \cup A \setminus B$; since we have membership access to L_j , it suffices to check if $u_\ell \in L_j$ and then, since we have explicit description of the (finite) sets A, B , we can check if $u_\ell \in A, u_\ell \in B$. Thus, since we can implement a membership oracle for $\widetilde{\mathcal{L}}$ we can run the algorithm from Lemma 4.4 on it. Then, Lemma 4.3 guarantees that this algorithm achieves generation in the limit. $\square$

Later in the paper, we will also use the finite expansion sub-routine in [Algorithm 2](#) to prove [Theorem 6.18](#), a result about element-based density under finite noise and omissions.

We will also consider a slightly different subroutine that only expands by adding a finite set to the existing languages, as opposed to the one in [Algorithm 2](#) that allows both addition of finitely many elements and deletion of finitely many elements.

Algorithm 3 Alternative Sub-Routine for Finite Expansion

Input: Countable collection $\mathcal{L} = \{L_1, L_2, \dots\}$

1: Output $\widetilde{\mathcal{L}} := \{L_A := L \cup A : L \in \mathcal{L}, A \subseteq U \setminus L, |A| < \infty\}$.

Lemma 4.6. *Suppose $\mathbb{G}$ is an element-based (set-based) generator that generates from arbitrary countable collection in the limit under enumeration without noise and with potentially infinite omissions. Let $\mathcal{L}$ be a countable collection, and $\widetilde{\mathcal{L}}$ be the expanded countable collection constructed from [Algorithm 3](#). Suppose $x_{1:\infty}$ is an enumeration of some target language $K \in \mathcal{L}$ with finite noise and potentially infinite omissions. Then, if we apply $\mathbb{G}$ to $\widetilde{\mathcal{L}}$, it will generate from K in the limit under the aforementioned enumeration.*

Proof. The proof is almost identical to that of [Lemma 4.3](#).

We start with element-based generators. Notice that, since $x_{1:\infty}$ contains finitely many noisy examples outside of K , it corresponds to an enumeration of $K_A \in \widetilde{\mathcal{L}}$ without noise and with potentially infinite omissions, where A is the set of noisy elements and every element of A is enumerated in $x_{1:\infty}$. Thus, $\mathbb{G}$ executed on $x_{1:\infty}$ and $\widetilde{\mathcal{L}}$ generates in the limit from K_A . Hence, there exists some $n^* \in \mathbb{N}$ such that for all $n \geq n^*$ its output is a fresh element of K_A . Next, notice that since $|A| < \infty$ it must be the case that there is some n' such that for all $n \geq n'$ the output of the algorithm is a fresh element of $K_A \setminus A = K$. This shows that $\mathbb{G}$ generates K in the limit.

Similarly, if $\mathbb{G}$ is a set-based generator that generates in the limit from K_A , there exists some $n^* \in \mathbb{N}$ such that for all $n \geq n^*$, the output set A_n is a subset of K_A . Again, since $|A| < \infty$ and every element of A is enumerated in $x_{1:\infty}$, there exists some n' such that for all $n \geq n'$, we have $K_A \setminus \{x_1, \dots, x_n\} \subseteq K_A \setminus A = K$. Since the output set satisfies $A_n \subseteq K_A \setminus \{x_1, \dots, x_n\} \subseteq K$ for all $n \geq \max\{n^*, n'\}$, $\mathbb{G}$ generates K in the limit. □

We will later utilize the above sub-routine and [Lemma 4.6](#) to show a result similar to the upper density guarantee for indexed-based generation under enumeration without noise or omissions proved by [\[KW25\]](#). In [Theorem 6.1](#), we will show that under finite noise and infinite omissions, there exists set-based generator that achieves the best possible set-based upper density.

5 Generation in the Limit under Contamination

In this section, we provide several results in the model where the adversary can contaminate the input enumeration with noise and omissions.

5.1 Generation with Vanishing Noise Rate and Arbitrary Omissions

Our first result shows that under the mild assumption that the noise rate of the enumeration converges to zero, all countable collections become generable.

We remark that even though the noise rate converges to zero, the amount of noisy examples the adversary can add to the enumeration is still infinite. Hence, this result significantly strengthens the result of [RR25], which only showed that all collections remain generable when the adversary introduces a *finite* number of noisy examples.

Theorem 5.1. *There is a generator $\mathbb{G}$ that, for any collection $\mathcal{L}$, target language $K \in \mathcal{L}$, given an enumeration of K with $o(1)$ -noise and arbitrary omissions, $\mathbb{G}$ generates in the limit from K .*

Pseudocode. The algorithm for Theorem 5.1 is Algorithm 4 instantiated with threshold parameters $c_i := \frac{1}{2^{i+1}}$.

We can view our algorithm as a sub-template of our generic meta-algorithm Algorithm 1, where the priorities are defined by thresholds and the stopping rule is based on the size of the intersection of the languages in the ordered prefix. The changes from Algorithm 1 are highlighted in blue.

Algorithm 4 Algorithm for Theorems 5.1 and 5.4

Input: Countable collection $\mathcal{L} = \{L_1, L_2, \dots\}$; thresholds $c_1, c_2, \dots \in (0, 1)$; enumeration $x_{1:\infty}$

- 1: Let $S_n \leftarrow \{x_1, \dots, x_n\}$ be the set of examples seen in the first n steps
- 2: Let $W_{n-1} \leftarrow \{w_1, \dots, w_{n-1}\}$ be the set of strings output before the n -th step
- 3: **for** $i = 1, 2, \dots, n$ **do**
- 4: Compute the smallest $N_i^{(n)}$ such that L_i is consistent with $x_{1:m}$ for each $N_i^{(n)} \leq m \leq n$

$$N_i^{(n)} \leftarrow \begin{cases} \min \{N \geq 1 : \forall m \in \{N, \dots, n\}, R(L_i; x_{1:m}) \leq c_i\} , & R(L_i; x_{1:n}) \leq c_i , \\ n + 1 , & \text{else.} \end{cases}$$

- 5: Assign the language L_i a priority of $P_i^{(n)} \leftarrow i + N_i^{(n)}$
- 6: **end for**
- 7: Re-order $\{L_1, \dots, L_n\}$ in increasing priority, tie-breaking by index, as $\{L_{i_n(1)}, \dots, L_{i_n(n)}\}$, i.e., for each $j \in [n - 1]$, ensure either $P_{i_n(j)}^{(n)} < P_{i_n(j+1)}^{(n)}$ or $P_{i_n(j)}^{(n)} = P_{i_n(j+1)}^{(n)}$ and $i_n(j) < i_n(j + 1)$.
- 8: Compute the largest index J_n such that the intersection in the re-ordering up to $L_{i_n(J_n)}$ is infinite

$$J_n \leftarrow \sup \left\{ \bar{j} \geq 1 : \left| \bigcap_{j=1}^{\bar{j}} L_{i_n(j)} \right| = \infty \right\} .$$

- 9: Output any $w_n \in \bigcap_{j \leq J_n} L_{i_n(j)} \setminus (S_n \cup W_{n-1})$.
-

Analysis. Algorithm 4 generates under both vanishing noise, omissions, or constant noise (assuming the c -constant noise generation property holds) when instantiated with an appropriate choice of thresholds.

Before presenting a useful lemma, recall the following notation related to the priorities from our generic meta-algorithm [Algorithm 1](#). For a fixed $p \geq 1$,

$$P_i^\infty := \lim_{n \rightarrow \infty} P_i^{(n)}, \quad \mathcal{L}(p) := \{L_i : P_i^\infty \leq p\}, \quad \text{and} \quad \text{Cl}(\mathcal{L}(p)) := \bigcap_{L \in \mathcal{L}(p)} L.$$

Lemma 5.2 (Sufficient Condition to Generate using [Algorithm 4](#)). *Consider executing [Algorithm 4](#) with some arbitrary choice of thresholds c_i . Fix $p \geq 1$ and suppose that the set $\mathcal{L}(p)$ satisfies $|\text{Cl}(\mathcal{L}(p))| = \infty$. Then [Algorithm 4](#) generates from $\text{Cl}(\mathcal{L}(p))$ in the limit.*

Proof. We argue using [Corollary 4.2](#). To do so, we first note that the priorities $P_i^{(n)}$ are non-decreasing by construction. Define the prefix class $\mathcal{L}_n = \{L_{i_n(j)} : j \leq J_n\}$. By the definition of [Algorithm 4](#), this set always has infinite intersection. Thus to apply [Corollary 4.2](#), it suffices to show that $\mathcal{L}(p) \subseteq \mathcal{L}_n$ for all sufficiently large n .

To see this, first note that the priorities $P_i^{(n)}$ of every L_i is non-decreasing by definition. Hence by [Lemma 4.1](#), there is some $n^* \in \mathbb{N}$ such that $\mathcal{L}(p)$ is ordered before all other languages for every $n \geq n^*$. But $|\text{Cl}(\mathcal{L}(p))| = \infty$ by assumption so that $\mathcal{L}_n$ must contain $\mathcal{L}(p)$ for $n \geq n^*$ by the definition of the stopping rule in [Algorithm 4](#). $\square$

We are now ready to prove [Theorem 5.1](#).

Proof of Theorem 5.1. Similar to the case of finite noise rate, our strategy is to apply [Lemma 5.2](#) to argue that [Algorithm 4](#) generates from $\mathcal{L}$, but this time with threshold parameters $c_i := \frac{1}{2^{i+1}}$.

Write $K = L_{i^*}$. Set $p := P_{i^*}^\infty$ and note that $p < \infty$ since by the $o(1)$ -noise rate assumption, $R(L_{i^*}; x_{1:n}) \leq c_{i^*}$ for all sufficiently large n . If we show that $|\text{Cl}(\mathcal{L}(p))| = \infty$, then we can apply [Lemma 5.2](#) to conclude that [Algorithm 4](#) generates from $\text{Cl}(\mathcal{L}(p)) \supseteq K$.

By [Lemma 4.1](#), since the priorities are non-decreasing, there is some $\bar{n} \in \mathbb{N}$ such that for $n \geq \bar{n}$, the priorities $P_i^{(n)}$ of languages $L_i \in \mathcal{L}(p)$ remain constant. For $n \geq \bar{n}$, we see that

$$\begin{aligned} |\text{Cl}(\mathcal{L}(p))| &\geq |S_n \cap \text{Cl}(\mathcal{L}(p))| \geq n \left(1 - \sum_{L_i \in \mathcal{L}(p)} R(L_i; x_{1:n}) \right) \\ &\geq n \left(1 - \sum_{i \geq 1} \frac{1}{2^{i+1}} \right) \geq \frac{n}{2}. \end{aligned}$$

In particular, $|\text{Cl}(\mathcal{L}(p))| = \infty$ as desired. $\square$

Remark 5.3. While [Algorithm 4](#) uses a priority that depends on how long a language's empirical noise rate has stayed below the threshold c_i , which significantly simplifies our analysis by using the stable prefix property guaranteed by [Lemma 4.1](#), it is unclear if some notion of priority similar to that used in [Algorithm 4](#) is inherently needed. Indeed, in the appendix, we show in [Theorem B.2](#) that [Algorithm 10](#) still achieves generation in the limit for any countable collection under vanishing noise rate, and does not use this notion of priority in the algorithm. In particular, [Algorithm 10](#) will not have the stable prefix property described in [Lemma 4.1](#).

In this paper, we repeatedly make use of similar notions of priorities as that in [Algorithm 4](#), e.g., in [Algorithms 6, 7 and 9](#). We believe it is an interesting question to investigate if there exist generation algorithms that achieve the same guarantees and do not use this notion of priority we defined.

5.2 Characterization of Generation with Constant Noise Rate and Arbitrary Omissions

Next, we investigate the setting where the adversaries can adopt an enumeration with constant noise rate and arbitrary omissions. While not all countable collections are generable in this setting, our next result provides a complete characterization of generation in the limit and describes the collections for which generation in the limit is not achievable.

Theorem 5.4. *Fix any constant $c \in (0, 1)$. A collection $\mathcal{L}$ is generable with c -noise and arbitrary omissions if and only if the following c -constant noise generation property holds.*

Condition 1 (c -constant noise generation property). *For every non-empty finite subcollection $\mathcal{L}' \subseteq \mathcal{L}$ and every enumeration $x_{1:\infty}$, either*

- (a) *there is some $L' \in \mathcal{L}'$ such that $R(L'; x_{1:n}) > c$ infinitely often, or*
- (b) *the intersection $|\text{Cl}(\mathcal{L}')| = \infty$ is infinite.*

Example 5.5. For any $k \in \mathbb{N}$, there are simple enumerations and collections of just k languages $L_1, \dots, L_k$ that do not satisfy [Condition 1](#) for $c = 1/k$. For instance, for $i \in [k]$, we can take $L_i := \{n \in \mathbb{N} : n \bmod k = i\}$ to be the positive integers with remainder i modulo k , and $x_{1:\infty}$ to be the canonical enumeration of the integers.

Necessity of [Condition 1](#) (Constant Noise Generation Property)

We first prove the necessity of [Condition 1](#).

Proof of Theorem 5.4 (Necessity). Suppose that the c -constant noise generation property does not hold. Then, there is some non-empty finite subcollection $\mathcal{L}' \subseteq \mathcal{L}$ and enumeration $x_{1:\infty}$ such that for every $L' \in \mathcal{L}'$, $R(L'; x_{1:n}) \leq c$ for all sufficiently large n , but the intersection $|\text{Cl}(\mathcal{L}')| < \infty$ is finite. Note this implies that $|\mathcal{L}'| \geq 2$.

Suppose towards a contradiction that $\mathcal{L}$ is generated by some generator $\mathbb{G}$. Order $\mathcal{L}' = \{L_1, \dots, L_k\}$ for some $2 \leq k < \infty$. Since $x_{1:\infty}$ is a valid enumeration with c -noise of each $L_j \in \mathcal{L}'$, $\mathbb{G}$ must simultaneously generate from all L_j for $j \in [k]$ given the same enumeration $x_{1:\infty}$. We will argue that this yields a contradiction.

Let $S_n := \{x_1, \dots, x_n\}$, $w_n := \mathbb{G}_n(x_{1:n})$, and $W_n := \{w_1, \dots, w_n\}$ denote the set of input and generated strings up to time n . By assumption, for each $j \in [k]$, there is some n_j^* such that $w_n \in L_j \setminus (S_n \cup W_{n-1})$ for all $n \geq n_j^*$. Take $n^* := \max_{j \in [k]} n_j^*$. Then

$$w_n \in \bigcap_{j \in [k]} (L_j \setminus (S_n \cup W_{n-1})) = \left(\bigcap_{j \in [k]} L_j \right) \setminus (S_n \cup W_{n-1}) \subseteq \text{Cl}(\mathcal{L}') \setminus W_{n-1}$$

for all $n \geq n^*$. However, this is impossible for $n > n^* + |\text{Cl}(\mathcal{L}')|$ as G has exhausted all elements in the finite intersection. $\square$

Next, we design an element-based generation algorithm to show that the constant noise generation property suffices to guarantee generation. Interestingly, the same algorithm achieves generation with unknown vanishing error as well as generation with arbitrary omissions.

Sufficiency of [Condition 1](#) (Constant Noise Generation Property)

Next, we prove the sufficiency of [Condition 1](#).

Pseudocode. The pseudocode is [Algorithm 4](#) instantiated with uniform thresholds $c_i = c$.

Analysis. We are now ready to prove the other direction of our characterization of constant noise generation ([Theorem 5.4](#)).

Proof of [Theorem 5.4](#) (Sufficiency). Suppose that the c -constant noise generation property holds. We will apply [Lemma 5.2](#) to prove that [Algorithm 4](#) can generate when instantiated with thresholds $c_1 = c_2 = \dots = c$.

Write $K = L_{i^*}$ and define $p := P_{i^*}^\infty$. By definition, there is some n_{i^*} such that $R(L_{i^*}; x_{1:n}) \leq c$ for $n \geq n_{i^*}$. Thus for $n \geq n_{i^*}$, $P_{i^*}^{(n)} = p < \infty$ remains constant moving forwards. By [Lemma 5.2](#), it suffices to check that $|\text{Cl}(\mathcal{L}(p))| = \infty$ in order to ensure that [Algorithm 4](#) generates from $\text{Cl}(\mathcal{L}(p)) \supseteq K$. Now, $|\mathcal{L}(p)| \leq p$ is finite and by [Lemma 4.1](#), the priorities $P_i^{(n)}$ of all its members $L_i \in \mathcal{L}(p)$ stabilize after some finite time. By the definition of the priorities in [Algorithm 4](#), this means that no $L_i \in \mathcal{L}(p)$ can satisfy $R(L_i, x_{1:n}) > c$ infinitely often. But then by the c -constant noise generation property, $|\text{Cl}(\mathcal{L}(p))| = \infty$.

This concludes the proof. $\square$

6 Generation with Density under Contamination

We now shift our attention to generation with breadth using the notions of generation with density defined in [Section 3.6](#). We will first discuss our results for generation with set-based densities in [Section 6.1](#) and then for generation with element-based densities in [Section 6.2](#).

6.1 Generation with Set-Based Density under Contamination

We first consider set-based generators. Recall that under this definition ([Definition 10](#)), the generator outputs a set in every timestep and we measure the set-based density using the sequence of set densities of the output sets at every timestep in the target language K . The validity requirement asks the generator to eventually output a subset of K for all large enough n , and the generator achieves set-based lower (upper) density if for all large enough n (infinitely often), the output set A_n has good lower density in K .

6.1.1 Set-Based Upper Density under Finite Contamination

In this section, we prove a result similar to the index-based upper density guarantee proved by [KW25]. While we have previously argued in Example 3.2 that even with 1 noisy example, index-based generator can fail to generate in the limit, we will show that there exists a set-based generator that achieves the best possible set-based upper density. Our algorithm is inspired by the approach of Kleinberg and Wei [KW25] who used a “fall-back” strategy to ensure that, infinitely often, the output is (exactly) the target language, but requires some important new ideas as now the adversary’s enumeration is contaminated by both noise and omissions. First, we use prove a result showing set-based upper density is achievable in the noiseless setting where the adversary’s enumeration contains potentially infinitely omissions but *no noise*. Then, we invoke the finite expansion sub-routine in Algorithm 3 and reduce the setting with finite noise and c -omissions to the setting without noise and with c -omissions.

To design an algorithm achieving set-based upper density for the noiseless setting with infinite omissions, at each round we consider an active set $\mathcal{A}_n$ consisting of all languages up to L_n . Then, we consider the longest prefix $\mathcal{L}_n$ of languages of $\mathcal{A}_n$ that satisfies

- ▷ the intersection of all languages in $\mathcal{L}_n, \text{Cl}(\mathcal{L}_n)$, is infinite;
- ▷ $\mathcal{L}_n$ also appears as a prefix of $\mathcal{L}_{n-1}$.

