Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 82 (2026)

Pseudorandomness Beating the Hybrid Argument for Insensitive Algorithms

Dean Doron* Dana Moshkovitz† Justin Oh‡ David Zuckerman§

Abstract

The hardness vs. randomness paradigm converts a function $f\colon \{0,1\}^n \to \{0,1\}$ that is hard for circuits of size s into a pseudorandom generator (PRG) $G\colon \{0,1\}^d \to \{0,1\}^{s'}$ that fools circuits of size $s' = s'(s)$. In the application for derandomization, such as proofs of $\mathbf{BPP} = \mathbf{P}$, the overhead of using such a PRG directly depends on two quantities: the runtime of G and its seed length d. Ideally, the runtime of a PRG fooling circuits of size s' should only be slightly larger than s', and d should be $\log s'$. A central challenge in optimizing these parameters is optimizing the relationship between s and s'. Ideally, s' should only be slightly less than s.

Impagliazzo and Wigderson [IW97] constructed PRGs where s is a large polynomial in s' and $d = c \cdot \log s$ for a large constant $c \geq 8$. This gives derandomization with a large polynomial slowdown. Doron, Moshkovitz, Oh and Zuckerman [DMOZ22] constructed *near optimal* PRGs where $s' = s^{1-\alpha}$ and $d = (1+\alpha)\log s$ for arbitrarily small $\alpha > 0$. However, they require that f is hard against nondeterministic randomized circuits rather than deterministic circuits. Subsequent works of Chen, and Tell [CT21, CT24], later joined by Ball [BCT25], provide fast derandomization under various new assumptions. However, no result recovers the nearly optimal PRG of [DMOZ22] under the *standard* deterministic hardness assumption of [IW97]. In fact, from such an assumption, even constructing a nearly optimal hitting set or pseudoentropy generator against algorithms that err rarely is open. A key bottleneck in obtaining such a result is the use of the *hybrid argument*, which is known to require a large polynomial loss from s to s'.

We identify a property of randomized algorithms that we call "insensitivity", which allows us to bypass the hybrid argument. Insensitivity is a simple property we identify that many randomized algorithms satisfy: flipping a uniformly random bit in a randomness string on which the algorithm is correct is likely to yield yet another randomness string on which the algorithm is correct. We construct hitting sets and pseudoentropy generators for insensitive algorithms with better parameters than the hybrid argument can offer. Additionally, we consider a two-sided variant of insensitivity. In this variant, flipping a uniformly random bit in a randomness string for which the algorithm is incorrect either always yields another incorrect string, or yields a correct string with decent probability. For such algorithms we construct hitting sets and pseudoentropy generators with near optimal parameters.

*Faculty of Computer and Information Science, Ben-Gurion University. deand@bgu.ac.il. Supported in part by the Israel Science Foundation grant 857/25.

†Department of Computer Science, University of Texas at Austin. danama@cs.utexas.edu. Supported in part by NSF Grant CCF-1705028 and CCF-2200956.

‡Department of Computer Science, University of Haifa. sung-ho.oh@fulbrightmail.org. Supported in part by a Fulbright Postdoctoral Fellowship, ISF grant 1006/23, and by the European Union (ERC, ECCC, 101076663).

§Department of Computer Science, University of Texas at Austin. diz@cs.utexas.edu. Supported in part by NSF Grant CCF-2312573 and a Simons Investigator Award (#409864).

ISSN 1433-8092

Contents

1 Introduction

1.1 Pseudorandom Generators and Derandomization

The dominant approach to time-bounded derandomization (i.e. proving $\mathbf{BPP} = \mathbf{P}$) is via pseudorandom generators (PRGs) based on hardness assumptions. A PRG maps a short random seed Y into a large number m of bits that look like the uniform distribution on m bits, U_m, to any bounded distinguisher:

Definition 1.1 (indistinguishability)**.** *We say that $X \sim \{0,1\}^m$ is ε-indistinguishable from uniform to a class $\mathcal{D}$ of distinguishers, if for all $D \in \mathcal{D}$,*

$$|\mathbb{E}[D(X)] - \mathbb{E}[D(U_m)]| \leq \varepsilon.$$

Definition 1.2 (pseudorandom generator)**.** *We say that $G : \{0,1\}^d \to \{0,1\}^m$ is a* pseudorandom generator (PRG) *with error $\varepsilon > 0$ fooling a class $\mathcal{D}$ of distinguishers, if $G(U_d)$ is ε-indistinguishable from uniform to $\mathcal{D}$.*

A standard observation is that one can deterministically simulate any randomized algorithm $A(x, r)$ on input x that uses m bits of randomness r, by constructing a PRG outputting m bits for the class of *circuits* of size $\mathrm{time}(A)$. This is because such an algorithm (for fixed x) is equivalent to a circuit on m bits of size $\mathrm{time}(A)$.The simulation of A simply evaluates $A(x, G(\cdot))$ on all 2^d seeds of G, and takes the majority vote. The runtime of the simulation is then naturally determined by (1) the number of seeds 2^d; and (2) the runtime of the PRG, $\mathrm{time}(G)$. Specifically, the runtime is roughly $2^d(\mathrm{time}(G) + \mathrm{time}(A))$.[1] We say that the *overhead* of the simulation is the multiplicative slowdown: $2^d(\mathrm{time}(G) + \mathrm{time}(A))/\mathrm{time}(A)$. Thus, the overhead is roughly 2^d, provided $\mathrm{time}(G)$ is comparable to $\mathrm{time}(A)$. Since $m \leq \mathrm{time}(A)$, it is convenient to assume the worst case, that $m = \mathrm{time}(A)$. By the probabilistic method, an inefficient PRG for size-m circuits with seed length $(1+\alpha) \cdot \log m$ exists, so we can conceivably hope for an efficient PRG with that seed length that runs in time only slightly larger than m, such as $\mathrm{time}(G) = m^{1+\alpha}$, for small α. We call a PRG with these two features for small constant α *nearly-optimal*.[2]

Hardness vs. Randomness. Constructions of pseudorandom generators fooling circuits *imply* circuit lower bounds that are far out of reach of the field's current knowledge. A central line of inquiry in hardness vs. randomness aims to construct PRGs for circuits under *hardness assumptions*. Blum, Micali [BM84] and Yao [Yao82] were the first to prove that hardness implies pseudorandomness. Specifically, the generators of Blum, Micali, and Yao had large seed $d = m^{\delta}$, implying that randomized polynomial time algorithms could be simulated deterministically with sub-exponential overhead. Since those generators worked under cryptographic assumptions, they could not obtain $\mathbf{BPP} = \mathbf{P}$, because it is impossible for cryptographic PRGs to achieve the necessary seed length $O(\log m)$. Together with the fact that cryptographic assumptions did not seem necessary for derandomization, subsequent works aimed to prove stronger hardness vs. randomness theorems, with

[1] More accurately, the important value is the time it takes to evaluate G on all inputs. This can often be faster than $2^d \cdot \mathrm{time}(G)$. However, in informal discussions, this distinction is not important, and we often use "runtime" to refer to both values.

[2] An optimal PRG can have seed length $\log m + O(1)$, and we note that historically, the term optimal meant $O(\log m)$ seed length.

the goal of both decreasing the number of seeds and weakening the assumptions. In landmark results, Impagliazzo, Nisan, and Wigderson [NW94, IW97] constructed an appropriate PRG for polynomial overhead under the following plausible and necessary assumption.

Standard Assumption ([IW97]). *There exists a constant $0 < \beta < 1$ and a function $f \in \mathbf{DTIME}[2^{O(n)}]$ that is not computable by circuits of size $2^{\beta n}$.*

Indeed, under such an assumption, for any m, they construct a PRG $G\colon \{0,1\}^d \to \{0,1\}^m$ fooling circuits of size m such that $d = C \cdot \log m$ and $\mathrm{time}(G) = m^C$ for some large constant C. The most efficient construction from the standard assumption due to Umans [Uma03], building off of a result from Shaltiel and Umans [SU05], gives $C \approx 8$. *Nearly optimal* PRGs, where $d = 1.01 \cdot \log m$ and $\mathrm{time}(G) = m^{1.01}$, were out of reach.

1.2 Fine-Grained Derandomization and Nearly Optimal Pseudorandomness

Doron, Moshkovitz, Oh, and Zuckerman [DMOZ22] remedied this situation by obtaining near optimal seed length $d = (1+\alpha)\log m$ for arbitrarily small constant $\alpha > 0$ and $\mathrm{time}(G) = m^{1+\alpha}$. Such a PRG implies near-minimal overhead derandomization of a time-t randomized algorithm in time $t \cdot \max\{t, n\}^{1+O(\alpha)}$, which is (nearly) tight for sub-linear time algorithms, and under the non-deterministic strong exponential time hypothesis (NSETH), this overhead is also (nearly) tight for the univariate polynomial identity testing problem [Wil16].[3]

Alas, the construction of [DMOZ22] relies on a stronger hardness assumption, postulating the existence of a problem in $\mathbf{E}$ that requires *nondeterministic, randomized* circuits of nearly the same exponential size. Followup works continued to make a variety of strong hardness assumptions to achieve similar derandomization results. Specifically, Chen and Tell [CT21, CT24], and later with Ball [BCT25] reintroduce cryptographic assumptions.

Ideally, one would only require lower bounds against circuits that do not use nondeterminism, and would not make any cryptographic assumptions. In particular, we would like to revisit assumptions involving lower bounds against deterministic circuits, such as the standard hardness assumption of [IW97]. While the recent works discussed indeed achieve the goal of minimal overhead derandomization under plausible assumptions, the standard assumption is historically the most plausible and widely accepted. Furthermore, revisiting the question under the standard assumption is important for two other reasons:

- The main technical challenge in improving [IW97], avoiding the *hybrid argument*, is an extremely well-known bottleneck for many results in pseudorandomness, and has been observed by many researchers [BSW03, GSV18, FSUV13, DMOZ22, CT21, BCT25].
- The construction of [IW97] (and followups such as [STV01, SU05]) has had influential consequences *beyond* derandomization, such as in Trevisan's construction of seeded randomness extractors [Tre01], extractors for samplable distributions [OS25] and results in Kolmogorov complexity [Hir22], pseudo-deterministic algorithms [CLO+23, CHR24] and space-bounded algorithms [PRZ23, LPT24, DPTW25]. An improved construction and analysis from the standard assumption could open the possibility of yet even more exciting consequences in these areas.

[3] Chen and Tell [CT21] give derandomization in time $t \cdot n^{1+\alpha}$, coinciding with the result of [DMOZ22] for polynomial identity testing, and offering a faster derandomization when $t \gg n$.

We consider the following assumptions, also about deterministic circuit lower bounds.

Fine-Grained Assumption. *There exists a small constant $0 < \alpha < 1$ such that for every $n \in \mathbb{N}$, there exists a function $f \in \mathbf{DTIME}[2^{(1+\alpha)n}]$ that is not computable by circuits of size $2^{(1-\alpha)n}$.*

Batch Fine-Grained Assumption. *There exists a small constant $0 < \alpha < 1$ such that for every $n \in \mathbb{N}$, there exists a function $f\colon \{0,1\}^n \to \{0,1\}$ whose truth table is printable in deterministic time $2^{(1+\alpha)n}$ and f is not computable by circuits of size $2^{(1-\alpha)n}$.*

Notice that both the standard assumption and fine-grained assumptions are about *deterministic* circuits. The differences between the fine-grained assumptions and the standard assumption are twofold. First, instead of having f computable in time $2^{O(n)}$, we assume f is computable in time *nearly linear* in 2^n, or even that one can "batch" compute f on *all* inputs in this time. Second, we assume f is hard for circuits of size very close to 2^n. The formulation of the (non-batch) fine-grained assumption corresponds to the assumption of [DMOZ22] without nondeterminism or randomness. Adding the batch assumption allows us to more conveniently construct nearly optimal PRGs after we've addressed the core difficulty of avoiding the hybrid argument.

Despite the recent advances and techniques for constructing pseudorandom generators and related objects that suffice for derandomization, such as targeted PRGs [DMOZ22, CT21, CT24, BCT25], it seems far from reach to rely solely on the fine-grained assumption to obtain nearly optimal constructions. Indeed, previous works crucially rely on the use of stronger assumptions to address the difficulty of the hybrid argument. See Shaltiel and Viola [SV22] for a discussion of some existing works and why they require stronger assumptions.

For many of the same reasons, it even seems far from reach to get nearly optimal *hitting set generators* and *pseudoentropy generators* for circuits under the fine-grained assumption:

Definition 1.3 (hitting set generator)**.** *We say that $H\colon \{0,1\}^d \to \{0,1\}^m$ is a* hitting set generator *with error $\varepsilon > 0$ for a class $\mathcal{D}$ of distinguishers, if for every $D \in \mathcal{D}$ with $\Pr_{r\in\{0,1\}^m}[D(r) = 0] \geq \varepsilon$, there exists $y \in \{0,1\}^d$ such that $D(G(y)) = 0$.*

Hitting set generators are only guaranteed to intersect dense distinguishers. They suffice for derandomization of **RP** algorithms, and also of **BPP** algorithms, as was first proved by Andreev, Clementi, and Rolim [ACR98] (alternative proofs of the latter fact follow from ideas similar to those that place **BPP** in $\mathbf{\Sigma}_2$ [CH20, Gol22]).

Definition 1.4 (pseudoentropy and pseudoentropy generators)**.** *We say that $X = X_1, \ldots, X_m \sim \{0,1\}^m$ has* pseudoentropy *k with error ε for a class $\mathcal{D}$ of distinguishers if for every $D \in \mathcal{D}$ we have that: $\Pr[D(X) = 1] \leq \frac{|\{x:D(x)=1\}|}{2^k} + \varepsilon$.*

We say $G\colon \{0,1\}^d \to \{0,1\}^m$ is a pseudoentropy generator *with pseudoentropy k and error $\varepsilon > 0$ for a class $\mathcal{D}$ of distinguishers, if for a uniform Y, $G(Y)$ has pseudoentropy k with error ε for $\mathcal{D}$.*

While there are several notions of pseudoentropy,[4] our notion suffices to derandomize algorithms that err extremely rarely (namely when the error probability is smaller than 2^{k-m} [DMOZ22]). Once again, we say such generators are nearly optimal if, for the class of circuits of size m, they have $d \approx (1+\alpha) \cdot \log m$ and runtime $m^{1+\alpha}$.

[4] This notion is equivalent to *metric pseudoentropy* when $\mathcal{D}$ is closed under complement [BSW03].

1.3 Insensitivity: A Structure of Randomness in Algorithms

We identify interesting properties about the structure of randomness in algorithms, and focus on constructing near optimal hitting set and pseudoentropy generators for circuits corresponding to such properties. We believe that many randomized algorithms induce some local structure on their randomness strings. In particular, a motivating question for us is if a randomness string r is "good," how many neighbors of r are also "good?" A neighbor of a randomness string $r \in \{0,1\}^m$ is a string of Hamming distance 1 from r. Alternatively, we consider r as a vector in $\mathbb{F}_2$ and denote a neighbor as $r + e_i$ for the standard basis vector e_i and for some i. Formally, we introduce and study the notion of *insensitive* algorithms.

Definition 1.5 (insensitivity)**.** *We say that a circuit $D\colon \{0,1\}^m \to \{0,1\}$ is β-*insensitive *if for every x such that $D(x) = 0$, it holds that $\Pr_i[D(x + e_i) = 1] \leq \beta$.*

We say that a randomized algorithm A is β-insensitive if for every input length n and input $x \in \{0,1\}^n$ to the algorithm, the circuit $D_x = A(x, \cdot)$ is β-sensitive. Note that here, $D_x(r) = 0$, for $r \in \{0,1\}^{m=m(n)}$, means that the algorithm is correct*.*

In an insensitive algorithm, nearly all neighbors of a good randomness string are also good, since we think of β as small. For the correspondence between algorithms and circuits, think of D as a circuit for A that fixes input x and assume without loss of generality that the algorithm errs whenever $D(x) = 1$.

In the following theorem and throughout our paper, we use .01 as shorthand for α for arbitrarily small α. Therefore, 1.01 means $1 + \alpha$, .99 means $1 - \alpha$, etc.

Theorem 1 (pseudorandomness beating the hybrid argument for insensitive algorithms, informal; see Corollary 9.4, Corollary 9.8)**.** *Assume the batch fine-grained assumption, and let f be the corresponding function. Then, for any m, and $k \leq m^{0.99}$, there is an explicit function*

$$G^f\colon \{0,1\}^{1.01(\log m + \log k)} \to \{0,1\}^m$$

that is a hitting set generator with error $1 - 2^{k-m}$, and a pseudoentropy generator with entropy k and error $m^{-0.01}$ for β-insensitive circuits with circuit size m, whenever $\beta \leq k^{-1.01}$. G^f is computable in time nearly linear in $m \cdot k$.

To the best of our knowledge, no previous construction from a deterministic circuit lower bound achieves such a short seed. As we discuss later, prior constructions rely on a hybrid argument, which imposes a larger polynomial relationship between the hardness s of f, namely $s \geq m^3$. Consequently, these constructions require seed length at least $3 \cdot \log m$ [GSV18, SV22], and in fact, the constant is significantly larger for other reasons as well. This generator allows us to more efficiently derandomize algorithms that err rarely, on at most $2^{m^{0.99}}$ randomness strings (see Theorem 9.10). We note that, for algorithms that err rarely, a small modification to the hybrid argument yields generators from standard constructions with runtime $m \cdot \mathrm{poly}(k)$ (see Section 5 for details).

An Insensitive Algorithm. To show that our notion of insensitive algorithm is natural, we give an example. In the color coding randomized algorithm [AYZ95], the input is a directed graph $G = (V, E)$ and a number $\ell \geq 1$. The goal is to decide whether there is a simple path of length ℓ in G. For $\ell = |V|$ this is the **NP**-hard Hamiltonian path problem. The randomized algorithm works

for $\ell \leq \log |V| / \log\log |V|$. The algorithm colors each vertex $v \in V$ with a uniformly random color $f(v) \in \{1, \ldots, \ell\}$. The algorithm then removes all the edges except those edges $(u, v) \in E$ where $f(v) = f(u) + 1$. The new graph is a directed acyclic graph, for which it is possible to find the longest simple path in linear time using dynamic programming. If there is a path of length at least ℓ in G, then a random coloring keeps the path with probability at least $\ell^{-\ell}$. Hence, by repeating the coloring algorithm $\Theta(\ell^\ell \log(1/\delta))$ times, a randomized algorithm finds a path of length at least ℓ, if one exists, with probability at least $1 - \delta$.

The color coding algorithm is insensitive: colorings that contain length-ℓ paths are likely unperturbed by a small random change. That is, for *every* coloring that keeps a length ℓ path, if we change the color of one uniformly random vertex, there is probability $1 - \frac{\ell}{n}$ that the coloring still identifies a length ℓ path. In other words, the algorithm is $(\beta = \frac{\ell}{n})$-insensitive.

As a disclaimer, we use color coding to demonstrate the notion of an insensitive algorithm. Recall that pseudoentropy generators that output entropy $k \leq m^{.99}$ only derandomize algorithms whose error probability is exponentially small in the randomness. Therefore, Theorem 1 doesn't apply to color coding.

Scoring Algorithms. We further introduce a subclass of insensitive algorithms for which we can construct near-optimal pseudoentropy generators. We call these algorithms *scorers* because they assign a fine-grained score to each randomness string. A randomness string that scores high enough leads to a correct outcome. Scores are unlikely to drop with a random flip. Finally, when a randomness string scores low, the score is pretty likely to improve with a random flip. Formally:

Definition 1.6 (scorer)**.** *Let $\alpha, \beta \in [0, 1]$. We say that randomized algorithm is an (α, β)-*scorer*, if per input x to the algorithm, there is a scoring function $\phi_x : \{0,1\}^m \to [0, 1]$, such that the "good" randomness strings $r \in \{0,1\}^m$ for the algorithm on input x are those with $\phi_x(r) \geq \tau$ for some $\tau \in (0, 1)$. Moreover, ϕ_x satisfies the following properties:*

- Efficiency: *For every x, on input $r \in \{0,1\}^m$ the score $\phi_x(r)$ can be computed by an efficient* **BPP** *algorithm.*
- Approximate Monotonicity: *For every x, for every randomness r, $\Pr_{i \sim [m]}[\phi_x(r + e_i) < \phi_x(r)] \leq \beta$.*
- Improvement: *For every x, for every randomness r with $\phi_x(r) < \tau$, $\Pr_{i \sim [m]}[\phi_x(r + e_i) > \phi_x(r)] \geq \alpha$.*

Scorers abstract out algorithms like the randomized minimum spanning tree algorithm of Karger, Klein, and Tarjan [KKT95]. This famous **ZPP** algorithm runs in linear time, except with exponentially small probability. In Section 10, we demonstrate that this algorithm for minimum spanning tree is a scorer (under a slight variant of the definition). Note that approximate monotonicity implies that the algorithm is insensitive.