The second requirement is exactly the fall-back strategy we used: whenever some language L_i leaves the active set at round n , we only consider prefixes of $\mathcal{A}_n$ consisting of languages that come before L_i . In other words, we only consider prefixes of the longest prefix of languages of $\mathcal{A}_n$ that did not change compared to $\mathcal{A}_{n-1}$. The strategy, implemented by Algorithm 5, ensures that we are neither forever “overshooting” by intersecting a long prefix, thus sacrificing density, nor infinitely often “undershooting” by intersecting a small prefix, thus sacrificing correctness.

Theorem 6.1. *For all countable collections of languages $\mathcal{L}$, there is a set-based generator that generates in the limit from $\mathcal{L}$ and achieves set-based upper density at least $1 - c$ under adversaries that use an enumeration with finite noise and c -omissions.*

In particular, if the adversary uses an enumeration with finite noise and finite omissions, there is a set-based generator that generates in the limit from $\mathcal{L}$ and achieves set-based upper density 1.

Remark 6.2. As a nice feature, the set-based generator in Theorem 6.1 does not assume the knowledge of the amount of omissions in the adversary’s enumeration.

To prove Theorem 6.1, we will first prove the following result in the noiseless setting, and then make use of the alternative finite expansion sub-routine in Algorithm 3 and Lemma 4.6.

Proposition 6.3. *For all countable collections of languages $\mathcal{L}$, there is a set-based generator that generates in the limit from $\mathcal{L}$ and achieves set-based upper density at least $1 - c$ under enumerations without noise and with c -omissions.*

Proof of Proposition 6.3. We will show that Algorithm 5 works for Proposition 6.3.

Let $K = L_{i^*} \in \mathcal{L}$ be the target language, and $x_{1:\infty}$ be an enumeration of K without noise and with c -omissions. Let $\hat{K} \subseteq K$ be the subset of K such that $x_{1:\infty}$ is an enumeration of $\hat{K}$ without noise or omissions. By assumption, $\mu_{\text{low}}(\hat{K}, K) \geq 1 - c$.

Algorithm 5 Algorithm for [Theorem 6.1](#)

Input: Countable collection $\mathcal{L} = \{L_1, L_2, \dots\}$; enumeration $x_{1:\infty}$

- 1: Let $S_n \leftarrow \{x_1, \dots, x_n\}$ be the set of examples seen in the first n steps
 - 2: Let $\mathcal{A}_n \leftarrow \{L_i : S_n \subseteq L_i, 1 \leq i \leq n\}$ be the subcollection of languages consistent with the seen examples.
 - 3: If $n = 1$, set $\mathcal{L}_n \leftarrow \emptyset$.
 - 4: If $n \geq 2$, compute the largest common prefix $\mathcal{L}_n$ of $\mathcal{A}_{n-1}$ and $\mathcal{A}_n$ with infinite intersection, i.e.,
$$\mathcal{L}_n \leftarrow \{L_i \in \mathcal{A}_n : \{L_1, \dots, L_i\} \cap \mathcal{A}_n = \{L_1, \dots, L_i\} \cap \mathcal{A}_{n-1}, \text{ and } |\text{Cl}(\{L_1, \dots, L_i\} \cap \mathcal{A}_n)| = \infty\}.$$
 - 5: Output $\text{Cl}(\mathcal{L}_n) \setminus S_n$.
-

Proof of Generation in the Limit. First, we show that [Algorithm 5](#) achieves generation in the limit. We observe that for $n \geq i^* + 1$, we always have $K \in \mathcal{A}_n$. Note that if $S_n \not\subseteq L_i$, then $S_{n+1} \not\subseteq L_i$. Thus, eventually, the prefix of languages in $\mathcal{A}_n$ that come before K stabilizes, since once some language leaves the set $\mathcal{A}_n$, it will never enter it again. Let $n^* \in \mathbb{N}$ be the time such that for all $n \geq n^*$, $\mathcal{A}_n \cap \{L_i : i \leq i^*\}$ is fixed. Denote this collection as $\mathcal{A}_\infty$. Note that $K \in \mathcal{A}_\infty$. We also note that every language in $\mathcal{A}_\infty$ has to be a superset of $\hat{K}$, since each of them has to be consistent with $x_{1:\infty}$, which is an enumeration of $\hat{K}$ without noise or omissions. As a result, $\hat{K} \subseteq \text{Cl}(\mathcal{A}_\infty)$ and thus the largest common prefix $\mathcal{L}_n$ of $\mathcal{A}_{n-1}$ and $\mathcal{A}_n$ satisfies $\mathcal{A}_\infty \subseteq \mathcal{L}_n$ for $n \geq n^* + 1$. This finishes the proof of generation in the limit, as for $n \geq n^* + 1$, $\text{Cl}(\mathcal{L}_n) \subseteq \text{Cl}(\mathcal{A}_\infty) \subseteq K$.

Proof of Density Guarantee. Next, we show that infinitely often, [Algorithm 5](#) outputs a set with lower set density at least $1 - c$. At time n , consider $\mathcal{L}_n$. Either $\hat{K} \subseteq \text{Cl}(\mathcal{L}_n)$, or there exists $L \in \mathcal{L}_n$ such that $\hat{K} \not\subseteq L$. In the first case, we readily get $\mu_{\text{low}}(\text{Cl}(\mathcal{L}_n) \setminus S_n, K) \geq \mu_{\text{low}}(\hat{K} \setminus S_n, K) \geq 1 - c$. Thus, we now assume we are in the second case.

List the languages in $\mathcal{L}_n$ in order as $\mathcal{L}_n = \{L_{i_n(1)}, \dots, L_{i_n(t)}\}$ where $t = |\mathcal{L}_n|$. For $n \geq n^* + 1$, we know that $\mathcal{A}_\infty$ appears as a nonempty prefix of $\mathcal{L}_n$, and moreover every language in $\mathcal{A}_\infty$ is a superset of $\hat{K}$. Let L be the first language in $\mathcal{L}_n$ which is not a superset of $\hat{K}$, which exists by the assumption. Since $L \not\subseteq \hat{K}$ and $x_{1:\infty}$ enumerates $\hat{K}$ without omissions, there exists some finite time $n' > n$ such that L is no longer consistent with $S_{n'}$. By our falling back strategy used in the construction of $\mathcal{L}_n$, at time n' , we set $\mathcal{L}_{n'}$ to be the prefix of languages of $\mathcal{L}_n$ coming before L . Since we assumed that L is the first language in $\mathcal{L}_n$ which is not a superset of $\hat{K}$, we know that every language in $\mathcal{L}_{n'}$ is a superset of $\hat{K}$, and thus $\hat{K} \subseteq \text{Cl}(\mathcal{L}_{n'})$. This means that infinitely often, the output set $\text{Cl}(\mathcal{L}_n) \setminus S_n$ achieves lower set density at least $1 - c$ in K , concluding the proof. $\square$

Now we are ready to prove [Theorem 6.1](#) by invoking [Algorithm 3](#) and [Lemma 4.6](#).

Proof of Theorem 6.1. Consider the following generation algorithm.

1. The Algorithm. First, we construct the expanded collection $\widetilde{\mathcal{L}}$ using the expansion subroutine from [Algorithm 3](#). Recall that

$$\widetilde{\mathcal{L}} := \{L_A := L \cup A : L \in \mathcal{L}, A \subseteq U \setminus L, |A| < \infty\}.$$

Then, we execute [Algorithm 5](#) on the collection $\widetilde{\mathcal{L}}$ and the enumeration $x_{1:\infty}$. Since $x_{1:\infty}$ is a valid enumeration of an unknown target language $K \in \mathcal{L}$, with finite noise set A and c -omission,

there exists $\hat{K} \subseteq K$ such that $x_{1:\infty}$ is an enumeration of $\hat{K} \cup A$ without noise or omissions, and $\mu_{\text{low}}(\hat{K}, K) \geq 1 - c$. Thus, $x_{1:\infty}$ corresponds to an enumeration for the language K_A without noise and with c -omission, where we used that $\mu_{\text{low}}(\hat{K} \cup A, K_A) = \mu_{\text{low}}(\hat{K}, K)$ for any finite set A .

2. Proof of Generation in the Limit. By [Proposition 6.3](#), [Algorithm 5](#) applied to $\tilde{\mathcal{L}}$ and $x_{1:\infty}$ generates in the limit from K_A . By [Lemma 4.6](#), we know that the same algorithm generates in the limit from K .

3. Proof of Density Guarantee. If we denote the output set of [Algorithm 5](#) applied to $\tilde{\mathcal{L}}$ and $x_{1:n}$ as A_n , by [Proposition 6.3](#), we know that $\limsup_{n \rightarrow \infty} \mu_{\text{low}}(A_n, K) = \limsup_{n \rightarrow \infty} \mu_{\text{low}}(A_n, K_A) \geq 1 - c$, where we used that $K_A = K \cup A$ differs from K only by a finite set. $\square$

Theorem 6.4. *There exists a countable collection of languages $\mathcal{L}$, such that no set-based generator can generate in the limit and achieve set-based upper density at least $1 - c + \varepsilon$ for any $\varepsilon > 0$, under adversaries that use an enumeration with c -omissions without noise.*

Proof. Consider a two language collection $L_1, L_2 \subseteq U$ such that $L_1 \subseteq L_2$ and $\mu_{\text{low}}(L_1, L_2) = \mu_{\text{up}}(L_1, L_2) = 1 - c$. Consider the canonical enumeration of L_1 without noise and without omissions. Note that this enumeration is a valid enumeration for both L_1 and L_2 with c -omissions without omissions. Therefore, any set-based generator that generates in the limit from the target language K has to generate from L_1 in the limit. Since L_1 has upper density $1 - c$ in L_2 , the set-based generator achieves set-based upper density at most $1 - c$ if the adversary sets $K = L_2$. $\square$

6.1.2 Characterization of Set-Based Lower Density under Finite Contamination

We now shift our attention to obtaining density guarantees that hold for all timesteps beyond a finite number of rounds. Recall that our results in [Theorem 6.1](#) give weaker density guarantees: for infinitely many timesteps the set outputted by the generator has good density in K ; here we ask that the set always has good density, except perhaps for a finite set of timesteps. Naturally, since the density requirement here is significantly stronger than in [Theorem 6.1](#) one would expect that the results we get are weaker. Indeed, even in the absence of noise, [\[KW25\]](#) showed that this type of guarantee is only achievable if the collection does not contain *infinite perfect towers*, a technical condition they introduced. Importantly, this condition is *not* satisfied by all countable collections of languages. Given this result, the main question we aim to understand is whether injecting noisy elements in the enumeration and omitting elements from the target hinges our ability to achieve this type of density guarantee compared to the noiseless setting. Perhaps surprisingly, the answer is that achieving this guarantee in the noisy setting is significantly harder than in the noiseless setting, even when the adversary is restricted to *finite* amount of noise. Our main result for the finite noise and omissions case is a complete characterization of when this type of generation is achievable. As an immediate corollary of our result, we show that there are collections consisting of just two languages in which the generator cannot achieve any non-trivial density guarantee. To compare that with the noiseless setting, recall that the much stronger requirement of *identification in the limit* is achievable for all finite collections of languages. Essentially, an informal interpretation of our result is that this type of generation is possible if all pairs of languages that are not dense in each other are “infinitely separated”.

We now proceed to the formal statement of our result.

Theorem 6.5 (Characterization of Set-Based Lower Density Generation with Finite Contamination). *A countable collection of languages $\mathcal{L}$ is generable in the limit with set-based lower density $c > 0$ under finite noise and finite omissions if and only if for all $L, L' \in \mathcal{L}$ with $|L \setminus L'| < \infty$ it holds that $\mu_{\text{low}}(L, L') \geq c$.*

Example 6.6. As a simple application of the above characterization, consider the following two language collection $\mathcal{L} = \{L_1, L_2\}$, where L_1 is the set of even numbers and L_2 is the set of all natural numbers. Then, by Theorem 6.5, under finite noise and finite omissions, there exists a set-based generator achieving set-based lower density $1/2$, but no generator can do better.

We prove Theorem 6.5 in two steps; first, we show that when the stated condition does not hold, then no algorithm can generate from $\mathcal{L}$ with set-based lower density; then, we show that if the condition holds, there exists an algorithm that generates from $\mathcal{L}$ with the desired lower density. The proof of Theorem 6.5 follows as direct corollary of these two results.

Lemma 6.7. *Let $\mathcal{L}$ be a countable collection of languages that contains L, L' with $|L \setminus L'| < \infty$ and $\mu_{\text{low}}(L, L') < c$. Then, no algorithm can generate from $\mathcal{L}$ in the limit with set-based lower density c under finite noise and finite omissions.*

Proof. We begin by formally recalling the requirements for an algorithm G to “generate from $\mathcal{L}$ in the limit with set-based lower density c ” under the specified adversaries. Let $x = (x_1, x_2, \dots)$ be an input sequence, and let $S_n = G(x_{1:n})$ be the algorithm’s hypothesis at timestep n . Recall that x is an **enumeration** of a target language $K \in \mathcal{L}$ with finite noise and omissions if:

1. **Finite Noise:** The set $\{x_i \mid x_i \notin K\}$ is finite.
2. **Finite Omissions:** The set $K \setminus \{x_i \mid i \in \mathbb{N}\}$ is finite.

The algorithm G must satisfy two properties for *every* $K \in \mathcal{L}$ and *every* (contaminated) enumeration x of K :

- (i) **Generation in the Limit:** There exists an n^* such that for all $n > n^*$, it holds that $S_n \subseteq K$.
- (ii) **Density Guarantee:** $\liminf_{n \rightarrow \infty} \mu_{\text{low}}(S_n, K) \geq c$.

We now proceed by contradiction.

Assume, for the sake of contradiction, that such an algorithm G exists. Let $L, L' \in \mathcal{L}$ be two languages satisfying the lemma’s hypotheses:

1. $F = L \setminus L'$ is a finite set.
2. $\mu_{\text{low}}(L, L') < c$.

We construct an input x and a sequence of time indices $n_0 < n_1 < n_2 < \dots$ inductively. Let $\mu_{\text{low}}(L, L') = c - \varepsilon$ for some $\varepsilon > 0$. This construction will force G to fail either property (i) or (ii) in these timesteps.

The construction proceeds in phases. We treat even-numbered and odd-numbered phases differently. For any phase $k \geq 1$:

Phase $2k - 1$ (Target L'):

1. Define a potential enumeration T_{2k-1} as follows:
 - ▷ It begins with the prefix $x_{1:n_{2k-2}}$ (which is empty for $k = 1$).
 - ▷ It is followed by a complete, ordered enumeration of all elements in $L' \setminus \{x_1, \dots, x_{n_{2k-2}}\}$.
2. T_{2k-1} is an **enumeration of L' with finite noise**.
 - ▷ **Noise:** The number of elements in T_{2k-1} not in L' is bounded by $|x_{1:n_{2k-2}}| < \infty$. Thus, this noise is finite.
 - ▷ **Omissions:** By construction, T_{2k-1} contains all elements of L' , so there are 0 omissions.
3. By the definition of successful generation in this setting, $\mathbb{G}$ running on T_{2k-1} must eventually be outputting subsets of L' that have c lower density in L' . Thus, there must exist a time $n_{2k-1} > n_{2k-2}$ such that $\mu_{\text{low}}(S_{n_{2k-1}}, L') \geq c - \varepsilon/2$, and $S_{n_{2k-1}} \subseteq L'$ (where $S_{n_{2k-1}}$ is the output on the prefix $T_{2k-1}[1 : n_{2k-1}]$). Thus, since $\mu_{\text{low}}(L, L') = c - \varepsilon$ it must be the case that $S_{n_{2k-1}} \not\subseteq L$.
4. If no such n_{2k-1} exists, $\mathbb{G}$ fails the generation requirement for L' on input T_{2k-1} . We halt the construction and have found our contradiction.
5. Otherwise, set $x_{1:n_{2k-1}} = T_{2k-1}[1 : n_{2k-1}]$ and proceed.

Round $2k$ (Target L):

1. Define a text T_{2k} as follows:
 - ▷ It begins with the prefix $x_{1:n_{2k-1}}$.
 - ▷ It is followed by a complete, ordered enumeration of all elements in $L \setminus \{x_1, \dots, x_{n_{2k-1}}\}$. (This includes all elements of F not already in the prefix).
2. T_{2k} is an **enumeration of L with finite noise**.
 - ▷ **Noise:** The set of elements in T_{2k} not in L is $x_{1:n_{2k-1}} \cap (L' \setminus L)$. This is a finite prefix of $L' \setminus L$, so the noise is finite.
 - ▷ **Omissions:** By construction, T_{2k} contains all elements of L so there are 0 omissions.
3. By the definition of successful generation in this setting, $\mathbb{G}$ running on T_{2k} must eventually be outputting subsets of L that have $c - \varepsilon/2$ lower density in L . Thus, there must exist a time $n_{2k} > n_{2k-1}$ such that $S_{n_{2k}} \subseteq L$.
4. If no such n_{2k} exists, $\mathbb{G}$ fails the generation requirement for L on input T_{2k} . We halt and have a contradiction.
5. Otherwise, set $x_{1:n_{2k}} = T_{2k}[1 : n_{2k}]$ and proceed to the next round.

This inductive process has two possible outcomes:

Case 1: The construction halts after a finite number of rounds. As shown in step (4) of the inductive rounds, if the construction halts, it is because $\mathbb{G}$ failed to find a required output in all the rounds of this phase. This constitutes a failure of the generation requirement for either L or L' on a complete enumeration of the corresponding language with finite noise and zero omissions.

Case 2: The construction proceeds for infinitely many rounds. This process defines a single, infinite input $x = \lim_{k \rightarrow \infty} x_{1:n_k}$ and an infinite sequence of time indices $n_1 < n_2 < n_3 < \dots$.

Let us show that x is a valid enumeration of L' with finite noise and zero omissions.

- ▷ **Noise:** The set of elements in x not in L' is $x \cap (L \setminus L') = x \cap F$. Elements from F are *only* added during the “Target L ” rounds (even k). Since F is finite, x contains at most $|F|$ noisy elements. The noise is finite.
- ▷ **Omissions:** The set $L' \setminus x$ is empty. In every “Target L' ” round (odd k), the text T_{2k-1} is defined to contain *all* elements of L' . Since x is the limit of these prefixes, it must contain all of L' .

Therefore, x is a **an enumeration of L' with finite noise**.

By property (ii) (Density Guarantee), A running on x must satisfy:

$$\liminf_{n \rightarrow \infty} \mu_{\text{low}}(S_n, L') \geq c.$$

However, consider the infinite subsequence of times $n_2, n_4, n_6, \dots, (n_{2k}, \dots)$. By construction, at every time n_{2k} (the end of a “Target L ” round), the algorithm’s output satisfies $S_{n_{2k}} \subseteq L$.

A key property of set-based density is that it is monotone in its first argument: if $A \subseteq B$, then $\mu_{\text{low}}(A, C) \leq \mu_{\text{low}}(B, C)$ for any C . Thus, for every $k \geq 1$:

$$S_{n_{2k}} \subseteq L \implies \mu_{\text{low}}(S_{n_{2k}}, L') \leq \mu_{\text{low}}(L, L').$$

By the lemma’s hypothesis, $\mu_{\text{low}}(L, L') < c$. This means we have an infinite subsequence of hypotheses $S_{n_{2k}}$ such that:

$$\mu_{\text{low}}(S_{n_{2k}}, L') \leq c - \varepsilon \quad \text{for all } k \geq 1.$$

This directly implies that the limit inferior of the *entire* sequence must be less than c :

$$\liminf_{n \rightarrow \infty} \mu_{\text{low}}(S_n, L') \leq \lim_{k \rightarrow \infty} \mu_{\text{low}}(S_{n_{2k}}, L') \leq \mu_{\text{low}}(L, L') < c.$$

This is a direct contradiction of the Density Guarantee (ii).

Thus, in both possible cases, the existence of algorithm $\mathbb{G}$ leads to a contradiction. In Case 1, $\mathbb{G}$ fails the Generation requirement. In Case 2, $\mathbb{G}$ fails the Density Guarantee. Therefore, no such algorithm $\mathbb{G}$ can exist. $\square$

Next, we proceed with describing an algorithm that achieves set-based lower density $c > 0$ whenever this condition holds. For our upper bound, we will utilize an algorithm that achieves the desired set-based lower density guarantee in the absence of any contamination in the dataset, and then we will utilize our expansion subroutine ([Algorithm 2](#)) to convert it to a generator that achieves this guarantee in the setting of finite contamination. First, we state a result which follows as an immediate corollary from Kleinberg and Mullainathan [[KM24](#)] and will be useful in our derivations.

Lemma 6.8 (Noiseless Set-Based Lower Density [[KM24](#)]). *Let $c \in [0, 1]$ and $\mathcal{L}$ be a countable collection for which every $L, L' \in \mathcal{L}$ with $|L \setminus L'| < \infty$ satisfies $\mu_{\text{low}}(L, L') \geq c$. Then, there exists a generating algorithm $\mathbb{G}$ that achieves set-based generation in the limit from $\mathcal{L}$ with set-based lower density at least c , under enumerations without noise or omissions.*

We remark that this is an immediate corollary of the algorithm of Kleinberg and Mullainathan [KM24]. The main property of their algorithm is that, in the limit, it outputs subsets of the target language. Thus, if every subset of the target language is c -dense in the target (as guaranteed by the premises of our characterization), the above result follows immediately. We are now ready to state and prove our result.

Lemma 6.9. *Let $\mathcal{L}$ be a countable collection of languages such that for all $L, L' \in \mathcal{L}$ either $|L \setminus L'| = \infty$ or $\mu_{\text{low}}(L, L') \geq c$. Then, there exists algorithm that generates from $\mathcal{L}$ in the limit with set-based lower density c under finite noise and finite omissions.*

Proof. We construct an algorithm $\mathbb{G}$ that generates from $\mathcal{L}$ in the limit and achieves the stated density guarantee, given the lemma's condition.

1. The Algorithm. First, we construct the expanded collection $\widetilde{\mathcal{L}}$ using the expansion subroutine from Algorithm 2. Recall that

$$\widetilde{\mathcal{L}} := \{L_{A,B} := L \cup A \setminus B : L \in \mathcal{L}, A \subseteq U \setminus L, B \subseteq L, |A| < \infty, |B| < \infty\}.$$

Then, we execute the algorithm described in Lemma 6.8 on the collection $\widetilde{\mathcal{L}}$ and the enumeration $x_{1:\infty}$. Since $x_{1:\infty}$ is a valid enumeration of an unknown target language $K \in \mathcal{L}$, with finite noise set A and finite omission set B , this corresponds to an enumeration without noise or omissions for the language $K_{A,B}$.

2. Proof of Generation in the Limit. Since the algorithm from Lemma 6.8 achieves (set-based) generation in the limit with respect to $\widetilde{\mathcal{L}}$, it follows as a direct corollary from Lemma 4.3 that it achieves (set-based) generation in the limit with respect to $\mathcal{L}$ when the enumeration contains finite amount of noise and omissions.

3. Proof of Density Guarantee. We need to show that $\liminf_{n \rightarrow \infty} \mu_{\text{low}}(A_n, K) \geq c$, where A_n is the output of the algorithm during the n -th step. First, notice that since every $L, L' \in \mathcal{L}$ with $|L \setminus L'| < \infty$ satisfy $\mu_{\text{low}}(L, L') \geq c$ it must also be the case that every $L_{A,B}, L'_{A',B'} \in \widetilde{\mathcal{L}}$ with $|L_{A,B} \setminus L'_{A',B'}| < \infty$ satisfy $\mu_{\text{low}}(L_{A,B}, L'_{A',B'}) \geq c$. This is simply because if $|L_{A,B} \setminus L'_{A',B'}| < \infty$, it must be the case that $|L \setminus L'| < \infty$, and lower density is invariant to adding or subtracting finitely many elements to its arguments. Thus, Lemma 6.8 guarantees that $\liminf_{n \rightarrow \infty} \mu_{\text{low}}(A_n, K_{A,B}) \geq c$. Since $K, K_{A,B}$ differ on finitely many elements, this directly implies that $\liminf_{n \rightarrow \infty} \mu_{\text{low}}(A_n, K) \geq c$. □

Remark 6.10 (Known Noise Level). It is worth highlighting that in our lower bound construction we made use of the crucial fact that the generator does not know the noise level; it merely knows that noise is finite. Interestingly, the previous characterization does not hold anymore when the generator knows a bound on the finite noise level. Recall that in the absence of the density requirement, knowledge of the noise rate does not change the set of countable collections that can be generated in the limit.

6.1.3 Characterization of Set-Based Lower & Upper Density under Vanishing Noise Rate and Arbitrary Omissions

In this section, we characterize when set-based density is achievable under vanishing noise rate.

Theorem 6.11 (Characterization of Set-Based Density with Vanishing Noise Rate). *Fix a collection $\mathcal{L}$ and $\rho \in (0, 1]$. Under vanishing noise rate and arbitrary omissions, there exists a set-based generator that generates in the limit and achieves set-based lower density ρ if and only if the following property holds.*

Condition 2 (Vanishing Noise ρ -Dense Set-Generation). *For every non-empty finite sub-collection $\mathcal{L}' \subseteq \mathcal{L}$, if languages in $\mathcal{L}'$ share infinitely many elements, i.e., $|\text{Cl}(\mathcal{L}')| = \infty$, then for any $L \in \mathcal{L}'$,*

$$\mu_{\text{low}}(\text{Cl}(\mathcal{L}'), L) \geq \rho.$$

Remark 6.12 (Upper Density). [Theorem 6.11](#)'s characterization is sharp in the following sense:

- ▷ Whenever [Condition 2](#) does not hold, no set-based generator can generate in the limit and achieve set-based *upper density* ρ , even when the adversary is restricted to use an enumeration with $o(1)$ -noise and *without omissions*.
- ▷ Whenever [Condition 2](#) holds, there exists a set-based generator that generates in the limit and achieves set-based *lower density* ρ , even when the adversary presents an enumeration with $o(1)$ -noise and *with infinite omissions*.

Example 6.13. As a simple application of the characterization in [Theorem 6.11](#), consider the following collection of languages $\mathcal{L} = \{L_1, L_2, L_3\}$, where $L_1 = \{i \in \mathbb{N} : i \not\equiv 0 \pmod{3}\}$, $L_2 = \{i \in \mathbb{N} : i \not\equiv 1 \pmod{3}\}$, $L_3 = \{i \in \mathbb{N} : i \not\equiv 2 \pmod{3}\}$. By the characterization, under vanishing noise rate and arbitrary omissions, the best set-based lower density any generator can achieve is $1/2$, since the intersection of any two languages have infinite cardinality and we can easily check the intersection has density $1/2$ in each of the two languages, whereas the intersection of all three languages is empty.

In the remainder of this section we prove [Theorem 6.11](#).

Necessity of [Condition 2](#)

We first prove the necessity of [Condition 2](#). As we will see, this condition is necessary even for enumerations without omission. Note that the case with enumerations can only be more challenging.

Proof of [Theorem 6.11](#) (Necessity). Assume that there exists a non-empty finite subcollection $\mathcal{L}' \subseteq \mathcal{L}$ such that $|\text{Cl}(\mathcal{L}')| = \infty$, and there exists $L' \in \mathcal{L}'$ such that $\mu_{\text{low}}(\text{Cl}(\mathcal{L}'), L') < \rho$.

We will construct an enumeration $x_1, x_2, x_3, \dots$ that is simultaneously an enumeration for any language $L \in \mathcal{L}'$ with $o(1)$ -noise and without omission. Given such an enumeration, any set-based generator that generates in the limit from $\mathcal{L}$ has to generate from all $L \in \mathcal{L}'$ in the limit, and thus will output a subset of $\text{Cl}(\mathcal{L}')$ for all large enough n . In particular, any subset of $\text{Cl}(\mathcal{L}')$

has lower set-density in L' strictly less than ρ . Since the adversary could have chosen $K = L'$, any set-based generator that generates in the limit cannot even achieve set-based upper density at least ρ , even if the adversary does not omit any elements of K in its enumeration.

Now, we will construct such an enumeration $x_1, x_2, x_3, \dots$ that is simultaneously an $o(1)$ -noisy enumeration for any language $L \in \mathcal{L}'$ without omission. Define a scheduling of time $T = \{1, 2, 4, 8, \dots\}$. We will only use that $\mu_{\text{up}}(T, \mathbb{N}) = 0$. For $n \notin T$, the adversary enumerates the next element in $\text{Cl}(\mathcal{L}')$ that has not been enumerated. For $n \in T$, the adversary enumerates the next element in $(\bigcup_{L \in \mathcal{L}'} L) \setminus \text{Cl}(\mathcal{L}')$ that has not been enumerated. Clearly, this enumeration will enumerate every elements of every $L \in \mathcal{L}'$. Clearly, for any $L \in \mathcal{L}'$, if the element x_n enumerated at time n does not belong to L , we must have $n \in T$. Since T has density 0 in $\mathbb{N}$, we conclude that $x_1, x_2, x_3, \dots$ is an enumeration for any language $L \in \mathcal{L}'$ with $o(1)$ -noise and without omission. This concludes the proof. $\square$

Sufficiency of [Condition 2](#)

We next show that if [Condition 2](#) holds, then there exists a set-based generator that generates in the limit and achieves set-based lower density ρ , under $o(1)$ -noise and potentially infinite omissions in its enumeration.

Pseudocode. The pseudocode is presented in [Algorithm 6](#).

Algorithm 6 Algorithm for [Theorem 6.11](#)

Input: Countable collection $\mathcal{L} = \{L_1, L_2, \dots\}$; thresholds $c_1, c_2, \dots \in (0, 1)$; enumeration $x_{1:\infty}$

- 1: Let $S_n \leftarrow \{x_1, \dots, x_n\}$ be the set of examples seen in the first n steps
- 2: **for** $i = 1, 2, \dots, n$ **do**
- 3: Compute the following number

$$N_i^{(n)} \leftarrow \min \{N \geq 1 : \forall m \in \{N, \dots, n\}, R(L_i; x_{1:m}) \leq c_i\}.$$

- 4: Assign the language L_i a priority of $P_i^{(n)} \leftarrow i + N_i^{(n)}$
- 5: **end for**
- 6: Re-order $\{L_1, \dots, L_n\}$ in increasing priority, tie-breaking by index, as $\{L_{i_n(1)}, \dots, L_{i_n(n)}\}$, i.e., for each $j \in [n-1]$, ensure either $P_{i_n(j)}^{(n)} < P_{i_n(j+1)}^{(n)}$, or $P_{i_n(j)}^{(n)} = P_{i_n(j+1)}^{(n)}$ and $i_n(j) < i_n(j+1)$.
- 7: Compute the largest index $j_n \in [n]$ such that the intersection of the prefix of the sorted list of languages in the re-ordering up to $L_{i_n(j_n)}$ is infinite, i.e.,

$$J_n \leftarrow \max \left\{ \bar{j} \in [n] : \left| \bigcap_{j=1}^{\bar{j}} L_{i_n(j)} \right| = \infty \right\}.$$

- 8: Output $\bigcap_{j \leq J_n} L_{i_n(j)} \setminus S_n$.
-

Proof of [Theorem 6.11](#) (Sufficiency). Assume that for any non-empty finite subcollection $\mathcal{L}' \subseteq \mathcal{L}$ with $|\text{Cl}(\mathcal{L}')| = \infty$, we have $\mu_{\text{low}}(\text{Cl}(\mathcal{L}'), L) \geq \rho$ for any $L \in \mathcal{L}'$.

Now consider the set-based generator described in [Algorithm 6](#), with threshold parameters $c_i := \varepsilon/2^i$ for all $i \in \mathbb{N}$.

Note that the priority of any language $P_i^{(n)}$ is non-decreasing. Also note that the priority of the target language K will remain fixed after some finite time n' , since the adversary uses an enumeration of $K = L_{i^*}$ with $o(1)$ -noise, and eventually the empirical noise rate $R(L_{i^*}; x_{1:n})$ stays below the positive threshold c_{i^*} for all large enough n . Let $p := P_{i^*}^\infty$ be the number that the priority of K stabilizes to.

By [Lemma 4.1](#), if we denote $\mathcal{L}(p) := \{L_i : P_i^\infty \leq p\}$ and $\mathcal{L}^{(n)}(p) := \{L_i : P_i^{(n)} \leq p\}$, there exists n^* such that for all $n \geq n^*$, we have $\mathcal{L}^{(n)}(p) = \mathcal{L}(p)$. Moreover, we know that $K \in \mathcal{L}(p)$ by definition.

Now we will show that $|\text{Cl}(\mathcal{L}(p))| = \infty$. Consider any $n \geq \max\{n^*, p\}$, we know that $\mathcal{L}^{(n)}(p) = \mathcal{L}(p)$. For any $L_i \in \mathcal{L}^{(n)}(p)$, we have $N_i^{(n)} = P_i^{(n)} - i < p$. Thus, for all $m \in \{p, p+1, \dots, n\}$, we have $R(L_i; x_{1:m}) \leq c_i$. In particular, $R(L_i; x_{1:n}) \leq c_i$. Therefore, we get that

$$\begin{aligned} |\text{Cl}(\mathcal{L}^{(n)}(p)) \cap S_n| &= |S_n| - \left| \bigcup_{L_i \in \mathcal{L}^{(n)}} (S_n \setminus L_i) \right| \\ &\geq |S_n| - \sum_{L_i \in \mathcal{L}^{(n)}} |S_n \setminus L_i| \\ &\geq n - \sum_{L_i \in \mathcal{L}^{(n)}} n \cdot c_i \\ &\geq n(1 - \varepsilon). \end{aligned}$$

Since the above inequality holds for any $n \geq \max\{n^*, p\}$ and $\mathcal{L}(p) = \mathcal{L}^{(n)}(p)$ for $n \geq \max\{n^*, p\}$, we conclude that $|\text{Cl}(\mathcal{L}(p))| = \infty$.

Therefore, at any round time $n \geq \max\{n^*, p\}$, we know that the finite subcollection $\mathcal{L}^{(n)}(p) = \mathcal{L}(p)$ appears as a prefix of the sorted list $\{L_{i_n(1)}, \dots, L_{i_n(n)}\}$. Since the intersection of this prefix, $\text{Cl}(\mathcal{L}(p))$, has infinite cardinality, we know that the computed index $j_n \in [n]$ satisfies that $\mathcal{L}(p) \subseteq \{L_{i_n(1)}, L_{i_n(2)}, \dots, L_{i_n(j_n)}\}$. Denote this subcollection as $\mathcal{L}_n := \{L_{i_n(1)}, L_{i_n(2)}, \dots, L_{i_n(j_n)}\}$. By definition of j_n , $|\text{Cl}(\mathcal{L}_n)| = \infty$.

Thus, for any $n \geq \max\{n^*, p\}$, we know that $\mathcal{L}(p) = \mathcal{L}^{(n)}(p) \subseteq \mathcal{L}_n$. Moreover, $\mathcal{L}_n$ is a finite subcollection of $\mathcal{L}$, and $K \in \mathcal{L}(p) \subseteq \mathcal{L}_n$. By the [Condition 2](#), for any $L \in \mathcal{L}_n$, we have $\mu_{\text{low}}(\text{Cl}(\mathcal{L}_n), \mathcal{L}_n) \geq \rho$. In particular, since $K \in \mathcal{L}_n$, we have $\mu_{\text{low}}(\text{Cl}(\mathcal{L}_n), K) \geq \rho$. Thus, the output

$$\bigcap_{j \leq j_n} L_{i_n(j)} \setminus S_n = \text{Cl}(\mathcal{L}_n) \setminus S_n$$

has lower density at least ρ in K , which concludes the proof. □

6.1.4 Characterization of Set-Based Lower & Upper Density under Constant Noise Rate and Arbitrary Omissions

In this section, we characterize when set-based density is achievable under constant noise rate and arbitrary omissions.

Theorem 6.14 (Characterization of Set-Based Density under Constant Noise Rate). *Fix a collection $\mathcal{L}$ and $c \in (0, 1]$. Under constant noise rate c and arbitrary omissions, there exists a set-based generator that generates in the limit and achieves set-based lower density ρ if and only if the following property holds.*

Condition 3 (c -constant Noise ρ -Dense Set-Generation). *For every non-empty finite subcollection $\mathcal{L}' \subseteq \mathcal{L}$ and every enumeration $x_{1:\infty}$, either*

- (a) *there exists some language $L' \in \mathcal{L}'$ such that $R(L'; x_{1:n}) > c$ infinitely often, or*
- (b) *each language $L \in \mathcal{L}'$ satisfies $\mu_{\text{low}}(\text{Cl}(\mathcal{L}'), L) \geq \rho$.*

In fact, we will show:

- ▷ Whenever [Condition 3](#) does not hold, no set-based generator can generate in the limit and achieve set-based *upper density* ρ when the adversary presents an enumeration with c -noise arbitrary omissions.
- ▷ Whenever [Condition 3](#) holds, there exists a set-based generator that generates in the limit and achieves set-based *lower density* ρ when the adversary presents an enumeration with c -noise and arbitrary omissions.