We can now state our theorem.

Theorem 2 (nearly optimal pseudorandomness for scorers; informal, see Theorem 10.2)**.** *Assume the batch fine-grained assumption and let f be the corresponding function. Then, for any m, there is an explicit function*

$$G^f : \{0,1\}^{1.01 \log m} \to \{0,1\}^m$$

such that for (α, β)-scorers with circuit size m, whenever $\alpha\varepsilon - \beta > m^{-0.001}$ we have for all x,

$$\Pr[\phi_x(G(Y)) \leq \tau] \leq \varepsilon.$$

Namely, G^f is a hitting set generator and a pseudoentropy generator for $D = \{r : \phi(r) > \tau\}$. Moreover, G is computable in time $m^{1.01}$.

In particular, when $\alpha \geq m^{-0.01}$ and $\beta \leq m^{-0.03}$, we can have $\varepsilon = m^{-0.01}$. Additionally, when α and β are constants, ε is a constant.

Two-Sided Variant of Insensitivity. Scorers are a very specific subclass of randomized algorithms. In general, decision problems in **BPP** may not come equipped with a notion of a score for the quality of the randomness string that satisfies the above conditions.

To generalize to such algorithms, we introduce a two-sided variant of insensitivity, though this is not exactly what one might expect from the phrase "two-sided insensitivity." Our variant of a two-sided insensitive algorithm has the additional property that any bad randomness string with at least one good neighbor actually has many good neighbors.

Definition 1.7 (two-sided insensitivity)**.** *We say that a circuit $D\colon \{0,1\}^m \to \{0,1\}$ is (α,β)-*insensitive *if for every x such that $D(x) = 0$, it holds that $\Pr_i[D(x+e_i) = 1] \leq \beta$, and for every x such that $D(x) = 1$, either $\Pr_i[D(x+e_i) = 0] = 0$ or $\Pr_i[D(x+e_i) = 0] \geq \alpha$.*

We say that a randomized algorithm A is (α,β)-insensitive if for every input length n and input $x \in \{0,1\}^n$ to the algorithm, the circuit $D_x = A(x,\cdot)$ is (α,β)-insensitive. Again, we use $D_x(r) = 0$ for correct answers.

Indeed, both the approximate monotonicity and improvement properties closely resemble the two properties of two-sided insensitivity. Given the stronger property of two-sided insensitivity, we can achieve nearly optimal pseudorandomness.

Theorem 3 (nearly optimal pseudorandomness for two-sided insensitive algorithms, informal; see Corollary 9.2, Corollary 9.6)**.** *Assume the batch fine-grained assumption, and let f be the corresponding function. Then, for any m, there is an explicit function*

$$G^f\colon \{0,1\}^{1.01\log m} \to \{0,1\}^m$$

that is a hitting set generator with error $1 - 2^{k-m}$ and a pseudoentropy generator with entropy k with error ε for (α,β)-insensitive circuits with circuit size m, whenever $\frac{\alpha}{2}\cdot\varepsilon - \beta\log m - 2^{k-m} > m^{-0.001}$. In particular:

- *G^f is a hitting set generator with error $1 - m^{-0.01}$, and a pseudoentropy generator with entropy $m - O(\log m)$ and error $m^{-0.01}$ for (α,β)-insensitive distinguishers of circuit size m, when $\alpha \geq m^{-0.01}$ and $\beta \leq m^{-0.02}$.*
- *For a constant ε, G^f is a hitting set generator with error $1-\varepsilon$, and a pseudoentropy generator with entropy $m - O(1)$ and error ε as long as α is any constant, and $\beta \leq \frac{\alpha\varepsilon}{\log m}$.*

Notice that for two-sided insensitive algorithms, the pseudoentropy is extremely high. When we consider constant α and $\beta \leq 1/\log m$, it's only a constant bits shy from "full entropy." When we think of $\alpha \geq m^{-0.01}$ and $\beta \leq m^{-0.02}$, we are only $\log m$ bits shy. This implies we can efficiently derandomize algorithms that err with constant probability or probability $m^{-\theta}$ (see Theorem 9.10).

Finally, note that Theorem 3 beats Theorem 1 in that it gives nearly-optimal seed length, though for more restricted algorithms. As we will discuss later in Section 1.7, we identify yet another barrier to obtaining the parameters of Theorem 3 for general circuits.

Our Approach. To achieve our results we provide an approach to the longstanding general question of *beating the hybrid argument*, by connecting it to the longstanding open problem in pseudorandomness of *weak unpredictability ⇒ pseudoentropy*, as discussed by Barak, Shaltiel, and Wigderson [BSW03]. In the next sub-sections we first identify the key parameter that the optimality of all hardness vs. randomness constructions hinges on: the "hardness loss". We then discuss the inherent significance of the hybrid argument in affecting this parameter. Then, we are finally ready to put forth our approach to both beating the hybrid argument and addressing the question of weak unpredictability ⇒ pseudoentropy. We identify barriers to this approach and demonstrate how structural assumptions on the distinguishers can circumvent the barriers.

1.4 The Hardness Loss and the Hybrid Argument

Typical hardness vs. randomness results transform a function $f\colon \{0,1\}^n \to \{0,1\}$ that is hard for circuits of size $s = s(n) < 2^n$ into a PRG $G^f\colon \{0,1\}^d \to \{0,1\}^m$ that fools circuits of size $m = m(n) \leq s(n)$. An important consideration when evaluating the quality of the construction is the relationship between s and m. We generally refer to the ratio between s and m as the *hardness loss* of the construction. To see why the hardness loss is a central parameter, consider the fine-grained assumption, where $s = 2^{(1-\alpha)n}$. If the construction of G requires $s = m^C$ for a large constant C, then both the seed length and runtime of the PRG will be suboptimal. This is because constructing G^f generally involves computing f on at least one input, which requires time at least $2^n \geq s = m^C$, and choosing a random coordinate of (an encoding of) f, requires seed of length at least $\log(2^n) \geq \log s = C \cdot \log m$. Therefore, if we wish to follow this framework in order to obtain nearly optimal constructions, it is important to minimize the exponent C in order to minimize the hardness loss. Indeed, at a high level, the question of providing constructions with m as close to s as possible is already interesting in its own right: it asks whether one can convert nearly all the hardness from f into pseudorandomness.

Unpredictability and the Loss of the Hybrid Argument. So far, we've explained why the hardness loss is a crucial parameter to optimize, but have not yet seen why this parameter is suboptimal in previous constructions. This is because all such constructions go through the *hybrid argument* which we explain now. For m bits to be indistinguishable from uniform, it must be that each bit is unpredictable given the previous bits, and the prediction errors add up across the m bits.

Definition 1.8 ((standard) unpredictability)**.** *We say that a sequence of m random bits $X = X_1, \ldots, X_m$ is ε-unpredictable for circuits of size s if for every $i \in [m]$ and every predicting circuit $P\colon \{0,1\}^i \to \{0,1\}$ of size s, it holds that*

$$\Pr_X[P(X_1, \ldots, X_{i-1}) = X_i] \leq \frac{1}{2} + \varepsilon.$$

We say that $G\colon \{0,1\}^d \to \{0,1\}^m$ is an ε-unpredictability generator if $G(U_d)$ is ε-unpredictable.

Suppose we wish to show that a distribution $X = X_1, \ldots, X_m$ on $\{0,1\}^m$ is ε-indistinguishable from uniform for circuits of size m via the hybrid argument. Proceeding with Yao's next bit predictability argument, we must show that for every $i \in [m]$, no circuit $C\colon \{0,1\}^{i-1} \to \{0,1\}$ of size roughly m can predict X_i with probability greater than $\frac{1}{2} + \frac{\varepsilon}{m}$ when fed $x_1, \ldots, x_{i-1} \sim X_1, \ldots, X_{i-1}$. That is, we must show that X is ε/m unpredictable.

Nearly every pseudorandom generator from the standard assumption indeed constructs an (ε/m)-unpredictability generator, and suffers a large hardness loss in the process. In fact, [GSV18] showed that the standard unpredictability-generator-based approach to constructing a pseudorandom generator G^f for size-m circuits from a function f of hardness s inherently requires $s \geq m^3$.

1.5 Beating the Hybrid Argument, Weak Unpredictability, and Pseudoentropy

Suppose we want to construct unpredictability generators with $\varepsilon = O(1)$ or even $m^{-\theta}$ for a small constant θ, rather than $\varepsilon = O(1/m)$ as required by the hybrid argument. Then we can already do so using existing constructions of ε-unpredictability generators (used in PRG constructions). Moreover, such constructions already have *minimal hardness loss*. That is, given f that is hard for circuits of size s, $G^f\colon \{0,1\}^d \to \{0,1\}^m$ is unpredictable for circuits of size $m \approx s$. In fact, the output distribution of these generators even satisfy a stronger definition of *symmetric* unpredictability, where for every index i, X_i is unpredictable even when given all other bits $X_{-i} = X_1, \ldots, X_{i-1}, X_{i+1}, X_m$:

Definition 1.9 (symmetric unpredictability)**.** *We say that a sequence of m random bits $X = X_1, \ldots, X_m$ is ε-symmetrically unpredictable for circuits of size s if for every $i \in [m]$ and every predicting circuit $P\colon \{0,1\}^{n-1} \to \{0,1\}$ of size s, it holds that*

$$\Pr_X[P(X_{-i}) = X_i] \leq \frac{1}{2} + \varepsilon.$$

We say that $G\colon \{0,1\}^d \to \{0,1\}^m$ is an ε-symmetric unpredictability generator for circuits of size s if $G(U_d)$ is ε-symmetrically unpredictable for circuits of size s.

Indeed, by composing two different instantiations of the Nisan-Wigderson generator (applied on top of a locally decodable code based construction as in [STV01]) with two different settings of parameters, we obtain the following unpredictability generator.

Lemma 1.10 (informal; see Theorem 8.9)**.** *Assume the batch fine-grained assumption. Then, for any small constant θ, there is an explicit $(\varepsilon = m^{-\theta})$-symmetric unpredictability generator $G\colon \{0,1\}^{d=(1+\theta)\log m} \to \{0,1\}^m$ for circuits of size m, that is computable in time $m^{1+O(\theta)}$.*

Notice the unpredictability generator is nearly optimal in the sense that the seed length is nearly $1 \cdot \log m$ and the runtime is close to the size of the circuits it is designed against. What useful properties could the distribution of such a generator have? Intuitively, we expect that such a distribution *already* has pseudoentropy for circuits of size close to m, and thus directly serves as a desired construction of a near-optimal pseudoentropy generator or hitting set generator.

The Problem of Weak Unpredictability to Pseudoentropy. If X is *information theoretically* ε-unpredictable (i.e., where the prediction algorithms is computationally unbounded), even for $\varepsilon \gg 1/m$, one can easily show that the entropy of X is $\Omega(n)$ (this is immediate from the chain rule). Thus, it is reasonable to expect that in the computational setting, X also has some useful notion of pseudoentropy. However, it turns out that this is generally open. In particular, for our purposes we formulate a guiding question of whether unpredictability implies pseudoentropy *with minimal hardness loss*:

Question 1.11. *Does a distribution $X = X_1, \ldots, X_m$ with $\varepsilon = \Omega(1)$-unpredictability for circuits of size m imply that $X_1, \ldots, X_m$ have pseudoentropy $k = \Omega(m)$ against circuits of size $\approx m$ (say, $\Omega(m^{1-o(1)})$)?*

Our main results show that while we do not answer this question exactly, for the families of circuits discussed in Section 1.3, we can conclude pseudoentropy with minimal (or better than previously known) hardness loss and large k. We provide a *new framework* for converting a distinguisher of size s to a predictor for X of size close to s in this setting. Near optimal pseudorandomness constructions for insensitive algorithms with sufficiently small error probability then follow.

1.6 Our Distinguisher-to-Predictor Approach

Informally, for each definition of pseudorandom, hitting set, and pseudoentropy generator, we say that a function D distinguishes too well if it violates the desired property. When it is clear from context, we may simply say "D is a distinguisher." Information theoretically, if $X = X_1, \ldots, X_m$ is far from uniform, then it's easy to conclude that at least one bit of X_i is predictable from the rest. Computationally, if some circuit D distinguishes X from uniform too well, then the hybrid argument is the standard tool that converts this distinguisher into an efficient predictor for some bit X_i.

Now, shifting the spotlight from the uniform distribution, if we instead know that $X = X_1, \ldots, X_m$ has low entropy (information theoretically), then intuitively we can conclude that *many* bits X_i are predictable. Once again, to translate this to the computational setting, we hope to argue that if there is a distinguisher D that works too well (according to our notion of pseudoentropy), then there are many i-s for which an efficient predictor exists. However, unlike the uniform case, where there is a standard way to go from distinguishing to predicting, there is no longer a known framework to achieve what we want. We work with the following natural definition, where a predictor is a randomized algorithm that is given as input i, X_{-i}, and attempts to predict X_i.

Definition 1.12 ((many-bit) predictor)**.** *An ε-predictor for $X = X_1, \ldots, X_n$ is a randomized circuit $P\colon [m] \times \{0,1\}^{m-1} \times \{0,1\}^{|R|} \to \{0,1\}$ that gets as input uniform i, X_{-i} and internal randomness R, such that,*

$$\Pr_{i,X,R}[P(i, X_{-i}, R) = X_i] > \frac{1}{2} + \varepsilon.$$

We often call ε the advantage of the predictor.

The key difference from a standard predictor is that i is given as input, and the probability of success is also quantified over i. Notice that given an ε-predictor, a simple averaging argument shows that there are many i-s that are predictable. Thus constructing such a predictor from a distinguisher for pseudoentropy (i.e., a circuit that violates Definition 1.4) demonstrates what should be true intuitively. Thus, if many i-s are predictable ($\gg 1/m$ fraction), then there exists a fixing to i and R such that X_i is predictable by P with the same probability (over X), contradicting Definition 1.9.

To understand at a high level how one utilizes a distinguisher to construct a predictor, we consider the easier case of proving that X is a hitting set generator. For notational convenience, we think of X as a vector in $\mathbb{F}_2$, and $X + e_i$ is then X with the i-th bit flipped. We suppose otherwise, that for every $x \in \mathrm{Supp}(X)$, $D(x) = 1$. The usual intuition is that we can predict coordinate i if $D(X)$ and $D(X + e_i)$ differ. Thus, we can predict X_i from X_{-i} by running D twice: once on X_{-i} with 0 in the i-th coordinate and also with 1 in the i-th coordinate. Whichever call to D outputs 1 tells us what bit to output.

Now suppose also that for every $x \in \mathrm{Supp}(X)$, for at least ε fraction of i-s, flipping the i-th bit of x causes it to leave D: $D(x + e_i) = 0$. Then in fact, the predictor described above succeeds with high probability over i and we are done. As a teaser, while we can't guarantee such a strong property, we will give a randomized procedure that "moves" x to an x' that has this property.

1.7 Pseudoentropy for Insensitive Algorithms

The main result of our work is that we can convert a distinguisher to a predictor even with large $\varepsilon \gg 1/m$, when the distinguisher circuit is *insensitive*. This then provides pseudorandomness against insensitive algorithms.

Theorem 1 follows from the previously discussed construction of the weak unpredictability generator in Lemma 1.10, and a new method of efficiently converting insensitive distinguishers into (many-bit) predictors with large success probability.

Theorem 1.13 (insensitive distinguisher to predictor, informal; see Theorem 7.10)**.** *For any m and $k \leq m^{0.99}$, let $D\colon \{0,1\}^m \to \{0,1\}$ be a ($\beta = k^{-1.01}$)-insensitive circuit of size m. Further suppose that $|D| = |\{x\colon D(x) = 1\}| \leq 2^k$. Then, there is a randomized predictor $P\colon [i] \times \{0,1\}^{m-1} \times \{0,1\}^{|R|} \to \{0,1\}$ of size $\approx m \cdot k$ such that for any $x \in D$,*

$$\Pr_{i,R}[P(i, x_{-i}, R) = x_i] \geq \frac{1}{2} + \frac{1}{16}.$$

This distinguisher to predictor theorem establishes that unpredictability generators are pseudoentropy and hitting set generators, following the intuition presented in Section 1.5.

Remark 1.14. *As will be apparent later on, our predictors satisfy a stronger notion of prediction, wherein P is allowed to output "I don't know". Whenever P* does *output an answer, which happens with probability roughly $1/16$, the prediction is almost certainly correct. We do not use this property in our results.*

Moreover, note that our predictor has an "instance-wise" guarantee, and succeeds with noticeable probability regardless of the fixed input x.

We do not know whether the additional property of insensitivity is necessary for beating the hybrid argument. However, we show that using a distinguisher to construct a predictor for a large fraction of coordinates i in a "black-box" manner inherently requires a linear loss in hardness.

Theorem 1.15 (barrier for distinguisher to many-bit predictor transformations for large ε, informal; see Theorem 4.3)**.** *Any (many-bit) predictor obtained in a "black-box" manner from a distinguishing circuit $D\colon \{0,1\}^m \to \{0,1\}$ of size m with $|D| = |\{x\colon D(x) = 1\}| = 2^k$ that succeeds on more than ε-fraction of coordinates i, requires size at least εmk.*

This lower bound also applies to the standard distinguisher to predictor transformation obtained via hybrid argument, which succeeds for $\varepsilon = 1/m$. In this case the lower bound states that the size of the predictor can be m, the same size as the original distinguisher. The standard predictor from the hybrid argument indeed matches this. Note however, that in standard constructions, while the loss in size from distinguisher to predictor is minimal, the loss in constructing the unpredictability generator itself (to error $1/m$) is large.

For large ε, such as constant (for which unpredictability generators with minimal hardness loss exist), the lower bound requires a factor k loss in size. Notice that matching this lower bound still

beats the hybrid argument, and that Theorem 1 and Theorem 1.13 do so for distinguishers that are also insensitive. However, the lower bound suggests that obtaining nearly optimal pseudorandomness (rather than that only beating the hybrid argument) requires additional structure on D. Our stronger notion of "two-sided" insensitivity circumvents this barrier. The result follows from an even more efficient distinguisher to predictor transformation, given the extra structure of D.

Theorem 1.16 (two-sided insensitive distinguisher to predictor, informal; see Theorem 6.8)**.** *Let $D\colon \{0,1\}^m \to \{0,1\}$ be a (α,β)-insensitive circuit of size m with $|D| \leq 2^k$. Further suppose that $\alpha - \beta \log m - 2^{k-m} > m^{-0.01}$. Then, there is a randomized predictor $P\colon [i]\times\{0,1\}^{m-1}\times\{0,1\}^{|R|} \to \{0,1\}$ of size $\approx m \cdot \log m$ such that for any $x \in D$,*

$$\Pr_{i,R}[P(i, x_{-i}, R) = x_i] > \frac{1}{2} + m^{-0.01}.$$

Finally, we also provide a distinguisher to predictor transformation for the algorithms are scorers. This transformation is the simplest and most efficient, give large prediction advantage with size *linear* in m. The treatment of the case of scorers appears in Section 10.

1.8 Prior Work on Unpredictability and Pseudoentropy

Our work is not the first to pose or study the question of whether unpredictability implies pseudoentropy. However, to the best of our knowledge, it is the first to consider the question for *deterministic* circuits with *minimal hardness loss*.

The most important work to address in this context is a PRG due to Sudan, Trevisan, and Vadhan [STV01]. Indeed, this generator is essentially the same unpredictability generator that we use. However, they argue that the resulting distribution X is computationally indistinguishable from some high entropy source Z. By doing so they once again require the use of the hybrid argument: they prove that X and Z are indistinguishable by changing X coordinate by coordinate into Z, and bounding the error at each step. This process then incurs a hardness loss: although X was unpredictable for circuits of size s, it is only indistinguishable from Z by much smaller circuits. In comparison with our approach, they prove a much stronger notion of pseudoentropy, and thus must pay a large hardness loss. Interestingly, the approach here circumvents the barrier presented by [GSV18], which implies a large polynomial loss in hardness as discussed in Section 1.4. However, their technique meets yet another barrier. They require applying Impagliazzo's Hardcore Lemma with error $1/m$ which also has an inherent large polynomial loss [BHKT24].

A result of [BSW03] shows that when X is unpredictable for circuits equipped with oracle gates to the polynomial hierarchy, then X has pseudoentropy for the same types of circuits. Interestingly, although they do not provide a result for deterministic circuits, their result does have minimal hardness loss.