Necessity of [Condition 3](#)

We first prove the necessity of [Condition 3](#). The proof is similar to that of [Theorem 6.11](#) but simpler, since the negation of the c -constant noise rate ρ -dense set-generation property provides us with an enumeration that the adversary could use to force the density achieved by any set-based generator to be less than ρ .

Proof of [Theorem 6.14](#) (Necessity). Assume that there exists a non-empty finite subcollection $\mathcal{L}' \subseteq \mathcal{L}$ and an enumeration $x_1, x_2, x_3, \dots$ such that

- (a) for every $L \in \mathcal{L}'$, the $x_1, x_2, x_3, \dots$ is a c -noisy enumeration for L ,
- (b) and there exists $L' \in \mathcal{L}'$ such that $\mu_{\text{low}}(\text{Cl}(\mathcal{L}'), L) < \rho$.

Since $x_1, x_2, x_3, \dots$ is an enumeration with c -noise for any language in the finite subcollection $\mathcal{L}'$, any set-based generator that generates in the limit from $\mathcal{L}$ has to generate from all $L \in \mathcal{L}'$ in the limit under this enumeration. Thus, such a set-based generator will output a subset of $\text{Cl}(\mathcal{L}')$ for all large enough n . In particular, any subset of $\text{Cl}(\mathcal{L}')$ has lower set-density in L' strictly less than ρ . Since the adversary could have chosen $K = L'$, any set-based generator that generates in the limit cannot even achieve set-based upper density at least ρ . $\square$

Sufficiency of [Condition 3](#)

We next show that by slightly adapting [Algorithm 6](#) with threshold parameters $c_i := c$ and a different stopping condition, we obtain a set-based generator that achieves set-based lower density whenever [Condition 3](#) holds.

Pseudocode. The pseudocode is presented in [Algorithm 7](#).

Algorithm 7 Algorithm for [Theorem 6.14](#)

Input: Countable collection $\mathcal{L} = \{L_1, L_2, \dots\}$; $c \in (0, 1]$; enumeration $x_{1:\infty}$

- 1: Let $S_n \leftarrow \{x_1, \dots, x_n\}$ be the set of examples seen in the first n steps
- 2: **for** $i = 1, 2, \dots, n$ **do**
- 3: Compute the following number

$$N_i^{(n)} \leftarrow \min \{N \geq 1 : \forall m \in \{N, \dots, n\}, R(L_i; x_{1:m}) \leq c\}.$$

- 4: Assign the language L_i a priority of $P_i^{(n)} \leftarrow i + N_i^{(n)}$
- 5: **end for**
- 6: Re-order $\{L_1, \dots, L_n\}$ in increasing priority, tie-breaking by index, as $\{L_{i_n(1)}, \dots, L_{i_n(n)}\}$, i.e., for each $j \in [n-1]$, ensure either $P_{i_n(j)}^{(n)} < P_{i_n(j+1)}^{(n)}$, or $P_{i_n(j)}^{(n)} = P_{i_n(j+1)}^{(n)}$ and $i_n(j) < i_n(j+1)$.
- 7: Compute the largest index $j_n \in [n]$ such that the intersection of the prefix of the sorted list of languages in the re-ordering up to $L_{i_n(j_n)}$ has lower set-density at least ρ for language $L_{i_n(j)}$ for any $j \leq j_n$, i.e.,

$$J_n \leftarrow \max \left\{ \bar{j} \in [n] : \forall j' \leq \bar{j}, \mu_{\text{low}} \left(\bigcap_{j=1}^{\bar{j}} L_{i_n(j)}, L_{i_n(j')} \right) \geq \rho \right\}.$$

- 8: Output $\bigcap_{j \leq J_n} L_{i_n(j)} \setminus S_n$.
-

Proof of [Theorem 6.14](#) (Sufficiency). Assume that for any non-empty finite subcollection $\mathcal{L}' \subseteq \mathcal{L}$ and every enumeration $x_1, x_2, x_3, \dots$, either

- (a) there is some $L' \in \mathcal{L}'$ such that $R(L'; x_{1:n}) > c$ infinitely often, or
- (b) each language $L \in \mathcal{L}'$ satisfies $\mu_{\text{low}}(\text{Cl}(\mathcal{L}'), L) \geq \rho$.

Now consider the set-based generator described in [Algorithm 6](#), with threshold parameters $c_i := c$ for all $i \in \mathbb{N}$.

Note that the priorities of any language $P_i^{(n)}$ is non-decreasing. Also note that the priority of the target language K will remain fixed after some finite time n' , since the adversary uses an enumeration with c -noise of $K = L_{i^*}$, and eventually the empirical noise rate $R(L_{i^*}; x_{1:n}) \leq c$ for all large enough n . Let $p := P_{i^*}^\infty$ be the number that the priority of K stabilizes to.

By [Lemma 4.1](#), if we denote $\mathcal{L}(p) := \{L_i : P_i^\infty \leq p\}$ and $\mathcal{L}^{(n)}(p) := \{L_i : P_i^{(n)} \leq p\}$, there exists n^* such that for all $n \geq n^*$, we have $\mathcal{L}^{(n)}(p) = \mathcal{L}(p)$, and $P_i^{(n)} = P_i^\infty$ for all $L_i \in \mathcal{L}(p)$, i.e., the priorities of all languages in $\mathcal{L}(p)$ stabilize. Moreover, we know that $K \in \mathcal{L}(p)$ by definition.

Therefore, at any round time $n \geq \max\{n^*, p\}$, we know that the finite subcollection $\mathcal{L}^{(n)}(p) = \mathcal{L}(p)$ appears as a prefix of the sorted list $\{L_{i_n(1)}, \dots, L_{i_n(n)}\}$. Moreover, since the priorities of all languages in $\mathcal{L}(p)$ stabilize, $N_i^{(n)} = P_i^{(n)} - i = P_i^\infty - i$ for any $L_i \in \mathcal{L}(p)$. In other words, by definition of $N_i^{(n)}$, the enumeration $x_1, x_2, x_3, \dots$ that the adversary uses is a c -noisy enumeration for any $L_i \in \mathcal{L}(p)$. Thus, each language $L \in \mathcal{L}(p)$ satisfies $\mu_{\text{low}}(\text{Cl}(\mathcal{L}(p)), L) \geq \rho$.

Since the intersection of this prefix, $\text{Cl}(\mathcal{L}(p))$, satisfies that $\mu_{\text{low}}(\text{Cl}(\mathcal{L}(p)), L) \geq \rho$ for any $L \in \mathcal{L}(p)$, we know that the computed index $j_n \in [n]$ satisfies that $\mathcal{L}(p) \subseteq \{L_{i_n(1)}, L_{i_n(2)}, \dots, L_{i_n(j_n)}\}$. Denote this subcollection as $\mathcal{L}_n := \{L_{i_n(1)}, L_{i_n(2)}, \dots, L_{i_n(j_n)}\}$. By definition of j_n , every $j' \leq j_n$

satisfies $\mu_{\text{low}}(\text{Cl}(\mathcal{L}_n), L_{i_n(j_n)}) \geq \rho$. In particular, since $K \in \mathcal{L}(p) \subseteq \mathcal{L}_n$, we have $\mu_{\text{low}}(\text{Cl}(\mathcal{L}_n), K) \geq \rho$. Thus, the output

$$\bigcap_{j \leq j_n} L_{i_n(j)} \setminus S_n = \text{Cl}(\mathcal{L}_n) \setminus S_n$$

has lower density at least ρ in K , which concludes the proof. $\square$

6.2 Generation with Element-Based Density under Contamination

Next we consider the notion of generation with element-based density introduced by [KW25]. Recall that under this definition (Definition 9), the generator outputs one element in every timestep and we measure the density of the *entire output sequence* of the generator in the target language K . The validity requirement remains the same as in the original framing of the setting [KM24], i.e., after some finite timestep the generator needs to output unseen elements of the target language K . In Section 6.2.1 we show a transformation from algorithms that achieve set-based lower density to algorithms that achieve element-based lower density. In Section 6.2.2 we discuss our results in the case of *finite* amount of noise and omissions from the adversary, and in Section 6.2.3 we discuss a general impossibility result for achieving element-based density under infinite noise.

6.2.1 From Set-Based Lower Density to Element-Based Lower Density

To begin with, we show that under some mild assumptions, which all the set-based generation algorithms in this paper satisfy, we may turn any generator that achieves set-based density into one that achieves element-based density.

Theorem 6.15. *Let $\mathcal{L}$ be a countable collection. Suppose under a specific enumeration $x_{1:\infty}$ of the target language K , potentially under contamination, there is a set-based generator $\mathbb{G}$ that generates in the limit from $\mathcal{L}$. Suppose at every round n , given $S_n = \{x_1, \dots, x_n\}$, $\mathbb{G}$ outputs an infinite set $A_n \subseteq U$ and computes a finite subcollection $\mathcal{L}_n \subseteq \mathcal{L}$ such that (a) $K \in \mathcal{L}_n$ for all large enough n and (b) $A_n \subseteq \text{Cl}(\mathcal{L}_n) \setminus \{x_1, \dots, x_n\}$ for all n .*

Assume further that $\mathbb{G}$ achieves $\mu_{\text{low}}(A_n, K) \geq \rho$, for all n sufficiently large. Then, there exists an element-based generator $\overline{\mathbb{G}}$ that, under the same enumeration $x_{1:\infty}$, generates in the limit from $\mathcal{L}$ and achieves element-based lower density at least $\rho/2$.

If we assume some set-based generator achieves set-based lower density at least ρ , then for any $\varepsilon > 0$, we have $\mu_{\text{low}}(A_n, K) \geq \rho - \varepsilon$ for all large enough n . Thus, we have the following simple corollary.

Corollary 6.16. *In the same setting as Theorem 6.15, if the set-based generator achieves set-based lower density at least ρ , i.e., $\liminf_{n \rightarrow \infty} \mu_{\text{low}}(A_n, K) \geq \rho$. Then, for any $\varepsilon > 0$, there exists an element-based generator that generates in the limit and achieves element-based lower density at least $(\rho - \varepsilon)/2$.*

We remark that the element-based generator we construct for Theorem 6.15 assumes the knowledge of ρ .

Pseudocode. The pseudocode is presented in Algorithm 8. We assume that at each round, the set-based generator computes a finite set of indices $I(n)$ such that $\mathcal{L}_n = \{L_i : i \in I(n)\}$, and

outputs an infinite set $A_n \subseteq \text{Cl}(\mathcal{L}_n) \setminus \{x_1, \dots, x_n\}$. Moreover, for all sufficiently large n , $K \in \mathcal{L}_n$ and $\mu_{\text{low}}(A_n, K) \geq \rho$ for some given $\rho \in [0, 1]$.

Algorithm 8 Algorithm for [Theorem 6.15](#)

Input: Countable collection $\mathcal{L}$; enumeration $x_{1:\infty}$; set-based generator $\mathbb{G}$

- 1: Let $S_n \leftarrow \{x_1, \dots, x_n\}$ be the set of examples enumerated by the adversary till round n
- 2: Let $W_{n-1} \leftarrow \{w_1, \dots, w_{n-1}\}$ be the set of elements output before round n
- 3: Compute $(A_n, \mathcal{L}_n) \leftarrow \mathbb{G}(S_n)$
- 4: Compute $m_n \in \mathbb{N}$ to be the smallest number such that for all $L \in \mathcal{L}_n$,

$$\frac{|A_n \cap \{\ell_1, \ell_2, \dots, \ell_m\}|}{m} \geq \frac{\mu_{\text{low}}(A_n, L)}{1 + 2^{-n}}, \quad \forall m \geq m_n, \quad (1)$$

where $L = \{\ell_1, \ell_2, \ell_3, \dots\}$ denote the listing of elements of L in the natural ordering.

- 5: **if** $n > 1$ **then**
- 6: Update $m_n \leftarrow \max\{m_n, m_{n-1} + 1\}$ so that the sequence $m_1 < \dots < m_n$ is increasing.
- 7: **end if**
- 8: Compute the index

$$k(n) \leftarrow \begin{cases} \max\left\{k \in [n] : \frac{m_k}{1+2^{-k}} \leq \frac{2n}{\rho}\right\}, & \text{if } \frac{m_1}{1+2^{-1}} \leq \frac{2n}{\rho}, \\ 1, & \text{else.} \end{cases}$$

- 9: Output the smallest $w_n \in A_{k(n)} \setminus (S_n \cup W_{n-1})$ in the canonical ordering.
-

Proof of Theorem 6.15. If $\rho = 0$, then the statement of the theorem becomes trivial. We assume that $\rho > 0$ is a positive constant.

Fix an enumeration $x_{1:\infty}$, potentially under contamination. Let $\mathbb{G}$ be a set-based generator satisfying the assumptions in [Theorem 6.15](#). Consider the element-based generator $\overline{\mathbb{G}}$ described in [Algorithm 8](#).

We first show that the element-based generator $\overline{\mathbb{G}}$ in [Algorithm 8](#) is well-defined. Since $\mathcal{L}_n$ is a finite subcollection of $\mathcal{L}$, if we define $m_n(L) \in \mathbb{N}$ to be the smallest number such that (1) holds for L , then

$$m_n = \max_{L \in \mathcal{L}_n} m_n(L)$$

is well-defined. Each A_n is infinite, hence there is always some $w_n \in A_{k(n)} \setminus (S_n \cup W_{n-1})$.

Now we proceed to show that $\overline{\mathbb{G}}$ generates in the limit and achieves element-based lower density at least $\frac{\rho}{2}$.

$\overline{\mathbb{G}}$ generates in the limit: Let N_1 be sufficiently large so that $\mathcal{L}_n \ni K$ for all $n \geq N_1$. Moreover, let N_2 be sufficiently large so that $N_2 \geq N_1$ and $\frac{2n}{\rho} \geq \frac{m_{N_1}}{1+2^{-N_1}}$ for all $n \geq N_2$. Since the sequence $\frac{m_n}{1+2^{-n}}$ is non-decreasing in n , we see that $k(n) \geq N_1$ for all $n \geq N_2$. This means that $K \in \mathcal{L}_{k(n)}$ and $A_{k(n)} \subseteq K$ for all $n \geq N_2$, so that $w_n \in A_{k(n)} \setminus (S_n \cup W_{n-1}) \subseteq K \setminus (S_n \cup W_{n-1})$ for all $n \geq N_2$.

$\overline{\mathbb{G}}$ achieves element-based lower density: Let $K = \{\ell_1, \ell_2, \dots\}$ denote the canonical ordering of K . We first prove the following claim:

Claim 6.17. *There exists $n^* \in \mathbb{N}$ such that $w_n \in \{\ell_1, \dots, \ell_{\lceil 2n(1+2^{-k(n)})/\rho} \}$ for all $n \geq n^*$.*

To see the claim, let N_3 be sufficiently large so that $\mathcal{L}_n \ni K$ and $\mu(A_n, K) \geq \rho$ for all $n \geq N_3$. Moreover, let N_4 be sufficiently large so that $\frac{2n}{\rho} \geq \frac{m_{N_3}}{1+2^{-N_3}}$ for all $n \geq N_4$. This means that $A_{k(n)} \subseteq K$ for all $n \geq N_4$. By construction, for all $n \geq N_4$, we have $k(n) \geq N_3$ and

$$|A_{k(n)} \cap \{\ell_1, \dots, \ell_m\}| \geq \frac{\mu_{low}(A_{k(n)}, K) \cdot m}{1 + 2^{-k(n)}} \geq \frac{\rho m}{1 + 2^{-k(n)}}$$

for all $m \geq m_{k(n)}$, and in particular for $m = \lceil 2n(1 + 2^{-k(n)})/\rho \rceil$ by construction of $k(n)$.

In other words,

$$\left| A_{k(n)} \cap \left\{ \ell_1, \dots, \ell_{\lceil 2n(1+2^{-k(n)})/\rho} \right\} \right| \geq \frac{\rho \left(2n(1 + 2^{-k(n)})/\rho \right)}{1 + 2^{-k(n)}} = 2n.$$

Since there are at most $(2n - 1)$ elements in $S_n \cup W_{n-1}$, the smallest unseen element w_n according to the canonical ordering of $A_{k(n)}$ that we output must belong to the first $\lceil 2n(1 + 2^{-k(n)})/\rho \rceil$ elements in the canonical ordering of K . This finishes the proof of the claim, in which we can take $n^* = N_4$.