The unpredictability to pseudoentropy problem has also been studied in the related but different cryptographic setting (see, e.g., [HLR07, SGP15, KPWW16, KPWW16, HMS23]). They typically consider a stronger notion where X is indistinguishable from some high entropy distribution for *all* polynomial size distinguishers. Naturally, their transformations also lose polynomial factors in the circuit size.

We dedicate the next section to a technical overview for proving Theorem 1, and Theorem 3 and our distinguisher to predictor transformations.

2 Technical Overview

Our main technical contributions are the efficient distinguisher-to-predictor transformations for the one-sided and two-sided insensitive distinguishers of Theorem 1.13 and Theorem 1.16. For simplicity, we consider the case of transforming a distinguisher D for a hitting set X into a predictor. The analysis for pseudoentropy is nearly identical, with only a small amount of additional work necessary. Thus we assume that for every $x \in \mathrm{Supp}(X)$, we have that $x \in D$ (we use this as shorthand for $D(x) = 1$, and say that "x is inside/outside D"). We wish to construct a predictor that succeeds with probability $1/2 + \varepsilon$ over i, X, and its own internal randomness R, for $\varepsilon \gg 1/m$. As mentioned earlier, we'll in fact prove something stronger, that there is a predictor that succeeds for large ε given *any fixed* $x \in D$. Namely, for any $x \in D$, we consider picking a random i, removing bit x_i, and feeding the predictor x_{-i}. The predictor should succeed for many choices of i in computing x_i, or in other words, recovering the original string x.

The Basic Idea. A natural way the predictor P can perform such a task is to consider the following two strings: x_{-i} with 0 in the i-th coordinate and 1 in the i-th coordinate. Although P does not know which one is the original x, it can correctly predict if exactly one of the two strings is in D: by assumption, the one in D is the original x (the predictor can output a random guess if there is no such discrepancy).

For any $x \in D$ we say that x is sensitive if at least α-fraction of its neighbors $x + e_i$ are outside D. Our first observation is that the predictor succeeds with advantage α immediately if *every* $x \in D$ is sensitive. We do not expect such a strong property to hold. Our core approach is to argue instead that a random walk starting at x on the Boolean hypercube $\{0,1\}^m$ reaches a sensitive point.

To develop intuition for our predictor, consider the case that D is a union of disjoint balls of radius r around the points in the range of G, for some small $r < m$. Our predictor takes two coupled random walks, one starting at $v = x_{-i}$ with 0 in the i-th coordinate and $w = x_{-i}$ with 1 in the i-th coordinate. This is equivalent to one starting at x and the other starting at $x + e_i$. Let the coupled walks be $v = v^0, v^1, \ldots, v^t$ and $w = w^0, \ldots, w^t$, where $w^j = v^j + e_i$. For the first j such that there is a discrepancy, $D(v^j) \neq D(w^j)$, the predictor outputs 0 if $D(v^j) = 1$ and outputs 1 otherwise. This is because we expect the walk starting from $x + e_i$ to be further and hence more likely that $D(x + e_i) = 0$.

To see why this predictor succeeds with large probability over input i, for the sake of analysis, imagine all coupled walks for every i running simultaneously. Specifically, there are $m + 1$ walks, all coupled together: there is the walk starting at x, and the "shifted" walks starting at $x + e_i$ for every $i \in [m]$. In effect, there is a ball of radius 1 centered at x, and this entire ball moves according to a random walk in $\{0,1\}^m$. Call this "neighborhood" ball N_x. The core intuition is that when this ball is at a step in its walk where the center is in D, and many shifted walks are outside, then there are many corresponding choices of i for which the predictor is correct.

We first note that the random walk starting at x, the center of N_x, converges to uniform in $\{0,1\}^m$. Thus, since D is small, the walk starting at x exits D. In particular, at some step, the walk starting at x is at distance exactly r from x: this is the step just before exiting. In this case, at least $m - r$ points in the ball N_x are outside D, while the center of the ball, which started at x, is inside. This means that for at least $m - r$ choices of i, our predictor notices a discrepancy at this step and terminates with a correct prediction.

Predicting with Two-Sided Insensitivity. The analysis of this predictor when D is (α, β)-two-sided insensitive, rather than specifically a union of disjoint balls, is similar. When the center of N_x exits, it exits at a sensitive node, by definition. At the step just before it exits, at least α fraction of the shifted walks are outside D, and so for the corresponding i-s the predictor is correct.

One caveat, however, is that for the other $(1-\alpha)$ fractions of i-s, the predictor does not see a discrepancy at this step, and the center of N_x exits D in the next step. This can be a problem: for some choices of i, the predictor may now see a discrepancy, and since now the center of N_x is outside of D, it will terminate with an *incorrect* prediction. However, by insensitivity, this can only happen with probability β over i. See Section 6.1 for details.

So far, the key feature of the random walk we've used is that it exits D with high probability. In particular, this is satisfied by the m-step random walk that, at each step $j \in [m]$, flips coordinate j with probability $1/2$. As is, the predictor makes $O(m)$ calls to D, two for each step of the random walk, and thus has size roughly m^2. To obtain Theorem 1.16 for two-sided insensitivity, we utilize binary search to reduce the number of calls to D from $O(m)$ to $O(\log m)$.

Our predictor succeeds by finding a step j when the random walk leaves D. Although our initial basic idea was to find such a step by examining every step of the walk, we can do so instead via a binary search. Since $x \in D$ and, with high probability, the end of the walk is outside D, the predictor (on input i) can query the midpoint of the walk by computing $D(v^{m/2})$ and $D(w^{m/2})$. Recall that the predictor does not know which call is from the walk starting at the original x. However, if the calls output the same value, we can shift the window of the binary search: for example, if both calls output 1, then we know that j is between $m/2$ and m. Furthermore, if the calls output different values, then it terminates with a prediction. The only caveat to this argument is that the algorithm may query several steps for which the center of N_x is outside D. At each such step, there is a risk of an incorrect prediction for some i. However, again at each step there are at most β fraction of such bad choices i, and by union bound at most $\beta \log m$ fraction of bad choices of i overall. Details appear in Section 6.2.

The Geometry of Distingishers, Isoperimetry. We now turn our attention to proving Theorem 1.13. This entails constructing a predictor when given only *one-sided* insensitivity. Again, the idea is to take a random walk starting at x, and find when it reaches a sensitive point in D. Before, any point in D with at least one neighbor outside D was sensitive, and so our algorithm succeeds as long as the walk eventually leaves D.

A crucial observation is that when a distinguisher is a very small subset of $\{0,1\}^m$, as is in our case, then a large fraction of points in D are sensitive. This fact is known as isoperimetry or "small set expansion".

Now consider a more standard random walk, where instead of considering each j from 1 to m in sequence and flipping it with probability $1/2$, the j-th step of the walk picks a random coordinate of the current string, and flips it with probability $1/2$. This is now exactly a lazy random walk on the Boolean hypercube starting from x. Ideally, we would hope that such a walk starting at x reaches a sensitive node in a small number of steps, since then, our previous intuition allows us to argue that many coordinates are predictable. Unfortunately, this is untrue. Consider the following example.

Example 2.1. *Let $J \subseteq [m]$, $|J| = \sqrt{m}$. For $x, y \in \{0,1\}^m$ let the set of indices where x and y differ be*

$\Delta(x, y)$. *Let D contain the elements close to X, except those that differ from $G(y)$ on indices in J:*

$$D = \{z : \exists x \in \mathrm{Supp}(X), (\Delta(z, x) \cap J = \emptyset) \wedge |\Delta(z, x)| \leq 0.1m\}.$$

In words, D consists of balls of radius $0.1m$ centered around points in X, but excludes points within a ball if they don't exactly agree with the center point on a fixed subset of $\sqrt{m}$ vertices J. We expect that a typical random walk starting at X flips a coordinate in J within $\sqrt{m} \ll 0.1m$ steps, and thus leaves D and never comes back. In particular, the walk will likely never reach a point of distance exactly $0.1m$ away from x that is also in D, which has $0.9m$ neighbors outside D.

We remedy this situation as follows. First, suppose that the random walk starting at x is not a random walk on all of $\{0,1\}^m$, but only on the subgraph induced by D. Then by standard mixing and expansion properties, this walk hits a sensitive node (which are a large fraction of points in D) in $O(\log |D|)$ steps. Observe that the predictor can perform such a random walk. At step j, whenever both $v^j \notin D$ and $w^j \notin D$, the predictor reverts the walk to the previous step. It then uses fresh randomness to take another step in $\{0,1\}^m$ and repeats, accepting the new step iff both the new $v^j, w^j \in D$. As usual, whenever $D(v^j) \neq D(w^j)$, the predictor terminates with the correct prediction. Again, we can bound the number of choices of i for which this predictor can be wrong using the insensitivity property. This effectively simulates a random walk in the subgraph induced by D. For details, see Section 7.2.

Weak Unpredictability Generators with Minimal Hardness Loss. We have so far established our efficient distinguisher to predictor transformations. We conclude by giving a brief overview of how to construct weak unpredictability generators with minimal hardness loss. The techniques to do so are by now standard, although our specific parameter settings are not usually considered. First, we can instantiate the unpredictability generator of [STV01] (by using a locally list decodable code with distance $1/2 - \varepsilon$ for $\varepsilon \in [n^{-.01}, .01]$). This incurs minimal hardness loss, but still has seed length $O(\log m)$. Call this generator $G_2 \colon \{0,1\}^{O(\log m)} \to \{0,1\}^m$.

In order to obtain $(1 + \alpha) \cdot \log m$ seed length, we use a trick from Chen and Tell [CT21]. They observe that there is a pseudorandom generator $G_1 \colon \{0,1\}^{1 \cdot \log m} \to \{0,1\}^{m^{0.01}}$ with a $1 \cdot \log m$ bit seed against circuits of size roughly m. Thus, if G_2 runs in time nearly linear in m, then $G_2 \circ G_1 \colon \{0,1\}^{1 \cdot \log m} \to \{0,1\}^m$ is an unpredictability generator for circuits of size m. Crucially, under the batch fine-grained assumption, G_2 is indeed computable in time nearly linear in m. Details appear in Section 8.

We remark that in [CT21], G_2 is a linear-time cryptographic PRG from one-way functions. One of the conceptual observations of our work is that we can instead take G_2 to be any PRG for circuits with minimal hardness loss. As several other works have pointed out, these two generators for G_2 "beat the hybrid argument" in two fundamentally different ways. A cryptographic PRG uses the assumption of one-way functions, which is strong enough that the losses of the hybrid argument are irrelevant. On the other hand, we construct G_2 from circuit lower bounds, where losses in the hybrid argument matter. The core goal of this work is to tackle this problem head on by suggesting new techniques and efficient reductions.

2.1 A New Perspective on the Hybrid Argument

So far we have outlined how our new distinguisher to predictor transformations work, which exploit the properties of insensitivity. We now discuss how our predictors relate to the standard

predictors obtained via the hybrid argument. We also provide intuition why exploiting such structure is necessary in order to achieve predictors with large advantage.

The Hybrid Argument Revisited. We first review the original hybrid argument at a high level, with a twist in perspective. Recall that the hybrid argument shows that between every pair of hybrid distributions $H_{i-1} = U_1, \ldots, U_{i-1}, X_i, \ldots, X_m$ and $H_i = U_1, \ldots, U_{i-1}, U_i, X_{i+1} \ldots, X_m$, there is a distinguishing advantage of D, which is the frequency that the former distribution lands in D, minus the frequency the latter does so.[5] Namely,

$$\varepsilon_i = \Pr[U_1, \ldots, U_{i-1}, X_i, \ldots, X_m \in D] - \Pr[U_1, \ldots, U_{i-1}, U_i, X_{i+1} \ldots, X_m \in D].$$

The assumption is that $\sum_i \varepsilon_i \geq \varepsilon$ and so D distinguishes with advantage ε/m between *some* pair of hybrids for some i. At a very high level, this suggests the following. Suppose that we are given a random i and X_{-i} and wish to predict X_i. We can first replace the first $i-1$ bits $X_1, \ldots, X_{i-1}$ with random bits $U_1, \ldots, U_{i-1}$, and call D on this new string with 0 for the i-th position and with 1 for the i-th position. The distinguishing advantage of D between H_{i-1} and H_i suggests that with probability ε (over randomness of i, X, and "internal" randomness $U_1, \ldots, U_m$) the string that is the original X (in other words, when $U_i = X_i$ rather than $U_i = \overline{X}_i$) is inside D and the other string is outside D. Thus a correct prediction can be made in such a case.[6] This gives a predictor with advantage ε/m.

A Barrier To the Hybrid Argument. Unfortunately, such a predictor can't have large advantage $\gg 1/m$ without new ideas. For intuition as to why, we consider the following informative example. Suppose that X, the distribution of our unpredictability generator, is supported uniformly on $\mathrm{poly}(m)$ strings in $\{0,1\}^m$. Suppose further that these strings form a code: for any $x, x' \in \mathrm{Supp}(X)$, the distance between x and x' is at least, say, $m/4$. We define a $D \subset \{0,1\}^m$ as follows. Partition $\mathrm{Supp}(X)$ into $m/1000$ equal size sets $S_1, \ldots, S_{m/1000}$. For each $i \in [m/1000]$ we place a ball of radius i centered around every point $x \in S_i$. Define D to be the union of these balls.

We assert that the distinguishing advantage of D between any two hybrids H_{i-1}, H_i is at most $1000/m$. To see this, we first present yet another small yet crucial change in perspective. The hybrid distribution $U_1, \ldots, U_i, X_{i+1}, \ldots, X_m$ is equivalent to $(X_1 \oplus U_1), \ldots, (X_i \oplus U_i), X_{i+1}, \ldots, X_m$. In other words, we view the randomness in the i-th hybrid $U_1, \ldots, U_i$ as instructions to shift $X_1, \ldots, X_m$ to another string. Importantly, the instructions are the same regardless of initial value of $X_1, \ldots, X_m$.

Now first consider $1 \leq i \leq m/16$. If for such an i, the distinguishing advantage of D was too large, there exists a fixing to $u_1, \ldots, u_{i-1} \sim U_1, \ldots, U_{i-1}$ such that the advantage is preserved. Let $\Delta \leq i \leq m/16$ be the number of 1-s in this fixing. This means that with probability at least $1000/m$ over X, adding $u_1, \ldots, u_{i-1}$ to X yields a string still in D, and flipping the i-th bit yields a string outside of D. However, this can only happen when X is in partition S_Δ and this can only happen with probability $1000/m$ (if $\Delta \leq m/1000$) or 0 (if $m/1000 < \Delta \leq m/16$).

On the other hand, suppose $i > m/16$. Notice that the total size of D is at most $\mathrm{poly}(m) \cdot 2^{H(1/1000)m} \leq 2^{0.012m}$. Thus, the probability that $H_i = U_1, \ldots, U_{i-1}, X_i, \ldots, X_m$ is outside of D is

[5] Historically, the hybrid argument replaces X_i-s with U_i-s starting at the end of the string.

[6] Note that this is not quite the same distinguisher-to-predictor transformation as the standard hybrid argument uses, but we phrase it this way to present it closer to our intuition.

at least $1 - 2^{(0.012-1/16)m} \geq 1 - 2^{-0.05m}$. Similarly, the probability that H_{i+1} is outside D is also at least this value. By union bound, with all but exponentially small probability, *both* hybrids are outside D, and so no prediction can be made. In other words, the distinguishing advantage in this case is exponentially small.

Remark 2.1. In general, it is unclear whether the standard constructions of unpredictability generators $G^f \colon \{0,1\}^d \to \{0,1\}^m$ are codes with good distance.[7] Nevertheless, the example above suggests that we require a fundamentally new idea to obtain large prediction advantage.

One may ask whether it is interesting to consider such a distinguisher D that depends so heavily on X itself, the output distribution of the PRG. We point out that many previously known constructions of PRGs (and unpredictability generators) G^f for circuits from a hard function f are "black box". This means that regardless of the function f and construction G^f, it is still interesting to consider arbitrary D-s. Indeed, for any such function, using the fact that it violates the guarantee of the PRG (no matter how severely) implies an efficient black-box reduction from D to computing f.

Thus, if there are constructions of black-box unpredictability generators that are also codes, the above example suggests that fooling such distinguishers requires generating $(1/m)$-unpredictability, even under the stronger notion of symmetric unpredictability, where all coordinates except i are given as input. This would then require using inefficient reductions and incur large polynomial losses in hardness.

We remark that [FSUV13] already provide a proof that the loss in the hybrid argument is inherent in black box reductions (see Fact 4.1 there). Our example provides intuition for a barrier to the hybrid argument as we interpret it for our purposes.

Circumventing the Barrier. The predictor of Theorem 1.13 obtains a prediction advantage much larger than $O(1/m)$ for the example above. The core reason is that it exploits the structure of the example as a $(1 - 1/1000, 1/1000)$-insensitive distinguisher. To compare the predictor to the standard predictor from the hybrid argument, there are three key perspective shifts.

- The purpose of considering hybrid H_j is less about predicting coordinate j itself, but rather moving X to a point of distance (roughly) j away from it. Thus, we can use hybrid H_j to predict *any* coordinate i, as long as X is moved to a point H_j such that $H_j \in D$ and $H_j + e_i \notin D$. This "decouples" the step j of the hybrid argument with the coordinate i we are trying to predict.

- In the example considered above, if hybrid H_j moves X to a point on the "border" of a ball of radius j (i.e. to a distance exactly j from X), then there are $m - j$ choices of i for which $H_j \in D$ and $H_j + e_i \notin D$. In other words, there are *many* choices of i for which we can correctly predict.

- Each hybrid H_j may only succeed in reaching a "border" for $O(1/m)$ fraction of starting points X. However, our predictor can call $D(H_j)$ and $D(H_j + e_i)$ *for every* j, and output a prediction the first time these calls differ in output. With high probability over the randomness of the hybrids $U_1, \ldots, U_m$, such a j exists, since it's very likely that H_j is outside of D eventually. This predictor succeeds with high probability over its internal randomness for *every* input $X \in D$.

[7] However, we do know that they satisfy the weaker property that the pairwise distances are large on average.

Overall the three perspective shifts above suggest that each hybrid H_j represents the distribution at step j obtained by starting from X and performing a very specific type of random walk. In this walk, each coordinate j is considered sequentially from 1 to m, and flipped with probability $1/2$. On input i, our predictor runs the two coupled random walks $H_0, \ldots, H_m$ and $H_0 + e_i, \ldots, H_m + e_i$. The former starts at X while the latter starts at $X + e_i$, although the algorithm does not know which one is which: it only places 0 and 1 in the i-th coordinate. By assumption, the walk starting at X is initially in D, and with high probability ends outside of D. Therefore, at some step j it reaches a point *on the border* of D. At this point, at least one neighbor of H_j is outside of D, and so there are at least α fraction of choices of i for which the predictor would have terminated with a correct prediction at this step.

3 Preliminaries

The density of a set $B \subseteq A$ is $\mu_A(B) = \frac{|B|}{|A|}$ (when A is clear from context, we shall omit it). For a set A, by $x \sim A$ we mean x is drawn uniformly at random from the uniform distribution over the elements of A. For a function $f\colon \Omega_1 \to \Omega_2$, we say f is *explicit* if there exists a deterministic procedure that runs in time $\mathrm{poly}(\log |\Omega_1|)$ and computes f. Finally, for $f, g\colon \mathbb{N} \to \mathbb{N}$, we say that $f(n) = \widetilde{O}(g(n))$ if there exists a constant k such that $f(n) = O(g(n) \log^k g(n))$. All logarithms are in base 2 unless otherwise specified.

Unless otherwise specified, all arithmetic on bits is done in $\mathbb{F}_2$. Vectors with entries in $\mathbb{F}_2$ are generally denoted in bold: $\mathbf{c}$. We denote $\mathbf{e}_i$ to be the standard basis vector with 1 in the i-th coordinate.

When capital letters denote numbers (as opposed to random variables or sets), the corresponding lower case letters generally denote the logarithms of the capital letters. For example, $\log M = m$ and $N = 2^n$. The only specific exception to this rule is T and t, where T is used to denote the maximum number of time steps, and t denotes an arbitrary index $t \in [T]$.

For a graph $G = (V, E)$, and subset $S \subset V$ define $\delta_G(S) = \{\{u, v\} \in E : |\{u, v\} \cap S| = 1\}$.

3.1 Circuits and Worst-Case Hardness

For a circuit $C\colon \{0,1\}^n \to \{0,1\}$, we often say that $x \in C$ if $C(x) = 1$, and $x \notin C$ otherwise. We also use $|C|$ to denote $\mathrm{Supp}(D) = |\{x : x \in C\}|$.

The size of a Boolean circuit C is the number of its *wires*. We note that the size of C is different than $|C|$. We'll often make the distinction clear by saying either the *circuit size* or the *support size*. We will extensively use the fact that $\mathbf{DTIME}(t(n)) \subseteq \mathbf{SIZE}(O(t(n) \log t(n)))$, where $\mathbf{SIZE}(t(n))$ is the set of languages $L \subseteq \{0,1\}^n$ for which there exists a circuit family $\{C_n\}_n$, such that each C_n is of size $t(n)$, and for every $x \in \{0,1\}^n$ it holds that $x \in L$ if and only if $C_n(x) = 1$ [PF79].

Definition 3.1 (worst case hardness against circuits)**.** *Let $f\colon \{0,1\}^n \to \{0,1\}$. We let $\mathsf{size}(f)$ denote the size of the smallest circuit that computes f. We say f is hard for circuits of size s if $\mathsf{size}(f) > s$.*

We also use the notion of average case hardness:

Definition 3.2 (average case hardness against circuits)**.** *Let $f\colon \{0,1\}^n \to \{0,1\}$. We say that f is δ-average case hard f for circuits of size s if for every circuit $C\colon \{0,1\}^n \to B$, it holds that:*

$$\Pr_{X \sim U_d}[f(X) = C(X)] \leq 1 - \delta.$$

We often identify f with its truth table, and thus refer to $f\colon \{0,1\}^{\log n} \to \{0,1\}$ as $f \in \{0,1\}^n$.

3.2 Random Variables, Min-Entropy

We let U_r denote the uniform distribution on r bits. The *support* of a random variable X distributed over some domain Ω is the set of $x \in \Omega$ for which $\Pr[X = x] \neq 0$, which we denote by $\mathrm{Supp}(X)$. The *min-entropy* of a random variable X is defined by

$$H_\infty(X) = \min_{x \in \mathrm{Supp}(X)} \log \frac{1}{\Pr[X = x]}.$$

A random variable X is an (n,k) source if X is distributed over $\{0,1\}^n$ and has min-entropy at least k. When n is clear from the context we sometimes omit it and simply say that X is a k-source.

3.3 Unpredictability

This work focuses mostly on the connection between unpredictability and pseudoentropy. We begin by recalling the definition of unpredictability:

Definition 3.3. *We say that a sequence of random bits $X_1, \ldots, X_m$ is ε-unpredictable against circuits of size s if for every $i \in [n]$, and for any circuit $P\colon \{0,1\}^{i-1} \to \{0,1\}$ of size s, we have:*

$$\Pr[P(X_1, \ldots, X_{i-1}) = X_i] \leq \frac{1}{2} + \varepsilon.$$

We also will work with a stronger notion of unpredictability, that is, unpredictable in any order:

Definition 3.4. *We say that a sequence of random bits $X_1, \ldots, X_m$ is ε-symmetrically unpredictable against circuits of size s if for every $i \in [n]$, and for any circuit $P\colon \{0,1\}^{m-1} \to \{0,1\}$ of size s, we have:*

$$\Pr[P(X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_m) = X_i] \leq \frac{1}{2} + \varepsilon.$$

Informally, we often say that a (symmetrically) unpredictable distribution is only "weakly unpredictable," or has "weak unpredictability" if $\varepsilon \gg 1/m$.

3.4 Pseudoentropy

Next, we discuss *computational* notions of min-entropy, or, (min-) *pseudoentropy*. We start with the standard, widely used, notion of pseudoentropy due to Håstad et al. [HILL99].

Definition 3.5 (HILL pseudoentropy)**.** *Let X be a random variable distributed over $\{0,1\}^n$, s a positive integer, and $\varepsilon > 0$. We say that $H^{\mathrm{HILL}}_{s,\varepsilon}(X) \geq k$ if there exists a random variable $Y \sim \{0,1\}^n$ with $H_\infty(Y) \geq k$ such that for every circuit $D\colon \{0,1\}^n \to \{0,1\}$ of size s, $|\mathbb{E}[D(X)] - \mathbb{E}[D(Y)]| \leq \varepsilon$.*

A weaker notion, given by Reingold [Rei03] and by Barak, Shaltiel and Wigderson [BSW03] allows the random variable having true min-entropy to depend on the distinguisher itself.

Definition 3.6 (metric pseudoentropy)**.** *Let X be a random variable distributed over $\{0,1\}^n$, s a positive integer, and $\varepsilon > 0$. We say that X has k bits of metric pseudoentropy against a class of circuits $\mathcal{D}$ if for every circuit $D \in \mathcal{D}$ where $D\colon \{0,1\}^n \to \{0,1\}$, there exists $Y \sim \{0,1\}^n$ such that $H_\infty(Y) \geq k$ and $|\mathbb{E}[D(X)] - \mathbb{E}[D(Y)]| \leq \varepsilon$.*

We record the following standard fact in pseudorandomness.

Claim 3.7. *If $X \sim \{0,1\}^n$ is a random variable such that the smooth min-entropy $H^{\varepsilon}_{\infty}(X) \geq k$ then for every set $D \subseteq \{0,1\}^n$ it holds that $\Pr[X \in D] \leq \frac{|D|}{2^k} + \varepsilon$.*

Barak et al. also give such a result in the computational world, when we use metric pseudoentropy. We will utilize this characterization of metric pseudoentropy extensively.

Lemma 3.8 ([BSW03])**.** *Let X be a random variable distributed over $\{0,1\}^n$, and $\varepsilon > 0$. Then, for any class of circuits $\mathcal{D}$ closed under complement, $H^{\mathrm{metric}}_{s,\varepsilon}(X) \geq k$ if and only if for every circuit $D \in \mathcal{D}$ where $D\colon \{0,1\}^n \to \{0,1\}$, it holds that $\Pr[D(X) = 1] \leq \frac{|D|}{2^k} + \varepsilon$.*

Note that X with this property may not have metric pseudoentropy against classes that are not closed under complement. Still, this characterization is interesting even for such classes, and we call it *evasive pseudoentropy*.

Definition 3.9 (evasive pseudoentropy)**.** *Let X be a random variable over $\{0,1\}^n$, and $\varepsilon > 0$. We say that X has ε-evasive pseudoentropy against a class of circuits $\mathcal{D}$ if for all $D \in \mathcal{D}$, it holds that*

$$\Pr[X \in D] \leq \frac{|D|}{2^k} + \varepsilon$$

3.5 Pseudoentropy Generators and Pseudorandom Generators

A pseudorandom generator against a class $\mathcal{D}$ is a function whose output distribution fools any function from $\mathcal{D}$.

Definition 3.10 (PRG)**.** *We say that a distribution $X \sim \{0,1\}^n$ ε-*fools *a function $D : \{0,1\}^n \to \{0,1\}$ if $|\Pr[D(X) = 1] - \Pr[D(U_n) = 1]| \leq \varepsilon$. Let $\mathcal{D} \subseteq \{\{0,1\}^n \to \{0,1\}\}$ be a class of functions. We say that $G\colon \{0,1\}^d \to \{0,1\}^n$ ε-fools $\mathcal{D}$, or is an ε-*PRG against *$\mathcal{D}$, if for every $D \in \mathcal{D}$, $G(U_d)$ ε-fools D. When $\mathcal{D}$ is the class of functions computable by circuits of size s, we say that G ε-fools circuits of size s.*

We also define hitting set generators.

Definition 3.11 (HSG)**.** *Let $\mathcal{D} \subseteq \{0,1\}^n \to \{0,1\}$ be a class of functions. We say that $G\colon \{0,1\}^d \to \{0,1\}^n$ is an ε-hitting-set-generator (HSG) against $\mathcal{D}$ if for every $D \in \mathcal{D}$, such that $\Pr[U_n \in D] \leq \varepsilon$, there exists a $y \in \{0,1\}^d$ such that $G(y) \notin D$.*

When the output of a generator is not pseudorandom but does have high pseudoentropy, we say that the generator is a *pseudoentropy generator*. Our pseudoentropy generators will output random variables having high evasive pseudoentropy.

Definition 3.12 (evasive PEG)**.** *We say that $G\colon \{0,1\}^d \to \{0,1\}^n$ is a (k, ε) evasive pseudoentropy generator (PEG) against a class of circuits $\mathcal{D}$ if $G(U_d)$ has k bits of pseudoentropy against $\mathcal{D}$.*

4 Barriers On Distinguisher to Predictor Transformations

As mentioned in the introduction, nearly all existing randomized distinguisher to predictor transformations use the distinguisher as a black box and don't depend on the generator. In this section, we prove that any such transformation requires a linear number of queries to the predictor. Therefore, it is no better than using the hybrid argument and the union bound.

Our results hold even for a strong type of distinguisher.

Definition 4.1. *The set $D \subseteq \{0,1\}^n$ is a strong ℓ-distinguisher for $G\colon\ \{0,1\}^d \to \{0,1\}^n$ if $\mathrm{range}(G) \subseteq D$ and $2^{2d} \leq |D| \leq 2^\ell$.*

We will consider the following type of black box transformations.

Definition 4.2. *An (ε, ℓ)-black box (bb) predictor on n bits is an oracle circuit $P^D\colon [n] \times \{0,1\}^{n-1} \to \{0,1\}$ such that for any function $G : \{0,1\}^d \to \{0,1\}^n$, whenever $D = D_G \subseteq \{0,1\}^n$ is a strong ℓ-distinguisher for G, we have that P^D is an ε-predictor for G.*

We now show that such black box transformations require many queries. Note that our theorem is stronger for using strong distinguishers, as it should be easier to get a predictor from a strong distinguisher. The reader might first think of $\ell = 0.9n$ and $\varepsilon = 0.01$. In particular, the lower bound below is $\Omega(n)$ for these parameters.

Theorem 4.3. *Any (ε, ℓ)-bb predictor on n bits makes at least $\Omega(\varepsilon\ell \log((2n)/\ell))$ queries to the distinguisher.*

Proof: Fix an (ε, ℓ)-bb predictor $P = P^D$ on n bits, as well as a function $G : \{0,1\}^d \to \mathbb{F}_2^n$. For $v \in \mathbb{F}_2^n$, let $G_v(y) = G(y) + v$. We construct distinguishers D_v for G_v, probabilistically, with certain correlations. We can then exploit that the predictor doesn't "know" which v was chosen, and hence which D_v it queries and which G_v it tries to predict.

Let $Z_v = \mathrm{range}(G_v)$. For each $v \in \mathbb{F}_2^n$, choose a uniformly random $R \in \{0, 1, \ldots, r-1\}$ for r to be chosen shortly. We let

$$s \in D_v \text{ iff } R_v \geq \Delta(s, Z_v),$$

where Δ denotes Hamming distance. Thus, $\Pr[s \in D_v] = \max(0, 1 - \Delta(s, Z_v)/r)$. In particular, $Z_v \subseteq D_v$.

Moreover, each D_v is small. Specifically, since D_v only contains s with $\Delta(s, Z_v) \leq r$, we have

$$|D_v| \leq |Z_v| \sum_{i=0}^{r} \binom{n}{i} \leq |Z_v| \cdot 2^{H(r/n)n} \leq 2^{H(r/n)n+d},$$

where $H()$ is the binary entropy function.

We want D_v to be a strong ℓ-distinguisher for G_v, so we need $2^{H(r/n)n+d} \leq 2^\ell$. Since $\ell \geq 2d$, this is implied by $\ell \geq 2H(r/n)n$. This equation is satisfied for some $r = \Theta(\ell/\log(2n/\ell))$, using that $H(p) = \Theta(p \log(1/p))$ for $p \leq 1/2$. This is our choice of r.

Now, crucially, observe that if $\Delta(v, v') = 1$, then for any s, $\Pr_R[D_v(s) \neq D_{v'}(s)] \leq 1/r$. This is because $|\Delta(s, Z_v) - \Delta(s, Z_{v'})| \leq 1$, so there is at most one choice of R for which $D_v(s) \neq D_{v'}(s)$.

We now show that if P makes at most q queries, then the advantage $\varepsilon \leq q/r$, implying $q \geq \varepsilon r$. This will complete the proof.

Consider the distribution where $v \in \mathbb{F}_2^n$ and $i \in [n]$ are chosen uniformly and independently. Further choose $Y \sim \{0,1\}^d$ uniformly and independently and set $X_v = G_v(Y)$. For ease of notation,

let $X \stackrel{\text{def}}{=} X_v$. Condition on i and v_{-i}. This determines $X_{-i} = G_v(Y)_{-i}$. The predictor P receives (i, X_{-i}), and seeks to determine X_i. Let v^j denote v_{-i} with a j in the i'th location, so $v^1 = v^0 + e_i$. The point is that each query answer to D_{v^0} is the same as for D_{v^1}, except with probability $1/r$ over R. Thus, the statistical distance of the q query answers for D_{v^0} and D_{v^1} is at most q/r. Hence, P returns the same answer for these two distinguishers except with probability at most q/r. However, X_{v^0} and X_{v^1} always disagree on the i-th coordinate, by definition. Hence,

$$\Pr[P \text{ correct for } v^0] + \Pr[P \text{ correct for } v^1] \leq 1 + q/r.$$

Thus the advantage $\varepsilon \leq q/(2r)$. ■

5 Pseudoentropy from Weak Unpredictability via a "Truncated" Hybrid Argument

In this section, we provide a simple argument that shows that a weakly unpredictable distribution has some amount of evasive pseudoentropy. The theorem presented here serves as a baseline for the parameters one would hope to beat using a new analysis.

Theorem 5.1 (truncated hybrid argument)**.** *For any $k \leq m$, suppose that $X = X_1, \ldots, X_m$ is ε/k-unpredictable for a class of circuits $\mathcal{D}$ of size at most s. Let $\mathcal{D}_k$ be the set of all $D \in \mathcal{D}$ of size $s - O(1)$ such that $|D| < 2^k$. Then, for any $D \in \mathcal{D}_k$, we have that*

$$\Pr[X \in D] \leq \frac{|D|}{2^k} + \varepsilon.$$

In other words, X has k bits of evasive pseudoentropy for the class of functions in $\mathcal{D}_k$.

Proof: Suppose otherwise that

$$\Pr[X \in D] > \frac{|D|}{2^k} + \varepsilon$$

for some $D \in \mathcal{D}_k$. Define the "hybrids" distributions $H_0, \ldots, H_k$, with $H_0 = X$, and

$$H_i = U_1, \ldots, U_i, X_{i+1} \ldots X_m,$$

where the U_j's are uniform and independent. Since the first k bits of H_k are uniform, H_k is a k-source, so

$$\Pr[H_k \in D] \leq \frac{|D|}{2^k},$$

and therefore

$$\Pr[H_0 \in D] - \Pr[H_k \in D] > \varepsilon.$$

The proof proceeds as usual (see e.g. [Vad12], Proposition 7.16), by concluding that there exists an i such that $\Pr[H_i \in D] - \Pr[H_{i-1} \in D] > \varepsilon/k$, in contradiction. ■

For concreteness, suppose that $k = m^\theta$ for some small constant θ. The above theorem then states that even $\approx m^{-\theta}$ unpredictability (for length X of length m) yields some form of computational entropy. We point out that the standard techniques to construct an ε/k-unpredictable distribution for circuits of size s from a worst-case hard function f requires f to be hard for circuits of size

$s \cdot \mathrm{poly}(k/\varepsilon)$. Since $k \approx \log|D|$, this means the size blowup is at least $\geq s \cdot \log^c|D|$ for some constant $c \geq 2$. As a rough sketch of why, consider the construction of [STV01] with unpredictability ε/k. One can recall that given a distinguisher D of size s, there is a predictor for some coordinate of X of the same size. To compute f, this predictor is called as many times as there are queries in a locally decodable code with distance $\frac{1}{2} + \frac{\varepsilon}{k}$, which is $\mathrm{poly}(k/\varepsilon)$. In Section 7, we'll give a predictor for which such a size blowup is only be about $s \cdot k$.

6 Predictable Distinguishers and Insensitive Distinguishers

In this section we will work with the notion of an *insensitive distinguisher*, which is a subset D of $\{0,1\}^n$ recognizable by a size s' circuit with some additional properties. We first show that these properties enable an efficient prediction algorithm for any x in D (for many i-s) with α-advantage (for $\alpha \gg 1/n$) that uses roughly $\log|D|$ queries. As we will see later, using such a predictor to contradict a worst case hard function f in the analysis of an α-unpredictability generator from [STV01], results in a circuit of size roughly $\mathrm{poly}(1/\alpha) \cdot \log|D| \cdot s'$ that computes f. When both $\mathrm{poly}(1/\alpha)$ and $\log|D|$ are small this algorithm is efficient. One should also note that unlike in Section 5, α and $\log|D|$ are *decoupled*. Before, we needed $\alpha \approx 1/\log|D|$, and this also incurred a blowup of $\mathrm{poly}(1/\alpha) \approx \mathrm{poly}\log|D|$ rather than $\log|D|$.

Definition 6.1 (predictable distinguisher)**.** *We say that a distinguisher $D\colon \{0,1\}^n \to \{0,1\}$ is α-predictable with q queries and runtime s if there exists a randomized predictor $P\colon [i] \times \{0,1\}^{n-1} \times \{0,1\}^\ell \to \{0,1\}$, where we think of $\{0,1\}^\ell$ as P's random coins, making at most q queries to D, running in time s, such that for every $x \in D$,*

$$\Pr_{i,R}[P(i, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n, R) = x_i] \geq \frac{1}{2} + \alpha.$$

We also consider a "two-sided" version of predictability, which further asserts that for every $x \notin D$, the probability of an incorrect prediction is not too large.

Definition 6.2. *We say that a distinguisher $D\colon \{0,1\}^n \to \{0,1\}$ is (α,β)-predictable with q queries and runtime s if there exists a randomized predictor $P\colon [i] \times \{0,1\}^{n-1} \times \{0,1\}^\ell \to \{0,1\}$, where we think of $\{0,1\}^\ell$ as P's random coins, making at most q queries to D, running in time s, such that for every $x \in D$,*

$$\Pr_{i,R}[P(i, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n, R) = x_i] \geq \frac{1}{2} + \alpha.$$

Furthermore, for every $x \notin D$,

$$\Pr_{i,R}[P(i, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n, R) \neq x_i] \leq \frac{1}{2} + \beta.$$

To ease into our results, we consider first a strong notion of insensitive distinguisher where we insist on "good" neighborhoods for both every $x \in D$ and $x \notin D$.

Definition 6.3 ((α,β)-insensitive distinguisher)**.** *We say that a distinguisher $D\colon \{0,1\}^n \to \{0,1\}$ is* (α,β)-insensitive *if:*

- *(Inside Sensitivity.) For every $x \in D$ with $\Delta_{\overline{D}}(x) = 1$, it holds that $\Pr_{i\sim[n]}[D(x + \mathbf{e}_i) = 0] \geq \alpha$.*

- *(Outside Insensitivity.) For every $x \notin D$, it holds that $\Pr_{i\sim[n]}[D(x+\mathbf{e}_i)=1] \leq \beta$.*

Definition 6.4. *We say that $x \in D$ is "on the border of D," or "just inside D" if there exists an $i \in [n]$ such that $x + \mathbf{e}_i \notin D$. Likewise, we say that $x \notin D$ is "on the border of $\overline{D}$," or "just outside D" if there exists an $i \in [n]$ such that $x + \mathbf{e}_i \in D$.*

Moreover, we say that $x \in D$ is "completely inside D" if x is not on the border of D. We say that $x \notin D$ is "completely outside D" if x is not on the border of $\overline{D}$.

Equipped with the notation above, we present two predictors that succeed with noticeable advantage, given an (α, β)-insensitive distinguisher.

6.1 Warmup: A "Sequential" Predictor

As a warmup that illustrates our technique, we give an algorithm showing that an insensitive distinguisher is predictable with $O(\log |D|)$ queries. The first intuition is a small but crucial shift in perspective of the hybrid argument: The hybrid $H_i = U_1, \ldots, U_i, X_{i+1}, \ldots, X_n$ is not just X with the first i bits replaced with uniform bits. It is X with the first i-bits *shifted* by uniform bits. Thus, we can view the distribution H_i as the distribution of the i-th step of a (very specific type of) "random walk" starting at X: one that randomly flips each bit starting from 1 to n.

Theorem 6.5. *For any $T \in [n]$, any (α, β)-insensitive distinguisher $D\colon \{0,1\}^n \to \{0,1\}$ is ε-predictable with $O(T)$ queries and runtime $O(n)$, where*

$$\varepsilon = \frac{1}{2}(\alpha - \beta) - |D| \cdot 2^{-T}.$$

Before proving the theorem, let us first interpret it. Suppose, for example, that $\alpha - \beta$ is a constant, and T is only $\Theta(\log |D|)$. Then, we gain a predictor with small constant advantage, that requires only $O(\log |D|)$ queries to D. Since the prediction advantage is now constant (or at least does not depend on $\log |D|$, as it did in Section 5), the reduction from a predictor to a circuit that computes a (worst-case) hard function would no longer suffer a blowup that is polynomial in $\log |D|$ as before. Instead, there is a *linear* blowup of $T = O(\log |D|)$ incurred by using queries to D.