With [Claim 6.17](#) in hand, for $n \geq n^*$, if we define

$$t(n) := \max_{\bar{n} \leq n} \left\lceil 2\bar{n}(1 + 2^{-k(\bar{n})})/\rho \right\rceil$$

then,

$$\frac{|\{w_{1:\infty} \cap \{\ell_1, \dots, \ell_{t(n)}\}\}|}{t(n)} \geq \frac{|\{w_{n^*}, w_{n^*+1}, \dots, w_n\} \cap \{\ell_1, \dots, \ell_{t(n)}\}|}{t(n)} \geq \frac{n - n^*}{t(n)}. \quad (2)$$

Since $t(n) = \max_{\bar{n} \leq n} \lceil 2\bar{n}(1 + 2^{-k(\bar{n})})/\rho \rceil \rightarrow \infty$ as $n \rightarrow \infty$ and is non-decreasing, for all m large enough, there exists a unique number $n(m) \in \mathbb{N}$ such that

$$t(n(m)) \leq m < t(n(m) + 1).$$

Moreover, $n(m)$ is non-decreasing in m , and $n(m) \rightarrow \infty$ as $m \rightarrow \infty$. Also note that $k(n)$ is an unbounded and non-decreasing integer sequence. Since $k(n) \rightarrow \infty$ and $t(n) \rightarrow \infty$ as $n \rightarrow \infty$, we

have

$$\begin{aligned}
\mu_{\text{low}}(w_{1:\infty}, K) &= \liminf_{m \rightarrow \infty} \frac{|w_{1:\infty} \cap \{\ell_1, \dots, \ell_m\}|}{m} \\
&\geq \liminf_{m \rightarrow \infty} \frac{|w_{1:\infty} \cap \{\ell_1, \dots, \ell_{t(n(m))}\}|}{t(n(m) + 1)} \\
&= \liminf_{m \rightarrow \infty} \frac{|w_{1:\infty} \cap \{\ell_1, \dots, \ell_{t(n(m))}\}|}{t(n(m))} \cdot \frac{t(n(m))}{t(n(m) + 1)} \\
&= \liminf_{m \rightarrow \infty} \frac{|w_{1:\infty} \cap \{\ell_1, \dots, \ell_{t(n(m))}\}|}{t(n(m))} \cdot \frac{\max_{\bar{n} \leq n(m)} \lceil 2\bar{n}(1 + 2^{-k(\bar{n})})/\rho \rceil}{\max_{\bar{n} \leq n(m)+1} \lceil 2\bar{n}(1 + 2^{-k(\bar{n})})/\rho \rceil} \\
&= \liminf_{m \rightarrow \infty} \frac{|w_{1:\infty} \cap \{\ell_1, \dots, \ell_{t(n(m))}\}|}{t(n(m))} \cdot \frac{t(n(m))}{\max\{t(n(m)), \lceil 2(n(m) + 1)(1 + 2^{-k(n(m)+1)})/\rho \rceil\}}
\end{aligned}$$

since $t(n(m)) = \max_{\bar{n} \leq n(m)} \lceil 2\bar{n}(1 + 2^{-k(\bar{n})})/\rho \rceil \geq \lceil 2n(m)(1 + 2^{-k(n(m))})/\rho \rceil$, we have

$$\begin{aligned}
&\geq \liminf_{m \rightarrow \infty} \frac{|w_{1:\infty} \cap \{\ell_1, \dots, \ell_{t(n(m))}\}|}{t(n(m))} \\
&\quad \cdot \frac{\lceil 2n(m)(1 + 2^{-k(n(m))})/\rho \rceil}{\max\{\lceil 2n(m)(1 + 2^{-k(n(m))})/\rho \rceil, \lceil 2(n(m) + 1)(1 + 2^{-k(n(m)+1)})/\rho \rceil\}} \\
&= \liminf_{m \rightarrow \infty} \frac{|w_{1:\infty} \cap \{\ell_1, \dots, \ell_{t(n(m))}\}|}{t(n(m))} \cdot \min \left\{ 1, \frac{\lceil 2n(m)(1 + 2^{-k(n(m))})/\rho \rceil}{\lceil 2(n(m) + 1)(1 + 2^{-k(n(m)+1)})/\rho \rceil} \right\}
\end{aligned}$$

since $n(m) \rightarrow \infty$ and $k(n(m)) \rightarrow \infty$ as $m \rightarrow \infty$, we get

$$= \liminf_{m \rightarrow \infty} \frac{|w_{1:\infty} \cap \{\ell_1, \dots, \ell_{t(n(m))}\}|}{t(n(m))}$$

by inequality (2), we get

$$\geq \liminf_{m \rightarrow \infty} \frac{n(m) - n^*}{t(n(m))}$$

since $n(m)$ is non-decreasing and unbounded, we have

$$\begin{aligned}
&\geq \liminf_{n \rightarrow \infty} \frac{n - n^*}{t(n)} \\
&= \liminf_{n \rightarrow \infty} \frac{n - n^*}{\max_{\bar{n} \leq n} \lceil 2\bar{n}(1 + 2^{-k(\bar{n})})/\rho \rceil}
\end{aligned}$$

note that if $\bar{n} < n/2$, we have $2\bar{n}(1 + 2^{-k(\bar{n})})/\rho \leq 2n/\rho \leq 2n(1 + 2^{-k(n)})/\rho$. Thus, we have

$$\begin{aligned}
&= \liminf_{n \rightarrow \infty} \frac{n - n^*}{\max_{\substack{\bar{n}: \\ n/2 \leq \bar{n} \leq n}} \lceil 2\bar{n}(1 + 2^{-k(\bar{n})})/\rho \rceil} \\
&\geq \liminf_{n \rightarrow \infty} \frac{n - n^*}{\max_{\substack{\bar{n}: \\ n/2 \leq \bar{n} \leq n}} \lceil 2n(1 + 2^{-k(\bar{n})})/\rho \rceil} \\
&= \frac{\rho}{2},
\end{aligned}$$

where the last equality follows from $k(n) \rightarrow \infty$ as $n \rightarrow \infty$.

This concludes the proof. $\square$

6.2.2 Element-Based Density with Finite Contamination

Our main result in this setting is reduction to the noiseless case: given an algorithm that achieves upper (lower) element-based density for *all* countable collections in the noiseless setting, we obtain an algorithm that achieves upper (lower) element-based density for all countable collections when the adversary is allowed to omit finitely many elements from K and inject finitely many elements outside of K in its enumeration. This reduction is based on the expansion subroutine described in [Algorithm 2](#).

Theorem 6.18 (Element-Based Density Guarantee with Finite Contamination). *Fix $\rho \in [0, 1]$. Suppose there is a generator G_{vanilla} , that, for any countable collection $\mathcal{L}$, generates $\mathcal{L}$ in the limit with upper (respectively lower) element-based density ρ_{up} (respectively ρ_{up}) under no noise and no omissions.*

Then there exists a generator G_{tolerant} , that, for any countable collection $\mathcal{L}$, generates $\mathcal{L}$ in the limit with upper (respectively lower) element-based density ρ_{up} (respectively ρ_{up}).

Before we give the proof of the result, we recall two results from [\[KW25\]](#) that are important to our derivations.

Theorem 6.19 (Noiseless Element-Based Density Guarantees [\[KW25\]](#)). *For every countable collection of languages there is an algorithm that generates in the limit and achieves $1/2$ (respectively $1/8$) upper (respectively lower) element-based density when there is no noise or omissions.*

This, combined with [Theorem 6.18](#), gives us the following corollary.

Corollary 6.20. *For all countable collections of languages $\mathcal{L}$, there is an algorithm that achieves $1/2$ (respectively $1/8$) upper (respectively lower) element-based density under finite noise and finite omissions.*

We are now ready to prove our result.

Proof of Theorem 6.18. The proof proceeds by reduction. We construct a new algorithm, G_{tolerant} , that uses the algorithm G_{vanilla} from [Theorem 6.19](#) as a subroutine. We will show that G_{tolerant} generates from $\mathcal{L}$ under the noisy setting and inherits the density guarantees of G_{vanilla} .

1. The Reduction. Let $\mathcal{L}$ be the countable collection of languages from the theorem statement. We define a new, expanded collection of languages, $\mathcal{L}'$ (also described in [Algorithm 2](#)), as follows:

$$\mathcal{L}' := \{L \cup A \setminus B \mid L \in \mathcal{L}, A, B \subseteq \mathcal{X}, |A| < \infty, |B| < \infty\}$$

In words, $\mathcal{L}'$ contains every language from $\mathcal{L}$ plus all possible “contaminated” versions of those languages that result from adding a finite set (A) and removing a finite set (B). Since $\mathcal{L}$ is countable and the set of all pairs of finite sets (A, B) is countable, $\mathcal{L}'$ is also a countable collection.

Our Algorithm (G_{tolerant}): Our new algorithm G_{tolerant} is defined simply as the algorithm G_{vanilla} from [Theorem 6.19](#) executed on the language class $\mathcal{L}'$.

The Reduction: The key insight is that an adversary presenting a *contaminated enumeration* of $K \in \mathcal{L}$ is indistinguishable from an adversary presenting an *enumeration of some $T \in \mathcal{L}'$ without noise or omissions*. Specifically, let the adversary choose a target language $K \in \mathcal{L}$ and present an enumeration x with a finite noise set $A_x = x \setminus K$ and a finite omission set $B_x = K \setminus x$. The set of all elements enumerated in the text x is therefore precisely the language $T = (K \setminus B_x) \cup A_x$.

By definition, this language T is a member of our augmented class $\mathcal{L}'$ (with $L = K, A = A_x, B = B_x$). Therefore, the contaminated enumeration of K is, from the algorithm’s perspective, a *complete and noiseless* enumeration of the language $T \in \mathcal{L}'$.

2. Proof of Correctness (Generation in the Limit). From [Lemma 4.3](#) we immediately get that our algorithm generates in the limit from K .

3. Proof of Density Guarantees. We again consider the adversary’s perspective (noisy enumeration of K) and the algorithm’s perspective (noiseless enumeration of T). Let $S_{\text{out}} = (s_1, s_2, \dots)$ be the infinite output sequence from G_{tolerant} . By [Theorem 6.19](#), G_{vanilla} achieves the density guarantees for its output with respect to its target T . Formally, per the specified definition, the guarantees on the output set S_{out} are

$$\begin{aligned} \triangleright \mu_{\text{up}}(S_{\text{out}}, T) &= \limsup_{n \rightarrow \infty} \frac{|S_{\text{out}} \cap \{t_1, \dots, t_n\}|}{n} \geq \rho_{\text{up}}, \\ \triangleright \mu_{\text{low}}(S_{\text{out}}, T) &= \liminf_{n \rightarrow \infty} \frac{|S_{\text{out}} \cap \{t_1, \dots, t_n\}|}{n} \geq \rho_{\text{low}}, \end{aligned}$$

where $\{t_1, \dots, t_n\}$ is the set of the first n elements of T (based on the canonical enumeration of the domain). It is not hard to see that since the symmetric difference of K, T is finite, the exact same bounds hold when T is replaced by K . $\square$

6.2.3 Impossibility of Element-Based Density Under Infinite Contamination

Next, we move to the study of generation with density under infinite noise. First, by [Theorem 6.5](#), we know that even under finite noise, set-based generation is not guaranteed to achieve any non-trivial density for arbitrary countable collection. Clearly, this already rules out the possibility of achieving non-trivial set-based density for arbitrary countable collection under infinite contamination. On the other hand, [Theorem 6.18](#) shows that element-based generation always achieves a non-trivial upper (lower) element-based density under enumeration with finite contamination. It is natural to wonder if it is still possible to achieve non-trivial element-based density under infinite contamination, or even vanishing noise rate without omissions. The following theorem answers this question in the negative.

Theorem 6.21. *There exists a collection of two languages L_1, L_2 and an enumeration with $o(1)$ -noise and without omissions from a target language K in this collection, such that no element-based generator can generate from K in the limit and achieve element-based upper density at least ρ for any $\rho > 0$.*

Proof. Consider two languages L_1, L_2 with the property that $L_1 \subseteq L_2$ and $\mu_{\text{up}}(L_1, L_2) = 0$. Consider the following enumeration of the elements of L_2 .

Define a scheduling of time steps $T = \{1, 2, 4, 8, \dots\}$. We only need the fact that T has upper density 0 in $\mathbb{N}$. At time $n \in T$, the adversary outputs the next element in $L_2 \setminus L_1$ that hasn't been generated. At time $n \notin T$, the adversary outputs the next element in L_1 that hasn't been generated.

Note that the adversary will enumerate all the elements of L_2 , and thus all the elements of L_1 . Clearly, the adversary enumerates from L_2 with noise rate 0. On the other hand, since the adversary enumerates elements not belonging to L_1 only at time $n \in T$, and T has density 0 in $\mathbb{N}$, the noise rate of the adversary's enumeration is $o(1)$ in L_1 .

Since the enumeration is a valid enumeration with $o(1)$ -noise for both L_1 and L_2 , and $L_1 \subseteq L_2$, any generator generating from K in the limit has to generate from L_1 in the limit. Since L_1 has upper density 0 in L_2 , the generator achieves element-based upper density 0 if the adversary sets $K = L_2$. $\square$

7 Generation with Density Beyond the Worst-Case

In this section we introduce a beyond-worst-case model that restricts the order in which the adversary can present the elements of its chosen language K ; importantly, this model does not restrict the choices of the adversary in selecting the target K , it merely restricts its power to present its elements in arbitrary orders. Informally, the adversary cannot, infinitely often, enumerate elements of K that appear arbitrarily later in the canonical enumeration than the elements it has enumerated so far. In particular, under the interpretation that the canonical enumeration places “easier” elements of languages before “harder” ones, we impose the restriction that the adversary cannot, infinitely often, enumerate very hard examples before easier ones. This is inspired by practical phenomena of LLM training; indeed, folklore results known as “curriculum learning” empirically show that it is crucial that LLMs are first trained on “easier” tasks before harder ones [BLCW09; HW19].

Definition 11 (*M-Bounded Displacement Enumeration*). *Let $x_{1:\infty}$ be any enumeration and L be an arbitrary language. Let $\{\ell_1, \ell_2, \dots\} = L$ be the canonical enumeration of L and define*

$$\sigma(n) = \sigma(x_n, L) := \begin{cases} j, & x_n = \ell_j \in L, \\ 0, & x_n \notin L. \end{cases}$$

We say that $x_{1:\infty}$ is an M -bounded displacement enumeration with respect to L if there is some $n^ \in \mathbb{N}$ such that for all $n \geq n^*$, $\sigma(n) \leq Mn$.*

Remark that by definition, $\ell_{\sigma(n)} = \ell_{\sigma(x_n, L)} = x_n$ for all $x_n \in L$. Note also that we do not require $\sigma : \mathbb{N} \rightarrow \mathbb{N} \cup \{0\}$ to be injective or surjective since it is possible that $x_{1:\infty} \neq L$ as sets.

In order to develop some intuition for [Definition 11](#), we consider several examples below. First, we consider some simple noiseless examples.

Example 7.1. Let $L = \mathbb{N}$ and consider the following enumerations.

- ▷ The canonical enumeration $x_n = n$ is M -bounded for $M = 1$
- ▷ Let $L := \mathbb{N} - M \cdot \mathbb{N}$ be the natural numbers that are not divisible by M . The enumeration where $x_n = M \cdot \lceil n/2 \rceil$ when n is odd and x_n being the $(n/2)$ -th element of L when n is even is $(M/2)$ -bounded.
- ▷ Let $L := \{n \in \mathbb{N} : \sqrt{n} \notin \mathbb{N}\}$ be the non-squares. The enumeration consisting of $x_n = \lceil n/2 \rceil^2$ when n is odd and x_n being the $(n/2)$ -th element of L when n is even is *not* M -bounded for any M .

Next, we consider a slightly more involved example with constant noise rate. As we will see in [Section 7.1](#), [Example 7.2](#) demonstrates that generation under constant noise remains hard even with bounded displacement enumerations.

Example 7.2. Let $k \geq 2$ be an integer and define the language

$$L_i := [k] \cup (\mathbb{N} \setminus i \cdot \mathbb{N}) = \{1, \dots, k\} \cup \{n \in \mathbb{N} : n \text{ is not divisible by } i\}.$$

Consider the finite collection $\mathcal{L} := \{L_i : i \in [k]\}$ and the canonical enumeration $x_n = n$ of the natural numbers. By construction, the noise rate w.r.t. each $L_i, i \in [k]$ is bounded above by $1/k$. Moreover, we have $\sigma(x_n, L_i) \leq (1 + 1/k)n$ for all $n \geq 1, i \in [k]$. Hence $x_{1:\infty}$ is a M -bounded displacement enumeration of every $L_i, i \in [k]$ with noise rate $1/k$.

Remark 7.3. Intuitively, [Definition 11](#) restricts the “speed” of the enumeration to a linear speed-up compared to the canonical enumeration. Slightly more precisely, the adversary cannot place elements from deep in the tail of the canonical enumeration at the front of the input enumeration. Moreover, the adversary cannot omit too many elements from the prefix of the canonical enumeration, as otherwise, it is forced to output elements from deeper in the tail of the canonical enumeration.

It is illustrative to consider the lower bound construction for generation with density under $o(1)$ -noise enumerations from [Theorem 6.21](#) which consists of two languages $L' \subseteq L$. Let $L = \{\ell_1, \ell_2, \dots\}$ be the canonical enumeration of L . [Theorem 6.21](#) essentially chooses $L' = \{\ell_{i_1}, \ell_{i_2}, \dots\}$ where $i_n = \omega(n)$ so that L' is a sparse subsequence of L . Then, the adversary mainly enumerates from L' , occasionally outputting from L so that the enumeration is a valid $o(1)$ -noise enumeration of both L', L . Roughly speaking, under this enumeration, any algorithm cannot distinguish whether the target language is L or L' and is forced to generate from the sparse subset L' . However, this worst case enumeration cannot be bounded w.r.t. L since $i_n = \omega(n)$. If we are promised that the enumeration is M -bounded w.r.t. the target language, the adversary cannot maliciously enumerate from such a sparse subset of the target language.

7.1 Properties of Bounded Displacement Adversaries

In order to develop a better understanding of [Definition 11](#), we study several elementary properties.

Closed under Subsets. An immediate but useful feature of bounded displacement enumerations is that they are closed under language subsets.

Proposition 7.4. *Suppose the enumeration $x_{1:\infty}$ is an M -bounded displacement enumeration with respect to some language L . Then for any $L' \subseteq L$, $x_{1:\infty}$ is an M -bounded displacement enumeration w.r.t. L' .*

Proof. Let $\{\ell_1, \ell_2, \dots\} = L$ be the canonical enumeration of L . The canonical enumeration of L' can be obtained from that of L by deleting some elements and decreasing the indices of remaining elements. In other words, there is some subsequence $i_1 \leq i_2 \leq \dots$ such that $\{\ell_{i_1}, \ell_{i_2}, \dots\} = L'$ is the canonical enumeration of L' . For any $x = \ell_{i_j} \in L'$, its index $j = \sigma(x, L')$ in L' is at most its index $i_j = \sigma(x, L)$ in L . Hence for all sufficiently large n ,

$$\sigma(x_n, L') \leq \sigma(x_n, L) \leq Mn,$$

as desired. □

Change of Density. The next lemma is the key lemma for generation with lower density. Intuitively, it says that we can estimate the density of a language up to a multiplicative factor by simply computing the empirical “density” w.r.t. the *input enumeration*. This is key since we do not know the canonical enumeration of the target language (otherwise generation with density is trivial) but we do have access to the input enumeration.

Lemma 7.5 (Change of Density). *Let $x_{1:\infty}$ be an M -bounded displacement enumeration of a language K with arbitrary noise rate (and possibly omissions). Then for any other language $L \subseteq K$,*

$$\begin{aligned} \mu_{\text{low}}(L, K) &\geq \frac{1}{M} \cdot \liminf_n \frac{|L \cap x_{1:n}|}{n} \\ \mu_{\text{up}}(L, K) &\geq \frac{1}{M} \cdot \limsup_n \frac{|L \cap x_{1:n}|}{n}. \end{aligned}$$

Proof. By [Proposition 7.4](#), $x_{1:\infty}$ is also an M -bounded displacement enumeration w.r.t. L . By [Definition 11](#), for each $n \geq n^*$, we have $\sigma(x_n, L) \leq Mn$. Define $B := \max\{\sigma(x_n, L) : n < n^*\}$ and consider the canonical enumeration $\{\ell_1, \ell_2, \dots\}$ of K . For $n \geq \max(n^*, B)$, we have $L \cap \{x_1, \dots, x_n\} \subseteq L \cap \{\ell_1, \dots, \ell_{Mn}\}$, where we understand ℓ_0 to be a special string not in K . This implies that

$$\begin{aligned} \frac{|L \cap \{x_1, \dots, x_n\}|}{n} &\leq \frac{|L \cap \{\ell_1, \dots, \ell_{Mn}\}|}{n} \\ &= M \cdot \frac{|L \cap \{\ell_1, \dots, \ell_{Mn}\}|}{Mn}. \end{aligned}$$

Note that the limit inferior (superior) of the RHS is exactly the lower (upper) density of L in K . Thus taking the limit inferior (superior) on both sides concludes the proof. □

Hardness of Identification. Since we have introduced a new model that is limiting the power of the adversary, it is worth understanding how restrictive this change is. Towards that end, we study how the landscape of identification changes under our definition. Our result below shows that, even for simple collections of languages, identification remains intractable for any $M > 1$. We leave a full characterization of identification for M -bounded adversaries as an interesting open question.

Theorem 7.6. *There is a countable collection of languages $\mathcal{L}$ for which no algorithm can identify in the limit, even when restricted to M -bounded enumerations for any $M > 1$.*

Proof. Let $\mathcal{L} = \{\mathbb{N}\} \cup \{\mathbb{N} \setminus \{n\} : n \in \mathbb{N}\}$ be the collection consisting of the natural numbers and all its subsets obtained by deleting a single number. Let $\mathcal{A}$ be any algorithm.

The adversary begins by enumerating the natural numbers $1, 2, \dots$. If $\mathcal{A}$ never outputs $\mathbb{N}$, we are done. Let n_1 be the first step when $\mathcal{A}$ outputs $\mathbb{N}$. Then at step $n_1 + 1$, we output $n_1 + 2$ instead of $n_1 + 1$ and then continue enumerating the natural numbers in the canonical order. If $\mathcal{A}$ never outputs $\mathbb{N} \setminus \{n_1 + 1\}$, we are done. Otherwise, let $n_2 > n_1$ be the first step when $\mathcal{A}$ outputs $\mathbb{N} \setminus \{n_1 + 1\}$. At step $n_2 + 1$, the adversary “fills in” the previously skipped element by outputting $n_1 + 1$, instead of $n_2 + 1$, and then continues enumerating the natural numbers in the canonical order again. Repeat.

All in all, the adversary can always ensure that $\mathcal{A}$ does not correctly identify after any finite time. On the other hand, if we let $\sigma(n)$ denote the value of the element the adversary enumerates at step n , then $\sigma(n) \leq n + 1$. Thus identification in the limit remains impossible in general even under M -bounded displacement enumerations, for any $M > 1$. $\square$

Hardness of Generation under Constant Noise. To further understand the landscape of generation under bounded displacement enumerations, and as earlier promised, we show that generation with finite noise remains hard.

Theorem 7.7. *For any $k \geq 2$, there is a finite collection of k languages $\mathcal{L}$ for which no algorithm can generate in the limit from $\mathcal{L}$ under constant $1/k$ -rate noise, even when restricted to $1/k$ -bounded enumerations.*

Proof. Consider the finite family $\mathcal{L}$ from [Example 7.2](#). Let $k \geq 2$ be an integer and define the language

$$L_i := [k] \cup (\mathbb{N} \setminus i \cdot \mathbb{N}) = \{1, \dots, k\} \cup \{n \in \mathbb{N} : n \text{ is not divisible by } i\}.$$

Then $\mathcal{L} = \{L_i : i \in [k]\}$.

As we have seen in [Example 7.2](#), the canonical enumeration of the natural numbers is $1/k$ -bounded and has $1/k$ -noise w.r.t. every $L_i, i \in [k]$. Thus any hypothetical algorithm that generates from $\mathcal{L}$ under the assumed conditions must simultaneously generate from all L_i . In other words, it generates from $\bigcap_{i \in [k]} L_i = [k]$. This must be a contradiction as this is a finite set. $\square$

All in all, we have seen that bounded displacement enumerations does not trivialize the identification problem, nor generation under constant rate noise. [Section 7.2](#) shows that we do gain an advantage for the regime of $o(1)$ -noise in terms of generation with density.

7.2 Results for Set-Based Density

We now present our results for generation with set-based density guarantee under bounded displacement adversaries. In [Section 7.2.1](#), we first design an algorithm which roughly obtains $1/M$ set-based lower density under M -bounded enumerations. [Section 7.2.2](#) demonstrates that this guarantee is tight in the worst case sense: In general, no algorithm can do better than $1/M$ in this setting.

7.2.1 Generation with Set-Based Lower Density

Having described the beyond-worst-case model we consider, we show how these types of enumerations help achieve non-trivial lower density (in a set-based sense) under vanishing noise rate for *all* countable collections. At each step, our algorithm outputs a finite set of languages indexed by $I(n) \subseteq \mathbb{N}$ such that for sufficiently large n , their intersection is an infinite subset of the target, i.e., $\cap_{i \in I(n)} L_i \subseteq K$ (generation in the limit), and is dense in K .

Theorem 7.8. *There is an algorithm $\mathbb{G}$ that, for any collection $\mathcal{L}$, target language $K \in \mathcal{L}$, and error parameter $\varepsilon \in (0, 1)$, given an M -bounded displacement enumeration $x_{1:\infty}$ with $o(1)$ -noise and arbitrary omissions, $\mathbb{G}$ outputs an intersection $\cap_{i \in I(n)} L_i$ of finitely many languages $|I(n)| < \infty$ with infinite cardinality $|\cap_{i \in I(n)} L_i| = \infty$ at step n such that*

- (a) $\cap_{i \in I(n)} L_i \subseteq K$ for all sufficiently large n and
- (b) $\mu_{\text{low}} \left(\cap_{i \in I(n)} L_i, K \right) \geq \frac{1-\varepsilon}{M}$ for all sufficiently large n .

Pseudocode. The pseudocode for our algorithm is an instantiation of our generic meta-algorithm [Algorithm 1](#) and is presented in [Algorithm 9](#). We highlight the changes in blue. We remark that the algorithm requires two non-standard parameters as input. $\varepsilon \in (0, 1)$ is an error parameter which is required to compute priorities appropriately. The parameter $M \geq 1$ is the boundedness parameter ([Definition 11](#)), which is required to correctly compute both the priorities and the stopping rule. [Algorithm 9](#) ensures $\frac{1-\varepsilon}{M}$ lower density.

Recall the following notation for a string x and language L with canonical enumeration $L = \{\ell_1, \ell_2, \dots\}$

$$\sigma(x, L) := \begin{cases} j, & x = \ell_j \in L, \\ 0, & x \notin L. \end{cases}$$

In other words, $\sigma(x, L)$ is the index of x in the canonical enumeration of L if $x \in L$ and otherwise $\sigma(x, L) = 0$ if $x \notin L$. As a shorthand for an enumeration $x_{1:\infty}$, we write

$$\sigma(x_{1:n}, L) := \max_{j \in [n]} \sigma(x_j, L).$$

We remark that by [Definition 11](#), any M -bounded enumeration $x_{1:\infty}$ of L satisfies $\sigma(x_{1:n}; L) \leq Mn$ for all sufficiently large n .

Analysis. Recall the following notation for a given $p \geq 1$

$$P_i^\infty = \lim_{n \rightarrow \infty} P_i^{(n)}, \quad \mathcal{L}(p) = \{L_i : P_i^\infty \leq p\}, \quad \text{Cl}(\mathcal{L}(p)) = \cap_{L \in \mathcal{L}(p)} L.$$

Algorithm 9 Algorithm for [Theorem 7.8](#)

Input: Countable collection $\mathcal{L} = \{L_1, L_2, \dots\}$; density error parameter $\varepsilon \in (0, 1)$; enumeration $x_{1:\infty}$; bounded displacement parameter $M \geq 1$

- 1: Let $S_n \leftarrow \{x_1, \dots, x_n\}$ be the set of examples seen in the first n steps
- 2: Let $W_{n-1} \leftarrow \{w_1, \dots, w_{n-1}\}$ be the set of strings output before the n -th step
- 3: **for** $i = 1, 2, \dots, n$ **do**
- 4: Compute the smallest $N_i^{(n)}$ such that L_i is nearly consistent with $x_{1:m}$ and $x_{1:m}$ is M -bounded w.r.t. L_i , for every $N_i^{(n)} \leq m \leq n$

$$N_i^{(n)} \leftarrow \begin{cases} \min \{N \geq 1 : \forall m \in \{N, \dots, n\}, R_m(L_i; x_{1:m}) \leq 2^{-i}\varepsilon \wedge \sigma(x_{1:m}; L_i) \leq Mm\} , & \text{exists,} \\ n+1, & \text{else.} \end{cases}$$

- 5: Assign the language L_i a priority of $P_i^{(n)} \leftarrow i + N_i^{(n)}$
- 6: **end for**
- 7: Re-order $\{L_1, \dots, L_n\}$ in increasing priority, tie-breaking by index, as $\{L_{i_n(1)}, \dots, L_{i_n(n)}\}$, i.e., for each $j \in [n-1]$, ensure either $P_{i_n(j)}^{(n)} < P_{i_n(j+1)}^{(n)}$ or $(P_{i_n(j)}^{(n)} = P_{i_n(j+1)}^{(n)}) \wedge (i_n(j) < i_n(j+1))$.
- 8: Compute the largest index $J(n)$ such that the intersection in the re-ordering up to $L_{i_n(J_n)}$ is dense in $L_{i_n(j)}$ for all $j \leq J_n$

$$J_n \leftarrow \max \left\{ \bar{j} \in [n] : \forall j \leq \bar{j}, \mu_{\text{low}} \left(\cap_{j \leq J_n} L_{i_n(j)}, L_{i_n(n)} \right) \geq \frac{1-\varepsilon}{M} \right\}.$$

- 9: **Output** $\cap_{j \leq J_n} L_{i_n(j)}$.
-

Proof of Theorem 7.8. Let $\mathcal{H}_n := \{L_{i_n(j)} : j \leq J_n\}$ denote the prefix class. Write $K = L_{i^*}$ for the target language and define $p := P_{i^*}^\infty < \infty$, which is guaranteed to exist by the assumption that $x_{1:\infty}$ is a valid M -bounded $o(1)$ -noisy enumeration of L_{i^*} . We have $L_{i^*} \in \mathcal{L}(p)$ by construction. We will argue that $\mathcal{L}(p) \subseteq \mathcal{H}_n$ for all sufficiently large n so that the output $\mathcal{L}(\mathcal{H}_n)$ satisfies $\mu_{\text{low}}(\text{Cl}(\mathcal{H}_n), K) \geq \frac{1-\varepsilon}{M}$ by the definition of [Algorithm 9](#) since $K \in \mathcal{L}(p) \subseteq \mathcal{H}_n$.

We first note that the priority of each language is increasing in n by definition. Hence by [Lemma 4.1](#), there is some $n^* \in \mathbb{N}$ such that for $n \geq n^*$, the priority $P_i^{(n)} = P_i^\infty$ of $L_i \in \mathcal{L}(p)$ no longer changes and moreover, $P_i^{(n)}$ takes on strictly smaller value than any language not in $\mathcal{L}(p)$. In other words, $\mathcal{L}(p)$ is ordered before all other languages. Thus, in order to argue that the stopping rule J_n stops after considering all members of $\mathcal{L}(p)$, it suffices to show that for any subset $\mathcal{L}' \subseteq \mathcal{L}(p)$, $\mu_{\text{low}}(\text{Cl}(\mathcal{L}'), L) \geq \frac{1-\varepsilon}{M}$ for every $L \in \mathcal{L}'(p)$.

Now, by the definition of the priorities in [Algorithm 9](#) for $n \geq n^*$ every $L \in \mathcal{L}' \subseteq \mathcal{L}(p)$ satisfies

$$R(L; x_{1:n}) := \frac{|x_{1:n} \setminus L|}{n} \leq \frac{\varepsilon}{2^i}.$$

Thus for $n \geq n^*$, and any non-empty $\mathcal{L}' \subseteq \mathcal{L}(p)$, we have

$$\frac{|\text{Cl}(\mathcal{L}') \cap x_{1:n}|}{n} \geq 1 - \sum_{i: L_i \in \mathcal{L}'} \frac{|x_{1:n} \setminus L_i|}{n} \geq 1 - \sum_{i \geq 1} \frac{\varepsilon}{2^i} \geq 1 - \varepsilon.$$

Also by the definition of the priorities in [Algorithm 9](#), for $n \geq n^*$ every $L \in \mathcal{L}' \subseteq \mathcal{L}(p)$ satisfies

$$\sigma(x_{1:n}, L) \leq Mn.$$

In other words, the input enumeration $x_{1:\infty}$ is also an M -bounded enumeration w.r.t. L . Thus, the change in density formula ([Lemma 7.5](#)) applies to the languages $\text{Cl}(\mathcal{L}') \subseteq L$ and we conclude that for all $L \in \mathcal{L}'$,

$$\begin{aligned} \mu_{\text{low}}(\text{Cl}(\mathcal{L}'), L) &\geq \frac{1}{M} \liminf_n \frac{|\text{Cl}(\mathcal{L}') \cap x_{1:n}|}{n} \\ &\geq \frac{1 - \varepsilon}{M}. \end{aligned}$$

This concludes the proof by our initial remark. $\square$

Implications to Generation without Noise. Interestingly, [Theorem 7.8](#) has direct implications for generation without noise, that could be of interest beyond the scope of our work. In particular, since this result shows that (even with noise) if we restrict the adversary to enumerations with M -bounded displacements, we can achieve $1/M$ lower density in a set-based manner for *all* countable collections. Recall that [\[KW25\]](#) showed that one can achieve c set-based lower density under worst-case enumerations if and only if $\mathcal{L}$ does not contain an infinite perfect tower, parametrized by c . Our result shows that this lower bound can be circumvented by restricting the adversary to $1/c$ bounded enumerations. We leave a complete characterization of the landscape of this beyond-worst-case generation without noise as an interesting open direction.

7.2.2 Lower Bound for Set-Based Density

In the previous sections, we demonstrated that given an M -bounded enumeration, it is always possible to perform set-based generation with set-based lower density $\frac{1-\varepsilon}{M}$, where $\varepsilon \in (0, 1)$ is arbitrary. A natural question is whether this is the best possible. In this section, we show that in the worst case, no generator can obtain density greater than $\frac{1}{M}$, even *upper density*.

Theorem 7.9. *There exists a collection of two languages L_1, L_2 such that if the adversary provides an M -bounded displacement enumeration with noise rate $o(1)$ from this collection, then any generator that generates from the target K in the limit achieves set-based upper density at most $1/M$.*

The proof is via a similar construction as [Theorem 6.21](#), except we must construct the enumeration and collection slightly more carefully to satisfy M -boundedness.

Proof. Consider two languages $L_1 = M \cdot \mathbb{N} := \{M \cdot n : n \in \mathbb{N}\}$ and $L_2 = \mathbb{N}$ with the property that $L_1 \subseteq L_2$ and $\mu_{\text{up}}(L_1, L_2) = \frac{1}{M}$. Consider the following enumeration of the elements of L_2 .

Define a scheduling of time steps $T = \{1, 2, 4, 8, \dots\}$. We only need the fact that T has density 0 in $\mathbb{N}$. Let $x_{1:\infty}$ be the following enumeration: At time $n \in T$, the adversary outputs the next

element in $L_2 \setminus L_1$ that it has not output. At time $n \notin T$, the adversary outputs the next element in L_1 that it has not output. Now, $\sigma(x_n, L_1) \leq n$ since we can interpret $x_{1:\infty}$ as the canonical enumeration of $L_1 = M \cdot \mathbb{N}$ with some elements of $L_2 \setminus L_1$ interleaved. Note that the latter can only decrease $\sigma(x_n, L_1)$. On the other hand

$$\sigma(x_n, L_2) \leq \begin{cases} n, & n \in T, \\ M \cdot n, & n \notin T. \end{cases}$$

Hence $x_{1:\infty}$ is an M -bounded enumeration w.r.t. both L_1, L_2 .

Note that the adversary will enumerate all the elements of L_2 , and thus all the elements of L_1 . Clearly, the adversary enumerates from L_2 with noise rate 0. On the other hand, since the adversary enumerates elements not belonging to L_1 only at time $n \in T$, and T has density 0 in $\mathbb{N}$, the noise rate of the adversary's enumeration is also $o(1)$ in L_1 .

Since the enumeration is a valid enumeration with $o(1)$ -noise for both L_1 and L_2 , and $L_1 \subsetneq L_2$, any set-based generator generating from K in the limit has to output $S_n \subseteq L_1$ for all sufficiently large n . Since L_1 has upper density $1/M$ in L_2 , the generator achieves set-based upper density at most $1/M$ if the adversary sets $K = L_2$. $\square$

7.3 Results for Element-Based Density

In this section, we extend our results for set-based density to element-based density. [Section 7.3.1](#) applies our black-box reduction from set-based generation with density to roughly attain element-based density $\frac{1}{2M}$ with noise under M -bounded displacement enumerations. Similarly, we show that no algorithm can do better than $\frac{1}{2M}$ in general within [Section 7.3.2](#). Note that, unlike the set-based density guarantee, our upper and lower bounds have a gap of $\frac{1}{2M}$.

7.3.1 Generation with Element-Based Lower Density

In this section, we design an algorithm for element-based generation with lower density given an M -bounded enumeration.

Theorem 7.10. *There is an generator $\mathbb{G}$ that, for any collection $\mathcal{L}$, target language $K \in \mathcal{L}$, and error parameter $\varepsilon \in (0, 1)$, given an M -bounded displacement enumeration $x_{1:\infty}$ with $o(1)$ -noise rate and arbitrary omissions, the output $w_{1:\infty}$ of $\mathbb{G}$ has lower density*

$$\mu_{low}(w_{1:\infty}, K) \geq \frac{1 - \varepsilon}{2M}.$$

Having completed the heavy lifting of set-based generation with density in [Section 7.2.1](#), we can now leverage our black-box reduction from set-based generation with density to element-based generation with density from [Section 6.2.1](#) to design a simple algorithm.

Proof. Our [Algorithm 9](#) achieving lower set-based density ([Theorem 7.8](#)) falls under the set-based generation algorithm in the premise of [Theorem 6.15](#). Hence, we can apply the black-box reduction from [Theorem 6.15](#) to construct the desired generator. $\square$

7.3.2 Lower Bound for Element-Based Upper Density

We now demonstrate that the same counterexample for set-based density also extends to the element-based density under bounded displacement enumerations.

Theorem 7.11. *There exists a collection of two languages L_1, L_2 such that if the adversary provides an M -bounded displacement enumeration with $o(1)$ -noise from this collection, then any generator that generates from the target K in the limit achieves element-based upper density at most $1/M$.*

The proof is identical to that of [Theorem 7.9](#). In the setting where arbitrary enumeration is allowed, the adversary can choose to skip enumerating elements that the generator outputs and force the generator to obtain density at most $1/2$. We can picture this as skipping every other element of some given enumeration. However, if the original enumeration is M -bounded, skipping every other element leads to a $2M$ -bounded enumeration. Thus, the adversary's strategy for arbitrary enumerations does not hold for bounded enumerations. This is the main reason we have a gap of $\frac{1}{2M}$ between the density guarantee of [Theorem 7.10](#) and the lower bound in [Theorem 7.11](#).