To prove the theorem, we consider the algorithm SeqPred, which uses randomness $R = (b_1, \ldots, b_T, b)$, where $b_i, b \in \{0,1\}$. As discussed before, the idea, similar to the original hybrid argument, will be to repeatedly replace x_i with the random bits b_i starting from the first bit. This is (a very specific kind of) "random walk" on the Boolean hypercube we take, starting from x.

SeqPred:
Input: $i \in [n]$, $x_{-i} \in \{0,1\}^{n-1}$, and $R = (b_1, \ldots, b_T, b)$.
Output: A bit that predicts x_i from x_{-i}.

1. Let $\mathbf{W}_0^0 \in \mathbb{F}_2^n$ be x_{-i} with 0 in the i-th coordinate, and likewise let $\mathbf{W}_0^1$ be x_{-i} with 1 in the i-th coordinate.
2. For each $t \in [T]$, let $\mathbf{v}_t = b_t \cdot \mathbf{e}_t$.
3. Let $\mathbf{W}_0^0, \ldots, \mathbf{W}_T^0$ and $\mathbf{W}_0^1, \ldots, \mathbf{W}_T^1$ be such that $\mathbf{W}_t^0 = \mathbf{W}_0^0 + \mathbf{v}_1 + \ldots + \mathbf{v}_t$, and likewise $\mathbf{W}_t^1 = \mathbf{W}_0^1 + \mathbf{v}_1 + \ldots + \mathbf{v}_t$.
4. For $t = 1, \ldots, T$,
   (a) If $t = i$ and $b_i = 1$, let fix $= 1$, otherwise let fix $= 0$.
   (b) If $\mathbf{W}_t^0 \in D$ and $\mathbf{W}_t^1 \in D$, continue to the next iteration.
   (c) If $\mathbf{W}_t^0 \in D$ and $\mathbf{W}_t^1 \notin D$, then return $0 + \text{fix} \pmod 2$.
   (d) If $\mathbf{W}_t^1 \in D$ and $\mathbf{W}_t^0 \notin D$, then return $1 + \text{fix} \pmod 2$.
   (e) If $\mathbf{W}_t^0 \notin D$ and $\mathbf{W}_t^1 \notin D$, return b.
5. Return b.

Proof (of Theorem 6.5): We show that SeqPred is the predictor with the claimed parameters. Fix any $x \in D$ and consider random i and R. Let $b^* \in \{0,1\}$ be such that $x = \mathbf{W}_0^{b^*}$, and denote, for any t, $\mathbf{W}_t = \mathbf{W}_t^{b^*}$, noting that this is the random walk that truly starts at x. Also, let $\mathbf{W}'_t = \mathbf{W}_t^{1-b^*}$. Notice that $\mathbf{W}'_t = \mathbf{W}_t + \mathbf{e}_i$.

Call the randomness R *good* (for x) if $\mathbf{W}_0$ eventually leaves D. That is, $R = (b_1, \ldots, b_T, b)$ is good if $\mathbf{W}_t \notin D$ for some $t \in [T]$. Observe the following:

- If $\mathbf{W}_t$ is completely inside D, then the algorithm does not terminate on iteration t.
- If $\mathbf{W}_t$ is just inside D, then the algorithm may or may not terminate on iteration t, depending on the choice of i. If it does terminate, it must terminate with a correct prediction for x_i.
- Likewise, if $\mathbf{W}_t$ is just outside D, then the algorithm may terminate, and if it does, it will terminate with an incorrect prediction for x_i.
- If $\mathbf{W}_t$ is completely outside D, then the algorithm terminates with a random guess for x_i.

Fix any good R. Notice that the probability that R is good is at least $1 - |D| \cdot 2^{-T}$, since $(b_1, \ldots, b_T)$ is a T-source. Let $\ell \in [T]$ be the first time that the walk starting at x reaches the border of D, that is, $\mathbf{W}_\ell$ is just inside D. We know that such an ℓ exists since the walk $\mathbf{W}_t$ eventually leaves D. We further observe two things:

- Assume without loss of generality that $\mathbf{W}_0^0 = x$. If $\mathbf{W}_t^0 = \mathbf{W}_0 + \mathbf{v}_1 + \cdots + \mathbf{v}_t$ is just inside D, then we will observe $\mathbf{W}_t^0 \in D$. If i is chosen so that $\mathbf{W}_t^1 \notin D$, this means that if $t < i$, the algorithm will correctly return 0. Otherwise, if $t \geq i$ and $b_i = 1$, the algorithm will correctly return $1 + \text{fix} = 0$.

- By the same token, if $\mathbf{W}_t^0$ is just outside D, the algorithm will ouput the wrong answer for x_i if $\mathbf{W}_t^0 \notin D$ and $\mathbf{W}_t^1 \in D$.

Let $G \subseteq [n]$ be the set of all i-s such that $\mathbf{W}'_\ell = \mathbf{W}_\ell + \mathbf{e}_i \notin D$. We call G the set of "good" i-s (and i is bad if $i \notin G$). Notice that if SeqPred on x, R, and $i \in G$ does not terminate before step ℓ, it will terminate on step ℓ, and correctly predict x_i. By the "inside sensitivity" property, $\Pr[i \in G] \geq \alpha$.

Claim 6.6. *For any $i \in G$, there is no step $\ell' < \ell$ such that*

$$\mathbf{W}_{\ell'} \notin D$$

Furthermore, there is no step $\ell' < \ell$ such that

$$\mathbf{W}_{\ell'} \in D$$

and

$$\mathbf{W}_{\ell'} + \mathbf{e}_i \notin D.$$

Proof: Suppose there is such a step ℓ' for which $\mathbf{W}_{\ell'} \notin D$. This implies that the walk starting at x reached the border of D at some point before step ℓ. Since ℓ is assumed to be the first such time, this is a contradiction. Similarly, if there is a step ℓ' when the last case happens, this implies that ℓ' is on the border of D, which is again a contradiction. ■

Therefore, for any $i \in G$, and any R such that $\mathbf{W}_t$ leaves D, the algorithm does not terminate before step ℓ. Therefore, for such i and R, SeqPred must terminate at step ℓ, and will do so with a correct prediction for x_i.

What we have shown so far, is that if we fix any good R, then for any good i (for R), the probability of a correct prediction is 1. Since we wish to take the average over all i-s, and still argue that we have some advantage in predicting, we need to ensure that on average over $i \notin G$, the probability of prediction is not too low. For example, we would not want the probability of prediction to be 0 for all $(1-\alpha)n$ of $i \notin G$, as this would counteract the advantage from good i-s.

Let $a \in [t]$ be the first time that $\mathbf{W}_a \notin D$. Notice this implies that $\mathbf{W}_a$ is just outside D, and a is the first such time that the walk is just outside D. Thus, SeqPred could not have terminated with an incorrect prediction before step a. Moreover, at step a, two things can occur:

- SeqPred terminates with an incorrect prediction, if the choice of i causes $\mathbf{W}_a + \mathbf{e}_i \in D$. Call such i *very bad*.
- SeqPred terminates with a random guess, if the choice of i causes $\mathbf{W}_a + \mathbf{e}_i \notin D$ as well. Call such i *bad*.

The number of choices of very bad i-s is at most βn by the "outside insensitivity" property.

Overall, for any fixed good R, the probability of a correct prediction is

$$\begin{aligned}
&\Pr_i\left[\mathsf{SeqPred}(x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_n,i,R)=x_i\right]\\
&=\Pr_i\left[\mathsf{SeqPred}(x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_n,i,R)=x_i|i \text{ is good}\right]\Pr[i \text{ is good}]\\
&+\Pr_i\left[\mathsf{SeqPred}(x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_n,i,R)=x_i|i \text{ is bad}\right]\Pr[i \text{ is bad}]\\
&+\Pr_i\left[\mathsf{SeqPred}(x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_n,i,R)=x_i|i \text{ is very bad}\right]\Pr[i \text{ is very bad}]\\
&\geq \alpha+\frac{1}{2}(1-\alpha-\beta)+0\\
&=\frac{1}{2}+\frac{1}{2}(\alpha-\beta).
\end{aligned}$$

■

We can also give a theorem about the two-sided predictability of such distinguishers using Theorem 6.5.

Corollary 6.7. *For any $T\in[n]$, any (α,β)-insensitive distinguisher $D\colon\ \{0,1\}^n\to\{0,1\}$ is*

$$\left(\frac{1}{2}(\alpha-\beta)-|D|\cdot 2^{-T},\frac{\beta}{2}\right)$$

predictable with $O(T)$ queries and runtime $O(n)$.

Proof: Theorem 6.5 already establishes that for any $x\in D$:

$$\Pr_{i,R}\left[\mathsf{SeqPred}(i,x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_n,R)=x_i\right]\geq\frac{1}{2}+\frac{1}{2}(\alpha-\beta)-|D|\cdot 2^{-T}.$$

For any $x\notin D$, by the outside insensitivity property, there are at most βn choices of i for which SeqPred does not terminate with the random guess b. Therefore,

$$\Pr_{i,R}\left[\mathsf{SeqPred}(i,x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_n,R)\neq x_i\right]\leq\frac{1}{2}(1-\beta)+\beta.$$

■

6.2 Using Binary Search

Note that Theorem 6.5 is nontrivial whenever the number of queries $T\geq\log|D|$, in which case the number of queries to D is $O(\log|D|)$. Using a "binary search" modification to the above predictor, we can in fact reduce the number of queries to $O(\log n)\approx\log\log|D|$.

We can summarize what we learned from the analysis of the sequential predictor SeqPred as follows. When the predictor reaches a time t for which $\mathbf{W}_t$ is just inside D, there is a good probability (over choice of i) that it will terminate with a correct prediction. There is a caveat to this statement. It requires that once time t is reached, the randomness over i is still uniform (or at least has high entropy), in order to use the inside sensitivity property. However, every query made before time t could *reveal information* about i and thus restrict the number of "live" choices for it.

A key observation about the sequential analysis is that when $\mathbf{W}_t$ is the first point on the border of D reached, in every previous time step $t' < t$, $\mathbf{W}_{t'}$ must have been completely inside D. In this case, observing that both $\mathbf{W}_{t'}^0$ and $\mathbf{W}_{t'}^1$ are in D provides no information on the choice of i, since every i leads to this outcome. Thus when time t is reached, i is still uniform.

Overall, we can summarize how the outcome at a given iteration affects the distribution of i:

- If $\mathbf{W}_t$ is completely inside D, there is no effect on the distribution of i.
- If $\mathbf{W}_t$ is completely outside D, there is also no effect on the distribution of i.
- If $\mathbf{W}_t$ is just inside D, then observing that both $\mathbf{W}_t^1$ and $\mathbf{W}_t^0$ are in D restricts the choices of i to those for which $\mathbf{W}_t + \mathbf{e}_i$ is inside D. This is also true when observing $\mathbf{W}_t^1$ and $\mathbf{W}_t^0$ differ in membership to D. However, in this case the algorithm terminates, so there are no future iterations for which one needs to be concerned about the possible choices of i.
- By the same token, if $\mathbf{W}_t$ is just outside D, then observing that both $\mathbf{W}_t^1$ and $\mathbf{W}_t^0$ are outside D restricts the choices of i to those for which $\mathbf{W}_t + \mathbf{e}_i$ is outside D.

Unfortunately, in the case of binary search, it's not guaranteed that we observe a time step t for which $\mathbf{W}_t$ is just inside D, without first observing roughly $\log n$ steps for which $\mathbf{W}_t$ is just *outside* D. Nevertheless, due to the outside insensitivity property, we can argue that each of the time steps on which we observe that $\mathbf{W}_t$ is just outside D cannot restrict the choices of i too much. Hence, if the algorithm eventually reaches a time when $\mathbf{W}_t$ is just inside D, the distribution over i still has high entropy, and so there is still a good chance of a correct prediction (using inside sensitivity).

We now present the binary search algorithm formally.

BinaryPred:
Input: $i \in [n]$, $x_{-i} \in \{0,1\}^{n-1}$, $R = (b_1, \ldots, b_n, b)$.
Output: A bit predicting x_i from x_{-i}.

1. Let $\mathbf{W}_0^0 \in \mathbb{F}_2^n$ be x_{-i} with 0 in the i-th coordinate, and likewise let $\mathbf{W}_0^1$ be x_{-i} with 1 in the i-th coordinate.
2. For each $t \in [n]$, let $\mathbf{v}_t = b_t \cdot \mathbf{e}_t \in \mathbb{F}_2^n$, as in the sequential predictor.
3. Let $\mathbf{W}_0^0, \ldots, \mathbf{W}_n^0$ and $\mathbf{W}_0^1, \ldots, \mathbf{W}_n^1$ be such that $\mathbf{W}_t^0 = \mathbf{W}_0^0 + \mathbf{v}_1 + \ldots + \mathbf{v}_t$, and likewise $\mathbf{W}_t^1 = \mathbf{W}_0^1 + \mathbf{v}_1 + \ldots + \mathbf{v}_t$.
4. Set $\mathsf{lower} = 0$ and $\mathsf{upper} = n$.
5. While $\mathsf{upper} \geq \mathsf{lower} + 1$,
   (a) Let $m = \left\lfloor \frac{\mathsf{lower}+\mathsf{upper}}{2} \right\rfloor$.
   (b) If $m \geq i$, set $\mathsf{fix} = b_i$.
   (c) If $\mathbf{W}_m^0 \in D$ and $\mathbf{W}_m^1 \in D$, set $\mathsf{lower} = m$.
   (d) If $\mathbf{W}_m^0 \in D$ and $\mathbf{W}_m^1 \notin D$, return $0 + \mathsf{fix} \pmod 2$.
   (e) If $\mathbf{W}_m^1 \in D$ and $\mathbf{W}_m^0 \notin D$, return $1 + \mathsf{fix} \pmod 2$.
   (f) If $\mathbf{W}_m^0 \notin D$ and $\mathbf{W}_m^1 \notin D$, set $\mathsf{upper} = m$.
6. Return b.

The following theorem establishes the performance of the binary search predictor.

Theorem 6.8. *Any (α, β)-insensitive distinguisher $D\colon \{0,1\}^n \to \{0,1\}$ is ε-predictable with $O(\log n)$ queries and runtime $O(n)$, where*

$$\varepsilon = \frac{\alpha}{2} - \beta \log n - |D| \cdot 2^{-n}.$$

Proof: As before, fix any $x \in D$ and let $\mathbf{W}_t$ be the walk that started at x, and let $\mathbf{W}'_t = \mathbf{W}_t + \mathbf{e}_i$. Call R good if $\mathbf{W}_n \notin D$ and fix any good R.

Claim 6.9. *For any choice of i, the algorithm either terminates with an incorrect prediction (at some iteration where $\mathbf{W}_m$ is just outside D and the choice of i is appropriate), or eventually reaches an m such that $\mathbf{W}_m$ is just inside D.*

Proof: We can show that at every iteration, $\mathbf{W}_{\mathsf{lower}} \in D$ and $\mathbf{W}_{\mathsf{upper}} \notin D$. This is certainly true at the beginning of the algorithm when R is good. The bounds lower and upper are updated whenever both $\mathbf{W}_m^0$ and $\mathbf{W}_m^1$ are both inside D or both outside D. In either case, this must mean that $\mathbf{W}_m \in D$ or $\mathbf{W}_m \notin D$ appropriately, so the loop invariant holds. Therefore, at every iteration there is always a time step $\mathsf{lower} \leq t < \mathsf{upper}$ for which $\mathbf{W}_t$ is just inside D. If the algorithm never terminates with an incorrect prediction (whenever $\mathbf{W}_m$ is just outside D), then eventually, when $\mathsf{upper} = \mathsf{lower} + 1$, $\mathbf{W}_{\mathsf{lower}}$ must be just inside D, and thus in the next iteration, $\mathbf{W}_m$ is just inside D. ∎

Claim 6.10. *There are at least $(\alpha - \beta \log n)n$ choices of i for which the algorithm will output a correct prediction.*

Proof: We will inductively build sets $B_1, \ldots, B_{\ell^*-1} \subset [n]$, each of size $\leq \beta n$, with $\ell^* \leq \log n$, such that if i avoids all such sets, then there is an iteration in which $\mathbf{W}_m$ is just inside D.

Let q_1 be the first iteration for which $\mathbf{W}_m$ is either just inside D or just outside D. Notice that on this iteration, the time step m we query will be the same regardless of the choice of i (for a fixed good R). This is because the sequence of queries up to q_1 are the same regardless of choice of i. If $\mathbf{W}_m$ is just inside D, then the number of choices of i that lead to a correct prediction is at least αn and we're done. Otherwise, if $\mathbf{W}_m$ is just outside D, there are at most βn choices of i for which the algorithm will terminate with an incorrect prediction on this iteration (by outside insensitivity). Call this set of choices B_1. Notice also, that for the good choices $i \notin B_1$, the next time step m queried in the next iteration is the same.

In general, let $1 \leq q_\ell \leq \log n$ be the ℓ-th iteration for which $\mathbf{W}_m$ is either just inside D or just outside D. If $\mathbf{W}_m$ is just outside D, let B_ℓ be the set of i-s such that $\mathbf{W}_m + \mathbf{e}_i \in D$. Observe that as long as $i \notin B_1 \cup \cdots \cup B_\ell$, the time step m queried in the q_ℓ-th iteration is the same regardless of i. Therefore, until the first iteration q_{ℓ^*} such that $\mathbf{W}_m$ is just inside D, for any $\ell < \ell^*$, as long as $i \notin B_1 \cup \cdots \cup B_\ell$, the next iteration when $\mathbf{W}_m$ is just inside or outside D, $q_{\ell+1}$, queries the same time step regardless of i, and so $B_{\ell+1}$ is well defined.

Eventually, there is an iteration q_{ℓ^*} in which $\mathbf{W}_m$ is just inside D. At this point, there are at least $(\alpha - \beta \log n)n$ choices of i for which the algorithm will terminate with a correct prediction. ■

Call the choices of i in Claim 6.10 good and denote the set of good choices by G. If the algorithm does not choose a good i, it is possible that it makes a mistake and outputs an incorrect prediction later on. However, by the same reasoning as before, such choices will be few.

Claim 6.11. *There are at most $(\beta \log n)n$ choices of $i \notin G$ for which the algorithm will output an incorrect prediction.*

Proof: We can continue identifying sets B_ℓ as before, going beyond iteration q_{ℓ^*}. To do so we assume that, whenever $\mathbf{W}_m$ is just inside D, a switch is not found. Continuing this way, the sets will be well defined inductively, as seen as before. As long as i avoids all such sets B_ℓ, of which there are at most $\log n$ of, the algorithm will not output an incorrect answer (although it might output b as a guess). ■

Again, call the set of choices of i in Claim 6.11 "very bad", and those that are not good and not very bad, as just bad. Overall, since the algorithm will output a random guess as long as i is just bad, the overall prediction advantage over i for a good R is

$$
\begin{aligned}
&\Pr_i\left[\mathsf{BinaryPred}(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n, i, R) = x_i\right] \\
&= \Pr_i\left[\mathsf{BinaryPred}(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n, i, R) = x_i | i \text{ is good}\right] \Pr[i \text{ is good}] \\
&+ \Pr_i\left[\mathsf{BinaryPred}(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n, i, R) = x_i | i \text{ is bad}\right] \Pr[i \text{ is bad}] \\
&+ \Pr_i\left[\mathsf{BinaryPred}(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n, i, R) = x_i | i \text{ is very bad}\right] \Pr[i \text{ is very bad}] \\
&\geq \alpha - \beta \log n + \frac{1}{2}(1 - \alpha - 2\beta \log n) + 0 \\
&= \frac{1}{2} + \frac{\alpha}{2} - \beta \log n.
\end{aligned}
$$

■

Again, by bounding the probability of an incorrect prediction using outside insensitivity when $x \notin D$, we also get two-sided predictability:

Theorem 6.12. *Any (α, β)-insensitive distinguisher $D\colon \{0,1\}^n \to \{0,1\}$ is*

$$\left(\frac{\alpha}{2} - \beta \log n - |D| \cdot 2^{-n}, \frac{\beta}{2} \log n\right)$$

predictable with $O(\log n)$ queries and runtime $O(n)$.

Proof: By similar reasoning to Claim 6.11, whenever $x \notin D$, there can be at most $(\beta \log n)n$ choices of i for which the binary search predictor terminates with an incorrect prediction. If i is none of such choices, the predictor will guess correctly with probability at least $1/2$. ■

7 The More General Case: Removing the Inside Sensitivity Condition

So far, we've considered distinguishers with both the inside sensitivity and outside insensitivity property. Next, we'll show that we can remove the inside sensitivity property by giving a new prediction algorithm. The predictor will make roughly $\log |D|$ queries.