Proof. Consider two languages $L_1 = M \cdot \mathbb{N} := \{M \cdot n : n \in \mathbb{N}\}$ and $L_2 = \mathbb{N}$ with the property that $L_1 \subseteq L_2$ and $\mu_{\text{up}}(L_1, L_2) = \frac{1}{M}$. Consider the following enumeration of the elements of L_2 .

Define a scheduling of time steps $T = \{1, 2, 4, 8, \dots\}$. We only need the fact that T has density 0 in $\mathbb{N}$. Let $x_{1:\infty}$ be the following enumeration: At time $n \in T$, the adversary outputs the next element in $L_2 \setminus L_1$ that it has not output. At time $n \notin T$, the adversary outputs the next element in L_1 that it has not output. Now, $\sigma(x_n, L_1) \leq n$ since we can interpret $x_{1:\infty}$ as the canonical enumeration of $L_1 = M \cdot \mathbb{N}$ with some elements of $L_2 \setminus L_1$ interleaved. Note that the latter can only decrease $\sigma(x_n, L_1)$. On the other hand

$$\sigma(x_n, L_2) \leq \begin{cases} n, & n \in T, \\ M \cdot n, & n \notin T. \end{cases}$$

Hence $x_{1:\infty}$ is an M -bounded enumeration w.r.t. both L_1, L_2 .

Note that the adversary will enumerate all the elements of L_2 , and thus all the elements of L_1 . Clearly, the adversary enumerates from L_2 with noise rate 0. On the other hand, since the adversary enumerates elements not belonging to L_1 only at time $n \in T$, and T has density 0 in $\mathbb{N}$, the noise rate of the adversary's enumeration is also $o(1)$ in L_1 .

Since the enumeration is a valid $o(1)$ -noisy enumeration for both L_1 and L_2 , and $L_1 \subsetneq L_2$, any set-based generator generating from K in the limit has to output $x_n \in L_1$ for all sufficiently large n . Since L_1 has upper density $1/M$ in L_2 , the generator achieves set-based upper density at most $1/M$ if the adversary sets $K = L_2$. $\square$

We leave as an open question the tight density guarantee given an M -bounded enumeration.

Acknowledgements

Felix Zhou acknowledges the support of the Natural Sciences and Engineering Research Council of Canada (NSERC).

References

- [AML25] Awni Altabaa, Omar Montasser, and John Lafferty. “CoT Information: Improved Sample Complexity under Chain-of-Thought Supervision”. In: *arXiv preprint arXiv:2505.15927* (2025) (cit. on p. 14).
- [Ang79] Dana Angluin. “Finding Patterns Common to a Set of Strings (Extended Abstract)”. In: *Proceedings of the Eleventh Annual ACM Symposium on Theory of Computing*. STOC ’79. Atlanta, Georgia, USA: Association for Computing Machinery, 1979, pp. 130–141 (cit. on p. 1).
- [Ang80] Dana Angluin. “Inductive Inference of Formal Languages From Positive Data”. In: *Information and Control* 45.2 (1980), pp. 117–135 (cit. on pp. 1, 11).
- [Ang88] Dana Angluin. *Identifying Languages From Stochastic Examples*. Yale University. Department of Computer Science, 1988 (cit. on p. 13).
- [AS23] Josh Alman and Zhao Song. “Fast Attention Requires Bounded Entries”. In: *Advances in Neural Information Processing Systems*. Ed. by A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine. Vol. 36. Curran Associates, Inc., 2023, pp. 63117–63135 (cit. on p. 1).
- [AS24] Josh Alman and Zhao Song. “The Fine-Grained Complexity of Gradient Computation for Training Large Language Models”. In: *Advances in Neural Information Processing Systems*. Ed. by A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang. Vol. 37. Curran Associates, Inc., 2024, pp. 61419–61454 (cit. on p. 1).
- [BJS92] Ganesh Baliga, Sanjay Jain, and Arun Sharma. “Learning from multiple sources of inaccurate data”. In: *International Workshop on Analogical and Inductive Inference*. Springer. 1992, pp. 108–128 (cit. on p. 12).
- [BLCW09] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. “Curriculum learning”. In: *Proceedings of the 26th annual international conference on machine learning*. 2009, pp. 41–48 (cit. on pp. 4, 8, 49).
- [Bou+21] Olivier Bousquet, Steve Hanneke, Shay Moran, Ramon van Handel, and Amir Yehudayoff. “A Theory of Universal Learning”. In: *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*. STOC 2021. Virtual, Italy: Association for Computing Machinery, 2021, pp. 532–541 (cit. on p. 13).
- [BPZ26] Yannan Bai, Debmalya Panigrahi, and Ian Zhang. “Language Generation in the Limit: Noise, Loss, and Feedback”. In: *Proceedings of the 2026 ACM–SIAM Symposium on Discrete Algorithms (SODA’26)*. To appear. SIAM, 2026 (cit. on pp. 2, 3, 6, 9, 10, 12, 13, 17, 19).
- [Che+23] Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. *Accelerating Large Language Model Decoding with Speculative Sampling*. 2023 (cit. on p. 1).
- [Cho56] Noam Chomsky. “Three Models for the Description of Language”. In: *IRE Transactions on Information Theory* 2.3 (1956), pp. 113–124 (cit. on p. 1).
- [CJS97] John Case, Sanjay Jain, and Arun Sharma. “Synthesizing noise-tolerant language learners”. In: *International Workshop on Algorithmic Learning Theory*. Springer. 1997, pp. 228–243 (cit. on p. 12).
- [CP25a] Moses Charikar and Chirag Pabbaraju. “Exploring Facets of Language Generation in the Limit”. In: *Thirty-eighth Conference on Learning Theory (COLT 2025)*. Ed. by Nika Haghtalab and Ankur Moitra. Proceedings of Machine Learning Research. PMLR, 2025 (cit. on pp. 1, 2, 10–14, 16, 19–22, 24).

- [CP25b] Moses Charikar and Chirag Pabbaraju. *Pareto-optimal Non-uniform Language Generation*. 2025 (cit. on pp. [2](#), [14](#), [16](#)).
- [CPW24] Lijie Chen, Binghui Peng, and Hongxun Wu. *Theoretical limitations of multi-layer Transformer*. 2024 (cit. on pp. [1](#), [14](#)).
- [Dod+21] Jesse Dodge, Maarten Sap, Ana Marasović, William Agnew, Gabriel Ilharco, Dirk Groeneveld, Margaret Mitchell, and Matt Gardner. “Documenting Large Webtext Corpora: A Case Study on the Colossal Clean Crawled Corpus”. In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 1286–1305 (cit. on p. [3](#)).
- [FJ89] Mark Fulk and Sanjay Jain. “Learning in the presence of inaccurate information”. In: (1989) (cit. on p. [12](#)).
- [Gol67] E. Mark Gold. “Language Identification in the Limit”. In: *Information and Control* 10.5 (1967), pp. 447–474 (cit. on pp. [1](#), [12](#)).
- [Han+24] Insu Han, Rajesh Jayaram, Amin Karbasi, Vahab Mirrokni, David Woodruff, and Amir Zandieh. “HyperAttention: Long-context Attention in Near-Linear Time”. In: *The Twelfth International Conference on Learning Representations*. 2024 (cit. on p. [1](#)).
- [HKKT18] Steve Hanneke, Adam Tauman Kalai, Gautam Kamath, and Christos Tzamos. “Actively Avoiding Nonsense in Generative Models”. In: *Proceedings of the 31st Conference On Learning Theory*. Ed. by Sébastien Bubeck, Vianney Perchet, and Philippe Rigollet. Vol. 75. Proceedings of Machine Learning Research. PMLR, June 2018, pp. 209–227 (cit. on p. [1](#)).
- [HKMV25] Steve Hanneke, Amin Karbasi, Anay Mehrotra, and Grigoris Velegkas. *On Union-Closedness of Language Generation*. 2025 (cit. on pp. [2](#), [13](#)).
- [Hua+25] Audrey Huang, Adam Block, Qinghua Liu, Nan Jiang, Akshay Krishnamurthy, and Dylan J Foster. “Is Best-of-N the Best of Them? Coverage, Scaling, and Optimality in Inference-Time Alignment”. In: *Forty-second International Conference on Machine Learning*. 2025 (cit. on p. [1](#)).
- [HW19] Guy Hach Cohen and Daphna Weinshall. “On the power of curriculum learning in training deep networks”. In: *International conference on machine learning*. PMLR. 2019, pp. 2535–2544 (cit. on pp. [8](#), [49](#)).
- [HWL25] Jianhao Huang, Zixuan Wang, and Jason D Lee. “Transformers learn to implement multi-step gradient descent with chain of thought”. In: *arXiv preprint arXiv:2502.21212* (2025) (cit. on p. [14](#)).
- [Jai94] Sanjay Jain. “Program synthesis in the presence of infinite number of inaccuracies”. In: *Algorithmic Learning Theory*. Ed. by Setsuo Arikawa and Klaus P. Jantke. Berlin, Heidelberg: Springer Berlin Heidelberg, 1994, pp. 333–348 (cit. on p. [12](#)).
- [Jos+25] Nirmal Joshi, Gal Vardi, Adam Block, Surbhi Goel, Zhiyuan Li, Theodor Misiakiewicz, and Nathan Srebro. “A theory of learning with autoregressive chain of thought”. In: *arXiv preprint arXiv:2503.07932* (2025) (cit. on p. [14](#)).
- [KM24] Jon Kleinberg and Sendhil Mullainathan. “Language generation in the limit”. In: *Advances in Neural Information Processing Systems* 37 (2024), pp. 66058–66079 (cit. on pp. [1–4](#), [8](#), [9](#), [12](#), [13](#), [15–18](#), [25](#), [36](#), [37](#), [43](#)).
- [KMSV25] Amin Karbasi, Omar Montasser, John Sous, and Grigoris Velegkas. “(Im)possibility of Automated Hallucination Detection in Large Language Models”. In: *Second Conference on Language Modeling*. 2025 (cit. on p. [14](#)).

- [KMOV24] Alkis Kalavasis, Anay Mehrotra, and Grigoris Velegkas. *Characterizations of Language Generation With Breadth*. <https://arxiv.org/abs/2412.18530>. 2024 (cit. on pp. 1, 2, 11, 13, 16, 20).
- [KMOV25] Alkis Kalavasis, Anay Mehrotra, and Grigoris Velegkas. “On the Limits of Language Generation: Trade-Offs Between Hallucination and Mode Collapse”. In: *Proceedings of the 57th Annual ACM Symposium on Theory of Computing (STOC’25)*. Prague, Czech Republic: Association for Computing Machinery, 2025 (cit. on pp. 2, 4, 5, 9, 11, 13, 14, 16, 20).
- [KNVZ25] Adam Tauman Kalai, Ofir Nachum, Santosh S. Vempala, and Edwin Zhang. *Why Language Models Hallucinate*. 2025 (cit. on pp. 1, 14).
- [Kre+22] Julia Kreutzer, Isaac Caswell, Lisa Wang, Ahsan Wahab, Daan van Esch, Nasanbayar Ulzii-Orshikh, Allahsera Tapo, Nishant Subramani, Artem Sokolov, Claytone Sikasote, Monang Setyawan, Supheakmongkol Sarin, Sokhar Samb, Benoît Sagot, Clara Rivera, Annette Rios, Isabel Papadimitriou, Salomey Osei, Pedro Ortiz Suarez, Iroko Orife, Kelechi Ogueji, Andre Niyongabo Rubungo, Toan Q. Nguyen, Mathias Müller, André Müller, Shamsuddeen Hassan Muhammad, Nanda Muhammad, Ayanda Mnyakeni, Jamshidbek Mirzakhalov, Tapiwanashe Matangira, Colin Leong, Nze Lawson, Sneha Kudugunta, Yacine Jernite, Mathias Jenny, Orhan Firat, Bonaventure F. P. Dossou, Sakhile Dlamini, Nisansa de Silva, Sakine Çabuk Balli, Stella Biderman, Alessia Battisti, Ahmed Baruwa, Ankur Bapna, Pallavi Baljekar, Israel Abebe Azime, Ayodele Awokoya, Duygu Ataman, Orevaoghene Ahia, Oghenefego Ahia, Sweta Agrawal, and Mofetoluwa Adeyemi. “Quality at a Glance: An Audit of Web-Crawled Multilingual Datasets”. In: *Transactions of the Association for Computational Linguistics* 10 (2022), pp. 50–72 (cit. on p. 3).
- [KV24] Adam Tauman Kalai and Santosh S. Vempala. “Calibrated Language Models Must Hallucinate”. In: *Proceedings of the 56th Annual ACM Symposium on Theory of Computing*. STOC 2024. Vancouver, BC, Canada: Association for Computing Machinery, 2024, pp. 160–171 (cit. on pp. 1, 14).
- [KW25] Jon Kleinberg and Fan Wei. “Density Measures for Language Generation”. In: *Proceedings of the 66th IEEE Symposium on Foundations of Computer Science (FOCS 2025)*. To appear. IEEE, 2025 (cit. on pp. 1–5, 7, 8, 11–13, 16, 20, 26, 31, 33, 43, 47, 55, 63).
- [Lee+22] Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. “Deduplicating Training Data Makes Language Models Better”. In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 8424–8445 (cit. on p. 3).
- [LKM23] Yaniv Leviathan, Matan Kalman, and Yossi Matias. “Fast Inference from Transformers via Speculative Decoding”. In: *Proceedings of the 40th International Conference on Machine Learning*. Ed. by Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett. Vol. 202. Proceedings of Machine Learning Research. PMLR, 23–29 Jul 2023, pp. 19274–19286 (cit. on p. 1).
- [LM25] Allen Liu and Ankur Moitra. “Model stealing for any low-rank language model”. In: *Proceedings of the 57th Annual ACM Symposium on Theory of Computing*. 2025, pp. 1755–1761 (cit. on p. 14).
- [LRT25] Jiaxun Li, Vinod Raman, and Ambuj Tewari. “Generation Through the Lens of Learning Theory”. In: *Thirty-eighth Conference on Learning Theory (COLT 2025)*. Ed. by Nika Haghtalab and Ankur Moitra. Proceedings of Machine Learning Research. PMLR, 2025 (cit. on pp. 2, 12).
- [Mal23] Eran Malach. “Auto-regressive next-token predictors are universal learners”. In: *arXiv preprint arXiv:2309.06979* (2023) (cit. on p. 14).

- [MKKZ23] Gaurav Mahajan, Sham Kakade, Akshay Krishnamurthy, and Cyril Zhang. “Learning hidden markov models using conditional samples”. In: *The Thirty Sixth Annual Conference on Learning Theory*. PMLR. 2023, pp. 2014–2066 (cit. on p. 14).
- [MS03a] Yasuhito Mukouchi and Masako Sato. “Refutable language learning with a neighbor system”. In: *Theoretical computer science* 298.1 (2003), pp. 89–110 (cit. on p. 12).
- [MS03b] Yasuhito Mukouchi and Masako Sato. “Refutable language learning with a neighbor system”. In: *Theoretical Computer Science* 298.1 (2003). Selected Papers in honour of Setsuo Arikawa, pp. 89–110 (cit. on p. 12).
- [NRT25] Ansh Nagda, Prabhakar Raghavan, and Abhradeep Thakurta. *Reinforced Generation of Combinatorial Structures: Applications to Complexity Theory*. 2025 (cit. on p. 1).
- [PNP24] Binghui Peng, Srini Narayanan, and Christos Papadimitriou. “On Limitations of the Transformer Architecture”. In: *First Conference on Language Modeling*. 2024 (cit. on pp. 1, 14).
- [PRR25] Charlotte Peale, Vinod Raman, and Omer Reingold. “Representative Language Generation”. In: *Forty-second International Conference on Machine Learning*. 2025 (cit. on pp. 2, 14).
- [Raf+20] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. “Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer”. In: *Journal of Machine Learning Research* 21.140 (2020), pp. 1–67 (cit. on p. 3).
- [RR25] Ananth Raman and Vinod Raman. “Generation from Noisy Examples”. In: *Forty-second International Conference on Machine Learning*. 2025 (cit. on pp. 2, 3, 5–7, 9, 12, 13, 17, 25, 27, 66).
- [Rud76] Walter Rudin. *Principles of Mathematical Analysis*. 3rd ed. McGraw–Hill, 1976 (cit. on p. 68).
- [Sch85] Gisela Schäfer. “Some results in the theory of effective program synthesis: Learning by defective information”. In: *Int. Spring School on Mathematical Methods of Specification and Synthesis of Software Systems*. Springer, 1985, pp. 219–225 (cit. on p. 12).
- [Sha51] Claude E. Shannon. “Prediction and Entropy of Printed English”. In: *The Bell System Technical Journal* 30.1 (1951), pp. 50–64 (cit. on p. 1).
- [SHT23] Clayton Sanford, Daniel J Hsu, and Matus Telgarsky. “Representational Strengths and Limitations of Transformers”. In: *Advances in Neural Information Processing Systems*. Ed. by A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine. Vol. 36. Curran Associates, Inc., 2023, pp. 36677–36707 (cit. on pp. 1, 14).
- [Sip12] Michael Sipser. *Introduction to the Theory of Computation*. Introduction to the Theory of Computation. Cengage Learning, 2012 (cit. on p. 1).
- [Ste97] Frank Stephan. “Noisy inference and oracles”. In: *Theoretical Computer Science* 185.1 (1997), pp. 129–157 (cit. on p. 12).
- [THJ06] Frédéric Tantini, Colin de la Higuera, and Jean-Christophe Janodet. “Identification in the Limit of Systematic-Noisy Languages”. In: *Grammatical Inference: Algorithms and Applications*. Ed. by Yasubumi Sakakibara, Satoshi Kobayashi, Kengo Sato, Tetsuro Nishino, and Etsuji Tomita. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 19–31 (cit. on p. 12).
- [Tur50] Alan M. Turing. “Computing Machinery and Intelligence”. In: *Mind* LIX.236 (1950), pp. 433–460 (cit. on p. 1).
- [Vaf+24] Keyon Vafa, Justin Y Chen, Ashesh Rambachan, Jon Kleinberg, and Sendhil Mullainathan. “Evaluating the world model implicit in a generative model”. In: *Advances in Neural Information Processing Systems* 37 (2024), pp. 26941–26975 (cit. on p. 14).