7.1 Fast Mixing of Random Walks Within Small Subsets of the Boolean Hypercube

Towards removing the inside sensitivity condition, we first observe that when D is small, a large fraction of nodes are "sensitive", in the sense that a random step from a sensitive node will exit D with a decent chance. We will then establish the fact that a walk that stays inside D will mix quickly and therefore reach a sensitive node, allowing us to utilize ideas from the previous section.

Let $Q = Q_n$ denote the hypercube with nodes $\{0,1\}^n$. For a set of nodes $D \subseteq \{0,1\}^n$, let $\delta(D)$ denote the number of edges leaving D (i.e., between D and its complement). We will be considering different subgraphs of Q, so we sometimes use $\delta_G(D)$ to indicate it's relative to graph G.

Theorem 7.1 (edge isoperimetry, [Har66])**.** *For any set $D \subseteq \{0,1\}^n$ of size at most $K = 2^k$, we have that $\delta_Q(D) \geq \gamma n |D|$, for $\gamma = 1 - \frac{k}{n}$.*

On a first reading, the reader should think of K as, say, $2^{n^{0.1}}$, in which case γ is very close to 1.

We can assume without loss of generality that D is connected within the Boolean hypercube; otherwise consider the appropriate connected component. Going forward, we fix some $D \subseteq \{0,1\}^n$ of size K, and let G_D be the subgraph of the hypercube that is induced by D. Notice that G_D may not be regular. Also, since D will be clear from context, we will often simply denote $G = G_D$.

Definition 7.2 (the sets H and S)**.** *We let H be the subset of nodes in G_D of degree at least $(1 - \gamma/2)n$. We let $S = D \setminus H$, and also denote G_H as the subgraph of G induced by H. We often call a node in $x \in S$ as "sensitive": flipping a random coordinate of x has at least $\gamma/2$ probability of leaving D.*

We will argue that for any sufficiently small D, a random walk on G_D starting in H "mixes quickly", by showing the following:

- In a random walk starting in H, the expected time to hit S is no slower than the expected time to hit S in a related graph $G^\star$.

- $G^\star$ is a good spectral expander, and thus the expected hitting time within $G^\star$ is $O(\log |G^\star|) \approx \log |D|$.

We argue this way because we cannot directly say that G_D is a good edge expander. Indeed, although we do expect sets within H to expand well, we cannot necessarily say the same about S. We start by arguing that sets within H expand well in G_D.

Lemma 7.3 (edge expansion of H)**.** *Fix D with $|D| \leq K$, and let $G = G_D$. For any subset $A \subseteq H \subseteq D$, $\delta_G(A) \geq (\gamma/2)|A|n$.*

Proof: Within the entire Boolean hypercube Q, there are $|A|n$ edges touching A. By Theorem 7.1, $\delta_Q(A) \geq \gamma n|A|$, since $|A| \leq |D|$. By the definition of H, at most $\gamma/2|A|n$ edges from A leave G_D.

$$\delta_G(A) \geq \delta_Q(A) - (\gamma/2)|A|n \geq (\gamma/2)|A|n.$$

■

We now define $G^\star$. Let G_{Exp} an n-regular expander on $V_{\mathsf{Exp}} = S \cup W$, where W is a set of $3|D|$ new nodes. Concretely, we'll take G_{Exp} to be a $1/4$-edge-expander. That is, for every subset $A \subseteq V_{\mathsf{Exp}}$ of density at most $1/2$, it holds that $\delta(A) \geq n|A|/4$. The probabilistic method shows that such graphs exist (and also, any $1/2$-spectral expander has this property). Note that it also means that sets T of size at most $2|D|$ have at least $|T|n/8$ edges going from T to $\bar{T}$. Indeed, if $|T| < |V_{\mathsf{Exp}}|/2$ this follows from the definition of an edge expander, and otherwise, we still have $|\bar{T}| \geq |D|$ (since $|V| \geq 3|D|$), and so $|D|/4 \geq |T|/8$. Now take $G^\star = (V, E)$ to be the graph over $V = D \cup W$ consisting of the edges in both $G = G_D$ and G_{Exp}. Notice that every node in $G^\star$ has degree at most $2n$.

Lemma 7.4. *$G^\star$ has conductance $\phi(G^\star) \geq \frac{1}{64}\gamma$. That is, for any $A \subseteq V$ of size at most $|V|/2$, we have $\delta_{G^\star}(A) \geq \frac{1}{64}\gamma \cdot \mathrm{vol}(A)$, where $\mathrm{vol}(A)$ is the sum of the degree of vertices in A.*

Proof: Let $\beta = \gamma/2$ and let $A \subseteq V$ be any subset of size at most $|V|/2 = 2|D|$. The idea is to split the nodes in A into those in H and those in G_{Exp}. Within $G^\star$, sets in H expand, and sets in G_{Exp} expand. Therefore A as a whole should expand, as long as there are not too many edges between the nodes of A in H and those in G_{Exp}.

First, suppose that $|A \cap H| \geq (1 - \beta/4)|A|$. By Lemma 7.3, there are at least

$$\beta\left(1 - \frac{\beta}{4}\right) n|A| \geq \frac{\beta}{2} n|A|$$

edges leaving $A \cap H$. We subtract out the number of such edges that lead into $A \cap V_{\mathsf{Exp}}$. These are either edges from the expander G_{Exp} or edges originally from G_D. By construction, no edge from G_{Exp} can touch H, so it suffices to consider the edges from G_D. Since $|A \cap V_{\mathsf{Exp}}| \leq \beta/4|A|$, there at most $(\beta/4)n|A|$ such edges touching A. Subtracting this from the number of edges leaving $A \cap H$, we see that A as a whole must have at least $(\beta/4)n|A| \geq (\gamma/8)n|A|$ edges leaving it.

Next, suppose that $|A \cap H| < (1 - \beta/4)|A|$. This implies that $|A \cap V_{\mathsf{Exp}}| \geq (\beta/4)|A|$. By the above property of G_{Exp}, at least $(\beta/32)n|A|$ edges leave A. This implies the conductance of this set is at least $\gamma/64$. ■

Cheeger's inequality readily gives us a bound on the spectral expansion of $G^\star$ (see, e.g., [Tre13, Chapter 3]).

Corollary 7.5. *Let $\lambda_2(G^\star)$ be the second smallest eigenvalue of the normalized Laplacian matrix of $G^\star$.*[8] *Then, $\lambda_2(G^\star) \geq \phi(G)^2/2 \geq \frac{1}{2^{13}}\gamma^2$.*

Next, we'll show that S has a noticeable probability under the stationary distribution of $G^\star$. This will allow us to argue that the expected time to reach S is roughly $\log |D|$.

Lemma 7.6. *Let $\pi_{G^\star}$ be the unique stationary distribution of $G^\star$ (note that such exists, since $G^\star$ is connected). Then, $\pi_{G^\star}(S) \geq \gamma/8$.*

Proof: We know that, by Theorem 7.1, and an averaging argument, that at least $\gamma/2$ fraction of the nodes in D must have at least $\gamma/2$ fraction of edges leaving D. That is, at least $\gamma/2$ fraction of the nodes in D are in S. For any node in u in S:

$$\pi_{G^\star}(u) = \frac{\deg_{G^\star}(u)}{2|E|} \geq \frac{n}{4n|D|} = \frac{1}{4|D|},$$

where we used the fact that the degree of each $u \in S$ is at least n since G_{Exp} is also over S, and the fact that each node in $G^\star$ has degree at most $2n$. Therefore, summing over all $u \in S$, we get that $\pi_{G^\star}(S) \geq \gamma/8$. ■

Finally, we can argue that the expected hitting time to S from a node in H is short.

Theorem 7.7. *Let u be any node in H. After*

$$O\left(\frac{\log(1/\gamma)}{\gamma^3} \cdot \log \frac{|D|}{\gamma}\right)$$

steps of a lazy random walk on G starting from u, the probability of not reaching S is at most $\gamma/16$.

Proof: The expected time is the same as the expected time for u to reach S in $G^\star$. This is because in constructing $G^\star$, we did not add any edges within H, or between S and H. Since all nodes in V_{Exp} have degree at least n, and all nodes in H have degree at least $n/2$ in $G^\star$, we have that

$$\min_{v \in V} \pi_{G^\star}(v) \geq \frac{1}{8|D|}.$$

Next, we will use standard mixing result of (lazy) reversible, irreducible, Markov chains (see, e.g., [Spi25, Chapter 10]).

Lemma 7.8. *Let $P = \frac{1}{2}(I + \Pi^{-1}A_G)$ be the transition matrix of a lazy random walk over an undirected graph $G = (V, E)$ with an adjacency matrix A_G and a diagonal degree matrix Π, and let $1 = \omega_1 \geq \omega_2 \geq \ldots \geq \omega_{|V|} \geq 0$ be the eigenvalues of P. Moreover, let $\pi_{min} = \min_{v \in V} \pi_G(v)$. Then, for any distribution τ over V, and any $t \in \mathbb{N}$, $\left\|\tau P^t - \pi\right\|_\infty \leq \frac{1}{\sqrt{\pi_{\min}}} \cdot \omega_2^t$ Consequently, for any $\varepsilon > 0$, and*

$$t = O\left(\frac{\log \frac{|V|}{\varepsilon \pi_{\min}}}{\log \frac{1}{\omega_2}}\right),$$

the total variation distance between τP^t and π is at most ε.

[8] Namely, the second smallest eigenvalue of the PSD matrix $I - \Pi^{-1/2} A_{G^\star} \Pi^{-1/2}$, where $A_{G^\star}$ is the adjacency matrix of $G^\star$ and Π is its diagonal degree matrix.

Now, observe that $D^{-1}A$ is similar to $D^{-1/2}AD^{-1/2}$, so $\omega_2 = 1 - \lambda_2/2$, where $\lambda_2 = \lambda_2(G^\star)$ is the second smallest eigenvalue of the normalized Laplacian matrix. Thus, the bound on t above becomes

$$t = O\left(\frac{1}{\lambda_2} \cdot \log \frac{|D|}{\varepsilon \cdot \min_{v \in V} \pi_{G^\star}(v)}\right) = O\left(\frac{1}{\gamma^2} \log \frac{|D|}{\varepsilon}\right),$$

where we used Corollary 7.5. Set $\varepsilon = \gamma/16$. For some $m \in \mathbb{N}$ to be chosen soon, let $t' = tm$. After every t steps, the probability of not landing in S is at most $1 - (\pi_{G^\star}(S) - \varepsilon) \leq 1 - \gamma/16$, using Lemma 7.6. Thus, after t' consecutive steps, the probability of not landing in S is at most $(1 - \gamma/16)^m$, which is at most $\gamma/16$ for $m = O(\frac{1}{\gamma}\ln(1/\gamma))$. ∎

7.2 A Predictor Using Random Walks

We're now ready to give a predictor that works even when D is only outside insensitive. We now restate the relevant part of Definition 1.5.

Definition 7.9 (β-insensitive distinguisher)**.** *We say that a distinguisher $D\colon \{0,1\}^n \to \{0,1\}$ is β-insensitive if for every $x \notin D$, it holds that $\Pr_{i \sim [n]}[D(x + e_i) = 1] \leq \beta$.*

We'll prove the following theorem.

Theorem 7.10. *Let $D\colon \{0,1\}^n \to \{0,1\}$ be a β-insensitive distinguisher with $|D| = K = 2^k$, and let $\gamma = 1 - \frac{k}{n}$, as in Theorem 7.1. Then, D is ε-predictable with $O(T)$ queries and runtime $O(n)$, where $T = O(k \log(1/\gamma)/\gamma^3)$, and*

$$\varepsilon = \frac{\gamma}{8} - \beta \cdot T.$$

Note that for the above theorem to be meaningful, the outside insensitivity parameter β needs to be small, roughly γ^4/k, and recall that $k = \log|D|$.

The idea behind the predictor is to modify the sequential predictor SeqPred, that does require inside sensitivity, in the following two ways:

- Instead of resampling each coordinate from 1 to n sequentially, at every step we resample a random coordinate in $[n]$. Every step essentially simulates a step in a (lazy) random walk on the Boolean hypercube.
- Instead of terminating with a random guess when both $\mathbf{W}_t^0$ and $\mathbf{W}_t^1$ are outside of D, we continue with the random walk: we go one step back, and use fresh randomness to sample yet another step on the hypercube. This essentially simulates a step in a (lazy) random walk *on* G_D.
- By setting the number of steps T to be the hitting time of G_D according to Theorem 7.7, we get that with high probability over the randomness of the walk, we would have reached a sensitive node in G_D: one with many neighbors outside of D. Similar to previous analyses, if x is at such a node, then since i is uniform, it is likely that $D(x) = 1$ and $D(x + e_i) = 0$. Thus we can argue that there would have been many choices of i for which the algorithm would have correctly predicted.

SeqPredWithRestarts:
Input: $i \in [n]$, $x_{-i} \in \{0,1\}^{n-1}$,
and $R = ((z_1, b_1), \ldots, (z_T, b_T), b)$, where $z_1, \ldots, z_T \in [n]$ and $b_1, \ldots, b_T, b \in \{0,1\}$.
Output: A bit predicting x_i from x_{-i}.

1. Let $\mathbf{W}_0^0 \in \{0,1\}^n$ be x_{-i} with 0 in the i-th coordinate, and likewise let $\mathbf{W}_0^1$ be x_{-i} with 1 in the i-th coordinate.
2. For each $t \in [T]$, let $\mathbf{v}_t \in \mathbb{F}_2^n$ be $\mathbf{e}_{z_t}$.
3. Let fix $= 0$, $b_0 = 0$. Set T to be as in Theorem 7.7 for failure probability $\gamma/16$, so $T = O(k \log(1/\gamma)/\gamma^3)$
4. For $t = 0, \ldots, T-1$,
   (a) If $z_t = i$ and $b_t = 1$, let fix $= 1 +$ fix $\pmod 2$.
   (b) If $\mathbf{W}_t^0 \in D$ and $\mathbf{W}_t^1 \in D$, continue to the next iteration.
   (c) If $\mathbf{W}_t^0 \in D$ and $\mathbf{W}_t^1 \notin D$, then return $0 +$ fix $\pmod 2$.
   (d) If $\mathbf{W}_t^1 \in D$ and $\mathbf{W}_t^0 \notin D$, then return $1 +$ fix $\pmod 2$.
   (e) If $\mathbf{W}_t^0 \notin D$ and $\mathbf{W}_t^1 \notin D$, set $\mathbf{W}_t^0 = \mathbf{W}_{t-1}^0$ and $\mathbf{W}_t^1 = \mathbf{W}_{t-1}^1$.
   (f) Set $\mathbf{W}_{t+1}^0 = \mathbf{W}_t^0 + b_{t+1} \cdot \mathbf{v}_{t+1}$, and $\mathbf{W}_{t+1}^1 = \mathbf{W}_t^1 + b_{t+1} \cdot \mathbf{v}_{t+1}$.
5. Return b.

Proof (of Theorem 7.10): Fix any $x \in D$. We'll show that SeqPredWithRestarts predicts x_i from x_{-i} with high probability over i and R. Let $\mathbf{W}_t^*(i, R) \in \{0,1\}^n$ be the location of the walk that started at x (rather than $x + e_i$) at iteration t. (This can be undefined if t is after the algorithm terminated, and it can depend on i because of the termination conditions.) Next, define the sequence $\mathbf{W}_0, \ldots, \mathbf{W}_T$ (which depends only on R) as the path of the walk starting at $\mathbf{W}_0 = x$, if the algorithm had ignored the termination conditions in Item 4c and Item 4d. That is, regardless of the choice of i,

$$\mathbf{W}_t(R) = \begin{cases} \mathbf{W}_{t-1}(R) + \mathbf{v}_t & \text{if } \mathbf{W}_{t-1} + \mathbf{v}_t \in D, \\ \mathbf{W}_{t-1}(R), & \text{otherwise.} \end{cases}$$

One should notice that for any choice of i, R, and any t before the algorithm terminates for this choice of i and R, we have $\mathbf{W}_t(R) = \mathbf{W}_t^*(i, R)$.

Call R *good* if for some t,

$$\mathbf{W}_t(R) \in S = \left\{ x' \in D : \Pr_{i \sim [n]} [D(x' + e_i) = 0] \geq \gamma/2 \right\}.$$

In words, the walk using R eventually reaches S (as in Definition 7.2) in T steps. The probability that R does not satisfy this condition is $\gamma/16$, by Theorem 7.7.

Thus, for any such good R, if t^* is the first t such that $\mathbf{W}_{t^*} \in S$, then the algorithm will correctly predict as long as the choice of i did not cause the algorithm to terminate before step t. That is, $\mathbf{W}_{t^*}^*$ is not undefined. There are at most $\beta T n$ such choices of i. This is because whenever W_t is just outside D, the outside insensitivity property guarantees that there are at most βn such choices of i.

On the other hand, there are at least $\gamma n/2$ choices of i that cause the algorithm to terminate with a correct prediction on step t^*. Thus, there are at least

$$\alpha \geq \gamma/2 - \beta T$$

fraction of "good" i-s (for a particular good R). As before, call i *very bad* for R if the algorithm terminates with an incorrect prediction. And here too, there can only be at most $\beta T n$ very bad i's. Call i *bad* otherwise (for such an i, the algorithm terminates with a random bit): we calculate the final probability of being correct given a fixed good R as (where we use P as shorthand for our predictor SeqPredWithRestarts):

$$\begin{aligned}
\Pr_i[P \text{ is correct}] &= \Pr_i[P \text{ is correct} | i \text{ is good}] \Pr[i \text{ is good}] \\
&\quad + \Pr_i[P \text{ is correct} | i \text{ is bad}] \Pr[i \text{ is bad}] \\
&\quad + \Pr_i[P \text{ is correct} | i \text{ is very bad}] \Pr[i \text{ is very bad}] \\
&= \alpha + \frac{1}{2}(1 - \alpha - \beta \cdot T) + 0 \\
&\geq \frac{1}{2} + \frac{\gamma}{4} - \beta T.
\end{aligned}$$

Since the probability that R is good is at least $1 - \gamma/16$, the final probability of a correct prediction over R and i is at least

$$\frac{1}{2} + \frac{\gamma}{8} - \beta T.$$

■

Once again, a basic argument about outside insensitivity extends the result to two-sided predictability.

Theorem 7.11. *Let $D\colon \{0,1\}^n \to \{0,1\}$ be a β-insensitive distinguisher with $|D| = K = 2^k$, and let $\gamma = 1 - \frac{k}{n}$, as in Theorem 7.1. Then, D is*

$$\left(\frac{\gamma}{8} - \beta \cdot T, \beta\right)$$

predictable with $O(T)$ queries and runtime $O(n)$, where $T = O(k \log(1/\gamma)/\gamma^3)$.

Proof: Whenever $x \notin D$, for at least $(1-\beta)n$ choices of i, the algorithm repeatedly falls in the branch Item 4e, for all T steps of the walk, after which it will output a random guess. ■

8 Weak Unpredictability from Worst-Case Hardness with Minimal Loss in Hardness

We show how to construct a *weakly unpredictable* source from standard worst-case hardness assumptions, with minimal loss in the hardness parameter. That is, we show how to transform a function with a truth table of length n that is hard against circuits of size $s = n^{1-\eta}$ into pseudorandom bits $X_1, \ldots, X_m$ (where $m \approx s$) that are (weakly) unpredictable for circuits of size $n^{1-2\eta}$. The construction uses (by now) standard ingredients and techniques, but we present it here in a self-contained manner, since our parameter regime is not commonly considered.

8.1 Worst Case Hardness to Mild Average Case Hardness with Minimal Hardness Loss

The first ingredient is converting a worst-case hard function $f \in \{0,1\}^n$ that is hard for circuits of size $n^{1-\eta}$ into a function $\bar{f} \in \{0,1\}^{\bar{n}}$ which is *mildly* average case hard for circuits of size $n^{1-O(\eta)}$. This uses a standard result from [STV01].