- [Wan+20] Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. *Linformer: Self-Attention with Linear Complexity*. 2020 (cit. on p. 1).
- [WCM22] Colin Wei, Yining Chen, and Tengyu Ma. “Statistically Meaningful Approximation: a Case Study on Approximating Turing Machines with Transformers”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh. Vol. 35. Curran Associates, Inc., 2022, pp. 12071–12083 (cit. on p. 1).
- [XJK24] Ziwei Xu, Sanjay Jain, and Mohan Kankanhalli. *Hallucination is Inevitable: An Innate Limitation of Large Language Models*. 2024 (cit. on p. 14).
- [Zah+20] Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. “Big Bird: Transformers for Longer Sequences”. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 17283–17297 (cit. on p. 1).

A Generation with Repetitions in the Input Enumeration

We first remark that our hardness results hold when allowing for repetitions as we have proven them for the special case with 0 repetition.

We briefly sketch how to handle repetitions in the input enumeration within our algorithms. For an enumeration $x_1, x_2, \dots$, possibly with repetitions, we define the sub-enumeration $x_{i_1}, x_{i_2}, \dots$ consisting of the first occurrence of the unique elements in $x_{1:\infty}$. That is, $x_{i_1:\infty}$ is obtained from $x_{1:\infty}$ by deleting elements x_n that already occurred at some step $m < n$. Modify all our assumptions so that they hold with respect to the unique elements observed so far. For example, redefine the empirical noise rate so that the denominator is the number of unique elements seen so far

$$R(x_{1:n}, L) := \frac{|x_{1:n} \setminus L|}{|x_{1:n}|}.$$

Hypothetically, if we could execute our algorithms on $x_{i_1:\infty}$, we would be done. This is essentially the case for set-based generation, as running our algorithm on $x_{1:\infty}$ is equivalent to running our algorithm on $x_{i_1:\infty}$, except we can simply output the same set on time steps when the adversary enumerates a repeated element. Note that this suffices for both generation and generation with density.

The case of element-based generation is slightly more involved. Fortunately, our element-based algorithms (with the exception of our reduction to [KW25], which already handles repetitions) first compute an infinite set as a function of only the adversary's enumeration, and then output an unseen element from this set. Thus we can also perform the same reduction as set-based generation, with the exception that on time steps when the adversary enumerates a repeated element, we output the first unseen element from the set chosen in the last step when the adversary did not enumerate a repeat. It is easy to see that this algorithm generates in the limit as long as the original algorithm generates in the limit with no repetitions. To see that density guarantees still hold, we remark that the set of elements generated on input $x_{1:n}$ is a superset of the elements generated on $x_{i_1:n}$. Indeed, in step n when the adversary enumerates a unique element, the algorithm will have identified the same set as the hypothetical algorithm executed on the unique sub-enumeration. It will also output the same first unseen element, unless it has already been output due to a previous step caused by a repetition. Note here we crucially used the fact that the infinite set we are picking from is only a function of the (unique) input enumeration.

B Additional Results

B.1 Generation with Vanishing and Unknown Noise Rate: Sorting by Index Only

Fix a countable collection of languages $\mathcal{L}$. Set $c_i := \frac{1}{2^{i+1}}$. At time n , let $\mathcal{L}^{(n)} \subseteq \mathcal{L}$ be the subcollection of languages L_i with empirical noise rate $R(L_i; x_{1:n}) \leq c_i$.

Proposition B.1. *Fix a subset of languages $\mathcal{L}_S := \{L_i : i \in S\}$. If $|\text{Cl}(\mathcal{L}_S)| < \infty$, then there exists T such that for $n \geq T$, $\mathcal{L}_S \not\subseteq \mathcal{L}^{(n)}$, i.e., there exists at least one language $L_i \in \mathcal{L}_S \setminus \mathcal{L}^{(n)}$.*

Proof. Assume that $\mathcal{L}_S \subseteq \mathcal{L}^{(n)}$. Then, we have

$$\begin{aligned} |\text{Cl}(\mathcal{L}_S) \cap \{x_1, \dots, x_n\}| &= |\{x_1, \dots, x_n\}| - \left| \bigcup_{i \in S} (\{x_1, \dots, x_n\} \setminus L_i) \right| \\ &\geq n - \sum_{i \in S} |\{x_1, \dots, x_n\} \setminus L_i| \\ &\geq n - \sum_{i \in S} n \cdot c_i \end{aligned}$$

since $L_i \in \mathcal{L}_S \subseteq \mathcal{L}^{(n)}$,

$$\geq \frac{n}{2},$$

where the last inequality follows from $\sum_{i=1}^{\infty} c_i \leq \frac{1}{2}$.

Since $|\text{Cl}(\mathcal{L}_S)| < \infty$, we know that the above inequality stops holding if $n > 2|\text{Cl}(\mathcal{L}_S)|$. Thus, there is a finite time T so that for $n \geq T$, $\mathcal{L}_S \not\subseteq \mathcal{L}^{(n)}$. $\square$

Now we prove the following algorithm also generates in the limit under vanishing noise rate, where we replaced the priority in [Algorithm 4](#) with simply sorting by indices of the languages with low empirical noise rate. Note that in particular, this algorithm does not enjoy the stable prefix property guaranteed by [Lemma 4.1](#).

Algorithm 10 Algorithm for [Theorem B.2](#)

Input: Countable collection $\mathcal{L} = \{L_1, L_2, \dots\}$; thresholds $c_1, c_2, \dots \in (0, 1)$; enumeration $x_{1:\infty}$

- 1: Let $S_n \leftarrow \{x_1, \dots, x_n\}$ be the set of examples seen in the first n steps
- 2: Let $W_{n-1} \leftarrow \{w_1, \dots, w_{n-1}\}$ be the set of strings output before the n -th step
- 3: Let $\mathcal{A}_n \leftarrow \{L_i : R(L_i; x_{1:n}) \leq c_i, 1 \leq i \leq n\}$ be the active set of languages with empirical noise rate bounded by c_i
- 4: **Compute the longest prefix $\mathcal{L}_n$ of $\mathcal{A}_n$ with infinite intersection, i.e.,**

$$\mathcal{L}_n \leftarrow \sup \{L_i \in \mathcal{A}_n : |\text{Cl}(\{L_1, \dots, L_i\} \cap \mathcal{A}_n)| = \infty\}.$$

- 5: **Output any $w_n \in \text{Cl}(\mathcal{L}_n) \setminus (S_n \cup W_{n-1})$.**
-

Theorem B.2. Let $\mathcal{L}$ be a countable collection of languages. Then, [Algorithm 10](#) generates in the limit from $\mathcal{L}$ under $o(1)$ -noise and infinite omissions.

Proof. Recall that $\mathcal{L}^{(n)} := \{L_i : R(L_i; x_{1:n}) \leq c_i\}$. Since the adversary enumerates from K with vanishing noise rate, we know there exists a time T' such that for all $n \geq T'$, $K \in \mathcal{L}^{(n)}$ and $K \in \{L_1, \dots, L_n\}$, and thus $K \in \mathcal{A}_n$.

Suppose $K = L_{i^*}$. Now consider the set

$$B = \left\{ S \subseteq [i^*] : \left| \bigcap_{i \in S} L_i \right| < \infty \right\}.$$

For any $S \in B$, by [Proposition B.1](#), there exists T_S such that for $n \geq T_S$, $\{L_i : i \in S\} \not\subseteq \mathcal{L}^{(n)}$. Without loss of generality, we may assume $T_S \geq T'$ for any $S \in B$.

Let $T = \max_{S \in B} T_S$. For $n \geq T$, we must have $\{i : L_i \in \mathcal{L}^{(n)}\} \cap [i^*] \notin B$, since for $n \geq T$, $\{L_i : i \in S\} \not\subseteq \mathcal{L}^{(n)}$ for any $S \in B$. Therefore, for $n \geq \max\{T, i^*\}$, $K = L_{i^*} \in \mathcal{L}^{(n)}$, and the intersection of the prefix up to K , $\mathcal{L}^{(n)} \cap \{L_1, \dots, L_{i^*}\}$, has infinite number of elements. This means that the generator will generate $w_n \in \text{Cl}(\mathcal{L}_n)$ from the intersection of a prefix $\mathcal{L}_n$ of $\mathcal{A}_n \supseteq \mathcal{L}^{(n)} \cap \{L_1, \dots, L_{i^*}\}$ containing $K = L_{i^*}$, and thus generating in the limit. $\square$

B.2 Generating from Finite Collections with Noise

In [Theorem 5.4](#), we characterize the collections that can be generated at noise rate c . In this section, we ask the dual question. We consider a class of collections, namely all finite collections of size at most k , and ask what is the highest noise rate $c > 0$ under which all these collections can be generated in the limit. We prove the following result.

Theorem B.3. *For each $k \geq 2$, the following holds:*

1. *If $c \geq 1/k$, then there is a finite collection $\mathcal{L}$ of size k that not generable in the limit under c -noise.*
2. *If $c < 1/k$, then all finite collections $\mathcal{L}$ of size k can be generated in the limit under c -noise.*

Necessary Condition for Generation from Finite Collections

First, we prove the necessary condition.

Proof of [Theorem B.3](#) (Necessity). Let $\mathcal{X} = \mathbb{N}$ and, for $i \in \{0, 1, \dots, k-1\}$, define

$$L_{i+1} = \mathbb{N} \setminus \{n \in \mathbb{N} : n \equiv i \pmod{k}\}.$$

In words, L_{i+1} omits exactly the residue class $i \pmod{k}$.

Adversary. Consider the adversary's enumeration $x_t = t$ (the canonical ordering). Note that this is a valid ordering for all languages in $\mathcal{L}$ with the fraction of noisy examples being $1/k + o(1)$; therefore the noise density is $c = 1/k$ in the limit.

Impossibility of Generation. Fix any generator producing outputs $w_1, w_2, \dots$ with $w_t \notin \{x_1, \dots, x_t\}$ for all $t \geq t^*$ for some finite t^* . Among the k residue classes modulo k , one class $r \pmod{k}$ appears infinitely often in the output sequence $\{w_t\}_{t \geq 1}$ by the pigeonhole principle. Take the target language to be $K = L_{r+1}$ (which is possible due to the earlier observation that the adversaries stream is valid for languages in $\mathcal{L}$). Then, whenever $w_t \equiv r \pmod{k}$, we have $w_t \notin K$ by construction; hence the generator makes infinitely many mistakes and, thus, fails to generate in the limit from K . $\square$

Sufficient Condition for Generation from Finite Collections

Next, we prove the sufficiency part of [Theorem B.3](#).

Overview of the algorithm and proof. At a high level, our algorithm is as follows: at each time t , we compute a robust version space allowing up to n_t mistakes among the first t distinct examples, with n_t carefully chosen so that it grows linearly in t and slightly faster than the realized number of noisy examples. The latter condition ensures that the robust version space eventually contains the true language K and a more careful argument shows that all languages in the version space share infinitely many elements.

Before formalizing our result we begin with some preliminaries.

Preliminaries: Bounds on the Noisy Closure Dimension. Let $\mathcal{L} = \{L_1, \dots, L_k\}$ be a finite class (where each L_i infinite) and let $K \in \mathcal{L}$ be the target language. For a finite sequence $x_{1:d}$ and a noise-budget $n \geq 0$, define

$$V(x_{1:d}; n) = \{L \in \mathcal{L} : |\{x_{1:d}\} \cap L| \geq d - n\}, \quad \text{and} \quad \langle x_{1:d} \rangle_{\mathcal{L}, n} = \bigcap_{L \in V(x_{1:d}; n)} L$$

(with $\langle x_{1:d} \rangle_{\mathcal{L}, n} = \perp$ if $H(x_{1:d}; n) = \emptyset$). The n -Noisy Closure dimension $\text{NC}_n(\mathcal{L})$ is the largest d for which there exist distinct $x_{1:d}$ with $\langle x_{1:d} \rangle_{\mathcal{L}, n} \neq \perp$ but finite. For a finite size- k collection $\mathcal{L}$, [\[RR25\]](#) show that there is a class-dependent constant $d^* < \infty$ such that¹⁰

$$\text{NC}_n(\mathcal{L}) \leq nk + d^* + 1 \quad \text{for all } n \in \mathbb{N}. \quad (3)$$

(See their Corollary 3.4 in [\[RR25\]](#) and its proof for the construction of d^* .)

Our Result. Next, we present the main result of this section

Theorem B.4 (Finite classes tolerate density $\nu < 1/k$). *Fix any size- k collection $\mathcal{L} = \{L_1, \dots, L_k\}$ and $\nu < 1/k$. There exists a generator $\mathbb{G}$ (not knowing ν) that generates from K in the limit against every adversary whose noise rate is at most ν in the limit.*

Proof. Recall that we consider a density-bounded adversary that presents a *density-bounded noisy enumeration* of K , i.e., a stream of distinct examples $(x_t)_{t \geq 1}$ in which every $x \in K$ appears at some finite time and

$$\limsup_{d \rightarrow \infty} \frac{1}{d} |\{t \leq d : x_t \notin K\}| \leq \nu.$$

Let $S_t = \{x_1, \dots, x_t\}$ be the stream of examples presented in the first t steps and $d_t = |S_t|$ be the size of S_t . Define the dynamic noise budget

$$n_t = \max \left\{ 0, \left\lfloor \frac{d_t - (d^* + 2)}{k} \right\rfloor \right\}. \quad (4)$$

Generation $\mathbb{G}$. We claim that it is sufficient for $\mathbb{G}$ proceed as follows: At time t ,

¹⁰The constant d^* depends only on $\mathcal{L}$ (finite) and can be precomputed from it as in the proof of Corollary 3.4 of [\[RR25\]](#).

1. Compute n_t by (4) and the robust noisy-closure $C_t := \langle x_{1:t} \rangle_{\mathcal{L}, n_t}$.
2. If $C_t \neq \perp$, output any $w_t \in C_t \setminus S_t$. Otherwise, output any arbitrary placeholder string.

Correctness of $\mathbb{G}$. In the remainder of the proof, we prove $\mathbb{G}$'s correctness in several steps.

Claim B.5 (Sufficiency of Dynamic budget). *There is a finite $T < \infty$ such that for all $t \geq T$, $N_t \leq n_t$.*

Proof of Claim B.5. Let $N_t := |\{i \leq t : x_i \notin K\}|$ be the number of corruptions by time t . Due to the limsup condition, there exists an $\varepsilon \in (0, (1/k) - c)$ and t_0 large enough that for all $t \geq t_0$,

$$N_t \leq (c + \varepsilon) d_t. \quad (5)$$

On the other hand, by (4), for t with d_t sufficiently large we have

$$n_t \geq \frac{d_t - (d^* + 2)}{k} - 1 = \frac{d_t}{k} - \frac{d^* + 2 + k}{k}.$$

Combining with (5) and using $c + \varepsilon < \frac{1}{k}$, we obtain $N_t \leq n_t$ for all $t \geq T$ for some finite T . In particular, it is sufficient to select $T = t_0 + \frac{1}{\varepsilon k}(d^* + 2 + k)$. $\square$

Claim B.6 (Closure is Eventually Infinite). *For any $t \geq T$ (for T from Claim B.5), $|C_t| = \infty$ and $C_t \subseteq K$.*

Proof. Fix any $t \geq T$. Since $N_t \leq n_t$, the true language K belongs to $V(x_{1:t}; n_t)$ and, hence, $C_t = \langle x_{1:t} \rangle_{\mathcal{L}, n_t} \neq \perp$. Moreover, because $d_t > \text{NC}_{n_t}(\mathcal{L})$ by (3) and the definition of n_t , the noisy-closure C_t is either empty or infinite (by the definition of the noisy closure dimension); moreover, since the previous sentence rules out empty, C_t must be infinite. Finally, $C_t \subseteq K$ because $K \in V(x_{1:t}; n_t)$ and C_t is the intersection of supports over $V(x_{1:t}; n_t)$. $\square$

Now we are ready to conclude: By Claim B.6, for all $t \geq T$ the set $C_t \setminus S_t$ is nonempty and contained in K . Thus, G outputs $w_t \in K \setminus S_t$ for all $t \geq T$. $\square$

Remark B.7 (On the criticality of $c = 1/k$). The sufficiency proof reveals why the constant $1/k$ appears. The generator succeeds eventually when its noise budget n_t satisfies both

$$c < \frac{n_t}{d_t} \quad \text{and} \quad \text{NC}_{n_t} < d_t$$

for all $t \geq T$ for some finite T . The first condition ensures K enters the version space, while the second (with K in the version space) ensures the intersection of all version space languages is infinite. Since the noisy closure dimension $\text{NC}_n(\cdot)$ scales as $\Theta(nk)$ (see Equation (3)), the $1/k$ bound emerges from the above constraints:

- ▷ The first condition requires $n_t = \Omega(cd_t)$.
- ▷ The second condition, using $\text{NC}_{n_t} = \Theta(n_t k)$, requires $\Omega(cd_t k) \leq \text{NC}_{n_t} < d_t$.

Thus $cd_t k < d_t$, implying $c < 1/k$.

C Additional Preliminaries

In this section, we present the definitions of $\limsup$ and $\liminf$ that are used throughout the paper.

Definition 12. The limit inferior and limit superior of a sequence of reals $a = (a_n)_{n \in \mathbb{N}}$ are defined as:

$$\liminf_{n \rightarrow \infty} a_n := \sup_{N \in \mathbb{N}} \inf_{n \geq N} a_n \quad \text{and} \quad \limsup_{n \rightarrow \infty} a_n := \inf_{N \in \mathbb{N}} \sup_{n \geq N} a_n.$$

Note that, unlike $\lim_{n \rightarrow \infty} a_n$, limit inferior and limit superior exist for all sequences of reals a . We note that these limits are allowed to take values in the extended reals $\mathbb{R} \cup \{\pm\infty\}$.

Some simple facts are as follows, we refer the reader to [Rud76] for more extensive discussion.

Fact C.1. Given a sequence of real numbers $a = (a_n)_{n \in \mathbb{N}}$, the following always hold:

1. $\liminf a_n \leq \limsup a_n$
2. If $\liminf a_n = \limsup a_n$, the $\lim_{n \rightarrow \infty} a_n$ exists and $\liminf a_n = \lim_{n \rightarrow \infty} a_n = \limsup a_n$.

Next, we discuss some examples to build intuition.

Example C.2. The sequence $a_n = (-1)^n$ has $\liminf a_n = -1$ and $\limsup a_n = 1$

Example C.3. Consider the sequence $a = (a_n)_{n \in \mathbb{N}}$ where $a_n = 1$ if $n = 2^m$ for some $m \in \mathbb{N}$ and, otherwise, $a_n = 0$. It has $\liminf a_n = 0$ and $\limsup a_n = 1$.

Example C.4. The sequence $a_n = (-1)^n + 1/n$ has $\liminf a_n = -1$ and $\limsup a_n = 1$. In contrast, the related sequence $b_n = 1 + (-1)^n/n$ satisfies $\liminf b_n = \limsup b_n = 1$.