Theorem 8.1 ([STV01], Theorem 24.). *For any $n \in \mathbb{N}$ and $\varepsilon > 0$, there exists a code $C\colon \{0,1\}^n \to \{0,1\}^{\bar{n}=\mathrm{poly}(n,1/\varepsilon)}$ such that the following holds. Let $f \in \{0,1\}^n$ be the truth table of a function that is worst case hard for circuits of size s. Then, $\bar{f} = C(f)$ is $\frac{1}{2} - \varepsilon$ average case hard for circuits of size $s' = (\varepsilon/\log n)^c \cdot s$ for some universal constant c.*

Remark 8.2. For constant $\varepsilon > 0$, we note that given $f \in \{0,1\}^n$ as a string, it is possible to compute any coordinate of the code $C(f)_i$, $i \in [\bar{n}]$, in time $\widetilde{O}(n)$. In fact, evaluating the code on m coordinates is possible in time $(m+n)^{1+o(1)}$. Indeed, the code in Theorem 8.1 consists of a $(t = O(\log n))$-variate Reed-Muller code of degree at most $\log n$ in each variable over a field $\mathbb{F}$ of size $O(\log^2 n)$, concatenated with a Hadamard code. Thus, evaluating $C(f)_i$ requires interpreting $f \in \{0,1\}^n$ as (at most n) values in $\mathbb{F}$, interpolating those values into a polynomial, and evaluating the polynomial on a given point. This requires time $\widetilde{O}(n)$.

Finally, computing the Hadamard code simply requires computing an inner product on a vector of length $O(\log\log n)$. In order to compute $C(f)$ on m coordinates, after interpolating the polynomial, one can use batch evaluation techniques to compute the Reed-Muller code on all m points in time $(m+n)^{1+o(1)}$. For more details on interpolation, we refer the reader to the discussion in [DMOZ22, Theorem 4.6]. For more details on batch evaluation, we refer the reader to [BGG+24]. Finally, we note that while there are newer constructions of locally list decodable codes (say, the recent [DHIP26]), they do not quantitatively improve our results, as crucially, the number of queries of such codes is $\mathrm{poly}(1/\varepsilon)$, (and cannot be smaller than $1/\varepsilon^2$).

We record the following corollary, setting $s = n^{1-\eta}$, while aiming for small unpredictability $n^{-\theta}$.

Corollary 8.3. *For every sufficiently large $n \in \mathbb{N}$ the following holds. Let $\eta > 0$ be a sufficiently small constant and $\theta \le \eta/(2c)$, where c is from Theorem 8.1. Set $\varepsilon = n^{-\theta}$. Then, there exists a code $C\colon \{0,1\}^n \to \{0,1\}^{\bar{n}=\mathrm{poly}(n,1/\varepsilon)}$ such that the following holds.*

Let $f \in \{0,1\}^n$ be the truth table of a function that is worst case hard for circuits of size $s = n^{1-\eta}$. Then, $\bar{f} = C(f)$ is $\frac{1}{2} - \varepsilon$ average case hard for circuits of size $s' = n^{1-2\eta}$. Moreover, given $f \in \{0,1\}^n$ as a string, $C(f)$ can be computed on $m = n^{1-\eta}$ coordinates in time $m^{1+\eta}$.

The upshot of this corollary is that it allows us to output m bits of "pseudoentropy", even though the "unpredictability" is only $\varepsilon = n^{-\theta} \gg 1/m$, which is too large for the hybrid argument. Moreover, because $\varepsilon = n^{-\theta}$ for a small θ, the *hardness loss* is very small, namely $s/s' = n^{\eta}$.

8.2 Mild Average Case Hardness to Unpredictability Using Designs

So far, we have established that the loss from s to s' can be small for unpredictability $\varepsilon = n^{-\theta}$. We now use the average case hard function from Corollary 8.3 in the Nisan–Wigderson framework of constructing pseudorandom generators [NW94]. First, we use standard results about designs (see, e.g., [Vad12, Chapter 7]).

Definition 8.4. *A family of sets $S_1, \ldots, S_m \subseteq [d]$ is an (ℓ, a)-design if:*

- $\forall i, |S_i| = \ell$, *and,*
- $\forall i \neq j, |S_i \cap S_j| \leq a$.

Theorem 8.5. *For any constant $\gamma > 0$, and any $\ell, m \in \mathbb{N}$, there exists an (ℓ, a)-design $S_1, \ldots, S_m \subseteq [d]$ with $d = O(\frac{\ell^2}{a})$ and $a = \gamma \log m$. Such a design can be constructed deterministically in time $m \cdot \mathrm{poly}(d)$.*

We will use the following standard theorem about designs and the NW generator.

Theorem 8.6 ([NW94], see also [Vad12], Theorem 7.24)**.** *Let $G\colon \{0,1\}^d \to \{0,1\}^m$ be the Nisan-Wigderson generator based on a function $f\colon \{0,1\}^\ell \to \{0,1\}$ and an (ℓ, a)-design. For any $\varepsilon > 0$, if f is $1/2 - \varepsilon$ average-case hard for circuits of size s, then G is ε-unpredictable for circuits of size $s - m \cdot a \cdot 2^a$.*

Theorem 8.7. *For a sufficiently small constant $\eta > 0$, $\theta \leq \eta/(2c)$ where c is as in Theorem 8.1, the following holds. Suppose that for every $n \in \mathbb{N}$, there exists a function $f \in \{0,1\}^n$ that is printable in time $n^{1+\eta}$ and that is worst case hard for circuits of size $n^{1-\eta}$. Then, for all sufficiently large m, there exists a generator*

$$G\colon \{0,1\}^{d=O(\log m)} \to \{0,1\}^m$$

such that $G(Y) = X_1, \ldots, X_m$ is $(\varepsilon = m^{-\theta})$-symmetrically unpredictable for circuits of size $s = n^{1-3\eta} \geq m^{1+\eta}$. Moreover, G can be computed in time $m^{1+35\eta}$.

Proof: First, we observe that since the definition of designs does not specify an order for the the sets S_i, the result in Theorem 8.6 actually shows that G is ε-symmetrically unpredictable.

Now, choose n so that $m = n^{1-c'\eta}$ for $c' = 7$, and consider the hard function $f \in \{0,1\}^n$. Let $\bar{f}$ be the average case hard function obtained in Corollary 8.3. We know $\bar{f}$ is of length $\mathrm{poly}(n)$ and is $\frac{1}{2} - \varepsilon'$ average case hard for circuits of size $n^{1-2\eta}$, with $\varepsilon' = n^{-\theta}$. Following Theorem 8.5 and Theorem 8.6, using designs with $\ell = \log \bar{n} = \bar{c} \cdot \log n$, $\gamma = \eta$, and $m = n^{1-c'\eta}$, we get

$$G\colon \{0,1\}^{d=O(\log n)} \to \{0,1\}^m$$

that is ε'-symmetrically unpredictable for circuits of size

$$n^{1-2\eta} - n^{1-c'\eta} \cdot \eta(1-c'\eta)\log n \cdot 2^{\eta(1-c'\eta)\log n} \geq n^{1-2\eta} - n^{1-(c'-3)\eta} \geq n^{1-3\eta} \geq m^{1+\eta}$$

for a large enough n, where we took $a = \gamma \log m = \eta(1-c'\eta)\log n$ (see Theorem 8.5). One can verify using the fact that $n \geq m$ that $\varepsilon' \leq m^{-\theta}$. For the runtime of G, recall that it takes $n^{1+\eta}$ time to print f. Constructing the design takes time $\widetilde{O}(m)$, and finally evaluating $\bar{f}$ on m points takes time $m^{1+\eta}$. Overall, the runtime is $O(n^{1+\eta}) = O(m^{\frac{1+\eta}{1-c'\eta}}) = O(m^{1+5c'\eta})$. ∎

8.3 Seed Length Nearly $1 \cdot \log m$

We now demonstrate how to reduce the seed length in Theorem 8.7 from $O(\log m)$ to nearly $1 \cdot \log m$, using the composition idea of Chen and Tell [CT21]. To do so, we utilize their setting of parameters in the NW generator, stretching $1 \cdot \log m$ bits of seed to m^η bits of pseudorandomness.

Theorem 8.8 ([CT21], Theorem 4.1)**.** *For a sufficiently small constant $\eta > 0$, suppose that there exists a function $f \in \{0,1\}^n$ printable in time $O(n^{1+\eta})$, that is worst case hard for circuits of size $n^{1-\eta}$. Then, there exists an $n^{-\eta}$-pseudorandom generator*

$$G\colon \{0,1\}^{(1+O(\eta))\log n} \to \{0,1\}^{n^\eta}$$

against circuits of size $n^{1-O(\eta)}$. Computing the generator can be done in time $n^{1+O(\eta)}$.

We can compose our symmetric unpredictability generator with this generator to get $(1 + O(\eta)) \log m$ seed length. (We make no attempt to optimize the constants.)

Theorem 8.9. *For sufficiently small constant $\eta > 0$, $\theta \leq \eta/(2c)$ where c is as in Theorem 8.1, the following holds. Suppose that for every $n \in \mathbb{N}$, there exists a function $f \in \{0,1\}^n$ that is printable in time $n^{1+\eta}$ and that is worst case hard for circuits of size $n^{1-\eta}$. Then, for all sufficiently large m, there exists a generator*

$$G\colon \{0,1\}^{d=(1+O(\eta))\log m} \to \{0,1\}^m$$

such that $G(Y) = X_1, \ldots, X_m$ is $(\varepsilon = m^{-\theta})$-symmetrically unpredictable for circuits of size $s = m^{1+\eta}$. Moreover, G can be computed in time $m^{1+O(\eta)}$, and its support can be computed in time $m^{2+O(\eta)}$.

Proof: Let $G_2\colon \{0,1\}^{d_2=O(\log m)} \to \{0,1\}^m$ be the generator given by Theorem 8.7 whose output is $\frac{m^{-\theta}}{2}$-symmetrically unpredictable for circuits of size $m^{1+\eta}$. For a proper choice of n in Theorem 8.8, we can get, for some constant c',

$$G_1\colon \{0,1\}^{d_1=(1+c'\cdot\eta)\log m} \to \{0,1\}^{d_2=\omega(\log m)}$$

that is an $n^{-\eta}$-PRG against circuits of size $m^{1+c''\eta}$ for $c'' = 40$. Note the choice of n should be such that $n^{1-O(\eta)} = m^{1+c''\eta}$. Note that this also implies that $n^{-\eta} \leq \frac{m^{-\theta}}{2}$.

Our generator will be $G = G_2 \circ G_1$. We show that its output is ε-symmetrically unpredictable.[9] Fix an $i \in [m]$, and any predicting circuit $P\colon \{0,1\}^{m-1} \to \{0,1\}$ of size $m^{1+\eta}$. We know that

$$\Pr[P(G_2(U_{d_2})_{-i}) = G_2(U_{d_2})_i] \leq \frac{1}{2} + \frac{m^{-\theta}}{2}.$$

Consider the circuit $C\colon \{0,1\}^{d_2} \to \{0,1\}$ that computes $G_2(x)$, and $P(G_2(x)_{-i})$, and accepts if only if $P(G_2(x)_{-i}) = G_2(x)_i$. By Theorem 8.7, such a circuit has size $\widetilde{O}(m^{1+\eta} + m^{1+35\eta}) = o(m^{1+c''\eta})$. However, since G_1 is an $n^{-\eta}$-PRG for circuits of such size, we have that for sufficiently large n (and m):

$$|\Pr[C(G_1(U_{d_1})) = 1] - \Pr[C(U_{d_2}) = 1]| \leq \frac{m^{-\theta}}{2}.$$

Therefore,

$$\Pr[P(G_2(G_1(U_{d_1}))_{-i}) = G_2(G_1(U_{d_1}))_i] \leq \frac{1}{2} + m^{-\theta}.$$

For runtime, we note that both G_1 and G_2 will run in time $m^{1+O(\eta)}$ (including printing the respective f-s used in each one). Computing on all seeds therefore takes $m^{2+O(\eta)}$ time. ■

[9] We truncate the output of G_1 to fit the input length of G_2. The truncated version is still an $n^{-\eta}$-PRG for circuits of size $m^{1+O(\eta)}$

9 Hitting Set Generators and Evasive Pseudoentropy Against Insensitive Distinguishers

In this section, we utilize the constructions of Section 8, together with the prediction algorithms in Section 6 and Section 7, to construct efficient hitting set generators and evasive pseudoentropy generators. From Theorem 8.9, we can obtain m output bits of symmetric $m^{-\theta}$-unpredictability for any small constant θ with minimal loss in hardness when starting from the fine-grained assumption (e.g. $f \in \{0,1\}^n$ is hard for circuits of size $n^{1-\gamma}$). We will focus on unpredictable distributions obtained in this way. In particular, our efficient hitting generators in this regime immediately implies a super-fast derandomization of one-sided error randomized algorithms that have insensitive distinguishers.

9.1 Hitting Set Generators for (α, β)-insensitive Distinguishers

We now utilize Theorem 6.8 to show that an ε-symmetrically unpredictable X suffices as a hitting set generator for (α, β)-insensitive distinguishers.

Theorem 9.1 (hitting sets for (α, β)-insensitive distinguishers from unpredictability)**.** *For any sufficiently small constants $\eta, \theta > 0$, suppose $X = X_1, \ldots, X_m$ is $(\varepsilon = m^{-\theta})$-symmetrically unpredictable against circuits of size $m^{1+\eta}$.*

Then, X is a 2^{k-m}-hitting set generator for (α, β)-insensitive distinguishers $D\colon \{0,1\}^m \to \{0,1\}$ of size $|D| \leq K$ for circuits of size m, whenever $\frac{\alpha}{2} = \beta \log m - |D| 2^{-m} > m^{-\theta}$. In particular:

- *X is a 2^{k-m}-hitting set generator for (α, β)-insensitive distinguishers $D\colon \{0,1\}^m \to \{0,1\}$ of size $|D| \leq K$ for circuits of size m, as long as $K = 2^k \leq \frac{1}{2} m^{-\theta} \cdot 2^m$, $\alpha > 4m^{-\theta}$ and $\beta \leq m^{-2\theta}$.*
- *For any constant α, X is a 2^{k-m}-hitting set generator for (α, β)-insensitive distinguishers $D\colon \{0,1\}^m \to \{0,1\}$ of size $|D| \leq K$ for circuits of size m, as long as $\beta \leq \frac{\alpha}{8 \log m}$, and $K \leq \frac{\alpha}{8} \cdot 2^m$.*

Proof: Suppose towards a contradiction that $\mathrm{Supp}(X) \subseteq D$, for some (α, β)-insensitive distinguisher D of circuit size m. By Theorem 6.8, there is a predictor P that makes $O(\log m)$ queries to D, and runs in time $O(m)$, that predicts with advantage at least $\frac{\alpha}{2} - \beta \log m - |D| \cdot 2^{-m} \geq m^{-\theta}$. Namely for every $x \in \mathrm{Supp}(X)$,

$$\Pr_{i,R}[P(i, x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n, R) = x_i] > \frac{1}{2} + m^{-\theta},$$

and so certainly,

$$\Pr_{X,i,R}[P(i, X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_n, R) = X_i] > \frac{1}{2} + m^{-\theta}.$$

Thus, there exists a fixing $(i^\star, r)$ to i and R for which

$$\Pr_{X}[P(i^\star, X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_n, r) = X_i] > \frac{1}{2} + m^{-\theta}.$$

Since $|r| = O(m)$ and $|i^\star| = O(\log m)$, the predictor $P(i^\star, \cdot, r)$ can be implemented by a circuit of size $\widetilde{O}(m)$ and thus contradicts the unpredictability of X. ∎

Combining with the generator in Theorem 8.9, we get the following corollary.

Corollary 9.2. *Let $\eta > 0$ be a sufficiently small constant, and let $\theta \leq \eta/(2c)$ for some universal constant c. Suppose that for every $n \in \mathbb{N}$, there exists a function $f \in \{0,1\}^n$ that is printable in time $n^{1+\eta}$ and that is worst case hard for circuits of size $n^{1-\eta}$. Then, for all sufficiently large $m \in \mathbb{N}$, there exists a*

$$G\colon \{0,1\}^{d=(1+O(\eta))\log m} \to \{0,1\}^m$$

that is a 2^{k-m}-hitting set generator for (α,β)-insensitive distinguishers $D\colon \{0,1\}^m \to \{0,1\}$ of size $|D| \leq K$ for circuits of size m, whenever $\frac{\alpha}{2} - \beta \log m - |D|2^{-m} > m^{-\theta}$. In particular:

- *G is a 2^{k-m}-hitting set generator for (α,β)-insensitive distinguishers $D\colon \{0,1\}^m \to \{0,1\}$ of size $|D| \leq K$ for circuits of size m, as long as $K = 2^k \leq \frac{1}{2}m^{-\theta} \cdot 2^m$, $\alpha > 4m^{-\theta}$ and $\beta \leq m^{-2\theta}$.*
- *For any constant α, G is a 2^{k-m}-hitting set generator for (α,β)-insensitive distinguishers $D\colon \{0,1\}^m \to \{0,1\}$ of size $|D| \leq K$ for circuits of size m, as long as $\beta \leq \frac{\alpha}{8\log m}$, and $K \leq \frac{\alpha}{8} \cdot 2^m$.*

Moreover, G can be computed in time $m^{1+O(\eta)}$, and its support can be computed in time $m^{2+O(\eta)}$.

9.2 Hitting Set Generators for β-insensitive Distinguishers

Next, we use Theorem 7.10 to show that an ε-symmetrically unpredictable X suffices as a hitting set generator for β-insensitive distinguishers. In this scenario, we'll state our results to highlight the fact that the size loss is linear in $\log|D|$ (in contrast with the standard hybrid argument where the size loss is polynomial in $\log|D|$). As Theorem 7.10 requires $\beta < 1/\log|D|$ in order for the prediction advantage to be meaningful, we require, say, $|D| \leq 2^{m^{0.98}}$, so that $\beta \leq n^{-0.99}$ (recall that it is impossible to have $\beta < 1/n$).

Theorem 9.3 (hitting sets for β-insensitive distinguishers from unpredictability)**.** *Suppose $X = X_1,\ldots,X_m$ is $\frac{1}{16}$-symmetrically unpredictable against circuits of size $s = \omega(m\log m)$.*

Then, X is a ε-hitting set generator for β-insensitive distinguishers $D\colon \{0,1\}^m \to \{0,1\}$ of size $|D| \leq K = 2^k$ for circuits of size $s' = s/(k\log m)$, where $\varepsilon = K \cdot 2^{-m}$, whenever there exists some constant $\zeta \in (0,1)$ such that $k \leq m^{1-\zeta}$ and $\beta \leq k^{-(1+\zeta)}$.

Proof: Suppose not (for sufficiently large m). Then there is a D of size s' such that $\mathrm{Supp}(X) \subseteq D$. By Theorem 7.10, there is a predictor P (taking random i, X, R as inputs) such that for some constant C, we have that D is

$$\frac{1-k/m}{8} - 2C\beta\frac{k}{(1-k/m)^4} \geq \frac{1}{16}$$

predictable with $C\frac{k}{(1-k/m)^4} = O(k)$ queries to D, and runtime $O(m)$. Once again, there exists a fixing to i and R such that:

$$\Pr_X[P(i,X_1,\ldots,X_{i-1},X_{i+1},\ldots,X_n,R) = X_i] \geq \frac{1}{2} + \frac{1}{16}.$$

Overall, the size of this circuit is $O(m\log m + k \cdot s/(k\log m)) < s$, so this is a contradiction to the symmetric unpredictability of X. ∎

We can combine the above result with Theorem 8.9 to get the following corollary.

Corollary 9.4. *For any sufficiently small constant $\eta > 0$, the following holds. Suppose that for every $n \in \mathbb{N}$, there exists a function $f \in \{0,1\}^n$ that is printable in time $n^{1+\eta}$ and that is worst case hard for circuits of size $n^{1-\eta}$. Then, for all sufficiently large $m \in \mathbb{N}$, any constant $\zeta > 0$ and any $k \leq m^{1-\zeta}$, there exists a generator*

$$G\colon \{0,1\}^{d=(1+O(\eta))(\log m + \log k)} \to \{0,1\}^m$$

such that G is an ε-hitting set generator for β-insensitive distinguishers of size $K = 2^k$ for circuit size m, where $\varepsilon = K \cdot 2^{-m}$, whenever $\beta \leq k^{-(1+\zeta)}$. Moreover G can be computed in time $(mk)^{1+O(\eta)}$ on one input and $(mk)^{2+O(\eta)}$ on all inputs.

Proof: We instantiate Theorem 8.9 to get a generator

$$G\colon \{0,1\}^{(1+O(\eta))\log m'} \to \{0,1\}^{m'}$$

for $m' = mk \log m$ computable in time $m'^{1+O(\eta)} \leq (mk)^{1+O(\eta)}$ on one input and $(mk)^{2+O(\eta)}$ on all inputs. Truncating such a generator to m bits yields an $(1 + O(\eta))(\log m + \log k)$ seed length $1/16$-symmetric unpredictability generator for circuits of size $(m')^{1+\eta}$ (notice also that $(m')^{1+\eta} = \omega(m \log m)$). To see that truncating does not affect unpredictability, observe that for any of the first m coordinates i, it is unpredictable given all $m' - 1$ other bits, and is thus also unpredictable with the same parameters given only the other $m - 1$ first bits. Thus, G is a $K \cdot 2^{-m}$ hitting set generator for β-insensitive distinguishers by Theorem 9.3. ■

We point out that the dependence of the seed length of the PRG on k is nearly $1 \cdot \log k$. On the other hand, the standard techniques to obtain a similar result for $|D| \leq K$ using the truncated hybrid argument Section 5, would require roughly $1/k$ unpredictability, and thus $O(\log k)$ seed length.

9.3 Evasive Pseudoentropy

Next, we'll restate the theorems using the two-sided predictability results in Theorem 6.12 and Theorem 7.11.

Theorem 9.5 (evasive pseudoentropy for (α, β)-insensitive distinguishers)**.** *For any sufficiently small constants $\eta, \theta > 0$, suppose $X = X_1, \ldots, X_m$ is $(\varepsilon = m^{-2\theta})$-symmetrically unpredictable against circuits of size $m^{1+\eta}$.*

Then, X is a (k, ε')-evasive pseudoentropy generator for (α, β)-insensitive distinguishers $D\colon \{0,1\}^m \to \{0,1\}$ of size $K = 2^k$ with circuit size m, as long as $\frac{\alpha}{2} \cdot \varepsilon' - \beta \log m - 2^{k-m} > \varepsilon$. In particular:

- *X is a $(k, m^{-\theta})$-evasive pseudoentropy generator for (α, β)-insensitive distinguishers $D\colon \{0,1\}^m \to \{0,1\}$ of size $K = 2^k$ with circuit size m, as long as $K \leq \frac{1}{2} m^{-\theta} \cdot 2^m$, $\alpha > 4m^{-\theta}$ and $\beta \leq m^{-2\theta}$.*
- *For any constant ε', X is a (k, ε')-evasive pseudoentropy generator for (α, β)-insensitive distinguishers $D\colon \{0,1\}^m \to \{0,1\}$ of size $K = 2^k$ with circuit size m, as long as α is any constant, $\beta \leq \frac{\alpha\varepsilon'}{8 \log m}$, and $K \leq \frac{\alpha\varepsilon'}{8} \cdot 2^m$.*

Proof: Assume otherwise, that there exists an (α, β)-insensitive distinguisher $D\colon \{0,1\}^m \to \{0,1\}$ for circuits of size m such that

$$\Pr[X \in D] > \frac{|D|}{2^k} + \varepsilon'.$$

Assume that $|D| < 2^k$ (because otherwise we would have an immediate contradiction). By Theorem 6.12, since D is $\left(\frac{\alpha}{2} - \beta \log m - |D| \cdot 2^{-m}, \beta \log m\right)$-two-sided predictable with $O(\log m)$ queries to D and runtime $O(m)$, we have a predicting algorithm P that runs in time $O(m)$ such that:

$$\begin{aligned}
&\Pr_{X,i,R}[P(i, X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_n, R) = X_i] \\
&= \Pr_{X,i,R}[P(i, X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_n, R) = X_i | X \in D] \Pr[X \in D] \\
&\quad + \Pr_{X,i,R}[P(i, X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_n, R) = X_i | X \notin D] \Pr[X \notin D] \\
&\geq \varepsilon' \cdot \left(\frac{1}{2} + \frac{\alpha}{2} - \beta \log m - |D| \cdot 2^{-m}\right) + (1 - \varepsilon') \cdot \left(\frac{1}{2} - \beta \log m\right) \\
&\geq \frac{1}{2} + \frac{\alpha}{2} \cdot \varepsilon' - \beta \log m - K \cdot 2^{-m} \\
&> \frac{1}{2} + m^{-2\theta}.
\end{aligned}$$

Thus, by hardwiring R and i, there is a predicting circuit of size $\widetilde{O}(m)$ contradicting the symmetric unpredictability of X. ∎

Corollary 9.6. *Let $\eta > 0$ be a sufficiently small constant, and let $\theta \leq \eta/(2c)$for some universal constant c. Suppose that for every $n \in \mathbb{N}$, there exists a function $f \in \{0,1\}^n$ that is printable in time $n^{1+\eta}$ and that is worst case hard for circuits of size $n^{1-\eta}$. Then, for all sufficiently large $m \in \mathbb{N}$, there exists a*

$$G\colon \{0,1\}^{d=(1+O(\eta))\log m} \to \{0,1\}^m$$

that is a (k, ε')-evasive pseudoentropy generator for (α, β)-insensitive distinguishers $D\colon \{0,1\}^m \to \{0,1\}$ of size $K = 2^k$ with circuit size m, as long as $\frac{\alpha}{2} \cdot \varepsilon' - \beta \log m - 2^{k-m} > m^{-2\theta}$. In particular:

- *G is a $(k, m^{-\theta})$-evasive pseudoentropy generator for (α, β)-insensitive distinguishers $D\colon \{0,1\}^m \to \{0,1\}$ of size $K = 2^k$ with circuit size m, as long as $K \leq \frac{1}{2} m^{-\theta} \cdot 2^m$, $\alpha > 4m^{-\theta}$ and $\beta \leq m^{-2\theta}$.*
- *For any constant ε', G is a (k, ε')-evasive pseudoentropy generator for (α, β)-insensitive distinguishers $D\colon \{0,1\}^m \to \{0,1\}$ of size $K = 2^k$ with circuit size m, as long as α is any constant, $\beta \leq \frac{\alpha\varepsilon'}{8 \log m}$, and $K \leq \frac{\alpha\varepsilon'}{8} \cdot 2^m$.*

Moreover, G can be computed in time $m^{1+O(\eta)}$, and its support can be computed in time $m^{2+O(\eta)}$.

Theorem 9.7 (evasive pseudoentropy for β-distinguishers from unpredictability)**.** *Suppose that $X = X_1, \ldots, X_m$ is $(\varepsilon = m^{-2\theta})$-symmetrically unpredictable against circuits of size $s = \omega(m \log m)$, where $\theta \in (0,1)$ is any constant.*

Then, X is a $(k, m^{-\theta})$-evasive pseudoentropy generator for β-insensitive distinguishers $D\colon \{0,1\}^m \to \{0,1\}$ of size $|D| = K = 2^k$ for circuits of size $s' = s/(k \log m)$, whenever $k \leq m^{1-\zeta}$ and $\beta \leq k^{-(1+\zeta)}$ for some constant $\zeta \leq \sqrt{1 - 2\theta}$.

Proof: Once again, suppose not, and there is a β-insensitive distinguisher D such that:

$$\Pr[X \in D] > \frac{|D|}{2^k} + m^{-\theta}.$$

By Theorem 7.11, and using the same argument as in Theorem 9.5, since D is $(\alpha = 1/16, \beta)$-predictable with $O(k)$ queries, and runtime $O(m)$. We get for some predictor P of size smaller than s,

$$\begin{aligned}
&\Pr_{X,i,R}[P(i, X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_n, R) = X_i] \\
&\quad = \Pr_{X,i,R}[P(i, X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_n, R) = X_i | X \in D] \Pr[X \in D] \\
&\qquad + \Pr_{X,i,R}[P(i, X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_n, R) = X_i | X \notin D] \Pr[X \notin D] \\
&\quad \geq m^{-\theta} \cdot \left(\frac{1}{2} + \alpha\right) + (1 - m^{-\theta}) \cdot \left(\frac{1}{2} - \beta\right) \\
&\quad \geq \frac{1}{2} + \frac{m^{-\theta}}{16} - \beta \geq \frac{1}{2} + m^{-2\theta}.
\end{aligned}$$

The last inequality follows from our constraints. Indeed, $\beta \leq k^{-(1+\zeta)} \leq m^{-(1-\zeta^2)} \leq m^{-2\theta}$. ■

Corollary 9.8. *There exists a universal constant $c > 0$ such that for every sufficiently small constant $\eta > 0$, the following holds. Suppose that for every $n \in \mathbb{N}$, there exists a function $f \in \{0,1\}^n$ that is printable in time $n^{1+\eta}$ and that is worst case hard for circuits of size $n^{1-\eta}$.*

Then, for all sufficiently large $m \in \mathbb{N}$, any constant $0 < \zeta \leq \sqrt{1 - \eta/(4c)}$, and any $k \leq m^{1-\zeta}$ and $\beta \leq k^{-(1+\zeta)}$, there exists a generator

$$G\colon \{0,1\}^{d=(1+O(\eta))(\log m + \log k)} \to \{0,1\}^m$$

such that G is a $(k, m^{-\eta/(4c)})$-evasive pseudoentropy generator for β-insensitive distinguishers for circuit size m. Moreover, G can be computed in time $(mk)^{1+O(\eta)}$, and its support can be computed in time $(mk)^{2+O(\eta)}$.

Proof: We instantiate Theorem 8.9 to get a generator

$$G\colon \{0,1\}^{(1+O(\eta))\log m'} \to \{0,1\}^{m'}$$

for $m' = mk \log m$. Truncating such a generator to m bits yields an $(1 + O(\eta))(\log m + \log k)$ seed length $(m^{-\eta/(2c)})$-symmetric unpredictability generator for circuits of size $(m')^{1+\eta}$ (notice also that $(m')^{1+\eta} = \omega(m \log m)$). Thus, G is a $(k, m^{-\eta/(4c)})$ hitting set generator for β-insensitive distinguishers by Theorem 9.7. ■

Notice that compared to Theorem 9.5, for distinguishers with only outside sensitivity, we only achieve pseudoentropy $m^{0.99}$ rather than nearly m.

9.4 Derandomization of Insensitive Algorithms

Finally, we can apply our hitting set generators in order to get new derandomization results on insensitive algorithms.

Definition 9.9 (insensitive algorithm)**.** *Let $A\colon \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$ be a probabilistic algorithm running in time m. We say that A is (α, β)-insensitive with respect to a language L, if for any x, the following two properties hold.*

1. *For any r such that $A(x,r) \neq L(x)$ and there exists and i such that $A(x, r+\mathbf{e}_i) = L(x)$, we have:*

$$\Pr_i[A(x, r+\mathbf{e}_i) = L(x)] \geq \alpha$$

2. *For any r such that $A(x,r) = L(x)$, we have:*

$$\Pr_i[A(x, r+\mathbf{e}_i) \neq L(x)] \leq \beta$$

We say that A is β insensitive with respect to a language L, if for any x, we only require the second property to hold.

Theorem 9.10. *There exists a universal constant $c > 0$ such that for every sufficiently small constant $\eta > 0$, the following holds. Suppose that for every $n \in \mathbb{N}$, there exists a function $f \in \{0,1\}^n$ that is printable in time $n^{1+\eta}$ and that is worst case hard for circuits of size $n^{1-\eta}$.*

Let $\theta = \eta/(2c)$. Let $L \subset \{0,1\}^n$ be a language and let $A\colon \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$ be a probabilistic algorithm running in time m.

1. *(Derandomizing (α,β)-insensitive algorithms for **BPP** with polynomially small error) Suppose A is $(\alpha = 4m^{-\theta}, \beta = m^{-2\theta})$-insensitive with respect to L, and furthermore, for every x:*

$$\Pr_r[A(x,r) \neq L(x)] \leq \frac{m^{-\theta}}{8}.$$

Then there exists a deterministic algorithm $A'\colon \{0,1\}^n \to \{0,1\}$ that accepts L and runs in time $m^{2+O(\eta)}$.

2. *(Derandomizing (α,β)-insensitive algorithms for **BPP** with constant error) For any constant α, Suppose A is $(\alpha, \beta = \frac{\alpha}{64 \log m})$-insensitive with respect to L, and furthermore, for every x:*

$$\Pr_r[A(x,r) \neq L(x)] \leq \frac{\alpha}{512}.$$

Then there exists a deterministic algorithm $A'\colon \{0,1\}^n \to \{0,1\}$ that accepts L and runs in time $m^{2+O(\eta)}$.

3. *(Derandomizing β-insensitive algorithms for **RP** that err rarely) For some small constant ζ, suppose $k \leq m^{1-\zeta}$, and A is $\beta = k^{-(1+\zeta)}$-insensitive with respect to L. Suppose also that if $L(x) = 0$ then $A(x,r) = 0$ and if $L(x) = 1$, then*

$$\Pr_r[A(x,r) = 0] \leq 2^{k-m}.$$

Then there exists a deterministic algorithm $A'\colon \{0,1\}^n \to \{0,1\}$ that accepts L and runs in time $(mk)^{2+O(\eta)}$.

4. *(Derandomizing β-insensitive algorithms for **BPP** that err rarely) For some small constant $\zeta \leq \sqrt{1-\eta/(4c)}$, suppose $k \leq m^{1-\zeta}$, and A is $\beta = k^{-(1+\zeta)}$-insensitive with respect to L. Suppose also that for every x:*

$$\Pr_r[A(x,r) \neq L(x)] \leq \frac{2^{k-m}}{4}.$$

Then there exists a deterministic algorithm $A'\colon \{0,1\}^n \to \{0,1\}$ that accepts L and runs in time $(mk)^{2+O(\eta)}$.[10]

Proof: For the first result, A' first computes the generator of Theorem 9.5 on all seeds in time $m^{2+O(\eta)}$. It then runs $A(x,\cdot)$ and all $m^{1+O(\eta)}$ possible outputs of the generator, and takes the majority of all calls to A. This overall takes time $m^{2+O(\eta)} + m \cdot m^{1+O(\eta)} = m^{2+O(\eta)}$.

For correctness, fix any x and consider the circuit $\widetilde{A}\colon \{0,1\}^m \to \{0,1\}$ that implements $A(x,\cdot)$. The circuit has size $O(m \log m)$. Without loss of generality, we can assume that $\widetilde{A}$ outputs 1 whenever $A(x,r) \neq L(x)$ (since we can always hardwire a bit and negate if necessary). Since we use a pseudoentropy generator (with pseudoentropy k for $2^k = \frac{1}{2}m^{-\theta} \cdot 2^m$), we get that

$$\Pr[G(y) \in \widetilde{A}] \leq \frac{|D|}{2^k} + m^{-\theta} = \frac{1}{4} + m^{-\theta} < \frac{1}{2}.$$

Thus the majority of calls to A will be correct. For the second result, we can instead use the fact that it's a pseudoentropy generator with error $\varepsilon' = 1/8$, and k for $2^k = \frac{\alpha}{64}2^m$ to get $\Pr[G(y) \in \widetilde{A}] \leq \frac{|D|}{2^k} + \frac{1}{8} = \frac{1}{4} + \frac{1}{8} < \frac{1}{2}$.

For the last two bullets, similar arguments hold, using Corollary 9.4 and Corollary 9.8 instead. ■

10 Derandomizing Scorers

In this section, we use the generator of Theorem 8.9 to fool scoring algorithms. We first recall the definition of a scorer.

Definition 10.1 (scorer)**.** *Let $\alpha, \beta \in [0,1]$. We say that a randomized algorithm is an (α,β)-*scorer *if if for each input x. there is a scoring function $\phi_x : \{0,1\}^m \to [0,1]$, such that the "good" randomness strings $r \in \{0,1\}^m$ on input x are those with $\phi_x(r) \geq \tau$ for some $\tau \in (0,1)$. Moreover, ϕ_x satisfies the following properties:*

- Efficiency: *For every x, on input $r \in \{0,1\}^m$ the score $\phi_x(r)$ can be computed by an efficient* **BPP** *algorithm.*
- Approximate Monotonicity: *For every x, for every randomness r, $\Pr_{i\sim[m]}[\phi_x(r+e_i) < \phi_x(r)] \leq \beta$.*
- Improvement: *For every x, for every randomness r with $\phi_x(r) < \tau$, $\Pr_{i\sim[m]}[\phi_x(r+e_i) > \phi_x(r)] \geq \alpha$.*

The properties of approximate monotonicity and improvement highly resemble the properties guaranteed by (α,β)-insensitivity, In fact, since the improvement property holds for *every* bad randomness string, there is already an extremely simple predictor that compares the score of x and $x + e_i$ and predicts whichever has the higher score. Since the scoring function is efficient, this suggests that for many i, there is a discrepancy in score between r and $r' = r + e_i$ which allows for easy prediction.

Indeed, it follows easily that an $(\alpha-\beta)/3$-unpredictability generator is already a hitting set generator for such a distinguisher. To see this, suppose that such a generator is not a hitting set for the good randomness strings. Then every generator output has score less than τ. Consider the

[10] Notice the constraint on ζ required for the **BPP** result that requires it to be slightly bounded away from 1 is an artifact of our analysis. We generally think of ζ as very small, so that our result yields pseudoentropy $m^{0.99}$.

predictor P for coordinate i that given x_{-i}, computes the score on two possible randomness strings: one with 0 in the i-th position and one with 1 in the i-th position. The string with the lower score is likely x: there are at least αm choices of i for which this is true, and there are at most βm choices of i for which the opposite is true. Thus, if P outputs 0 or 1 accordingly, and outputs a random bit if the scores are equal, then it predicts correctly with probability $1/2 + (\alpha - \beta)/2$, a contradiction.

Extending this idea allows us to construct extremely efficient predictors from such distinguishers (linear in the runtime of the scoring algorithm!) and thus nearly optimal pseudorandomness.

Theorem 10.2. *There exists a universal constant $c > 0$ such that for every sufficiently small constant $\eta > 0$, the following holds. Suppose that for every $n \in \mathbb{N}$, there exists a function $f \in \{0,1\}^n$ that is printable in time $n^{1+\eta}$ and that is worst case hard for circuits of size $n^{1-\eta}$.*

Let $\theta = \eta/(2c)$. Suppose $A\colon \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$ is an (α,β)-scorer with scoring function $\phi_x\colon \{0,1\}^m \to [0,1]$ with threshold τ running in (randomized) time m, such that:

$$\Pr[\phi_x(r) \text{ outputs the correct score}] \geq 1 - m^{-3\theta},$$

where the probability is over the internal randomness of ϕ_x. Then, there exists a generator

$$G\colon \{0,1\}^{d=(1+O(\eta))\log m} \to \{0,1\}^m,$$

such that for any ε and for every x,

$$\Pr[\phi_x(G(y)) < \tau] - \Pr[\phi_x(U_m) < \tau] \leq \varepsilon,$$

as long as $\frac{\alpha}{2}\varepsilon - \frac{\beta}{2} > m^{-2\theta}$. In particular, when $\alpha \geq m^{-\theta}$ and $\beta \leq m^{-3\theta}$ we can take $\varepsilon = m^{-\theta/2}$. When α, β are constants, ε is a constant.

Proof: Let G be a $m^{-2\theta}/4$-symmetric unpredictability generator for circuits of size $m^{1+\eta}$ as in Theorem 8.9. Fix input x, and suppose that: $\Pr[\phi_x(G(y)) < \tau] - \Pr[\phi_x(U_m) < \tau] > \varepsilon$. In particular, this means that $\Pr[\phi_x(G(y)) < \tau] > \varepsilon$. Consider the following predictor P for G on input i, and $G(Y)_{-i}$.

1. Let $G(y)^0_{-i}$ be $G(y)_{-i}$ with 0 in the i-th coordinate. Let $G(y)^1_{-i}$ be $G(y)_{-i}$ with 1 in the i-th coordinate.
2. Compute $\phi_x(G(y)^1_{-i})$ and $\phi_x(G(y)^1_{-i})$. Output 1 if $\phi_x(G(y)^0_{-i}) > \phi_x(G(y)^1_{-i})$. Output 0 if $\phi_x(G(y)^0_{-i}) < \phi_x(G(y)^1_{-i})$. Output a random bit if the values are equal.

Let $D \subset \{0,1\}^m$ be the set of randomness strings with score less than τ. For any $G(y) \in D$, we know there are at least αm choices of i for which the predictor is correct by the improvement property. We also know that there are at most βm choices of i for which the predictor is wrong by the approximate monotonicity property. Finally, for any $G(y) \notin D$, there are at most βm choices of i for which the predictor is wrong, once again by approximate monotonicity.

Thus P is correct (over randomness i, y and internal randomness of ϕ) whenever:

- Both calls to ϕ are correct over their own internal randomness (this happens with probability at least $1 - 2m^{-3\theta}$), and one of the following holds.
- $G(y) \in D$ (which happens with probability at least ε), and i is "good," meaning a correct prediction is definitely made or "neutral," meaning a random guess is made (overall the probability of correctness over i here is at least $\alpha + \frac{1}{2}(1 - \alpha - \beta)$);

- $G(y) \notin D$ which happens with the remaining probability (at most $1-\varepsilon$), in which case the probability of a correct answer is $\frac{1}{2}(1-\beta)$.

Overall the probability of a correct prediction is at least:

$$\begin{aligned}
&(1-2m^{-3\theta})\left(\varepsilon\left(\alpha+\frac{1}{2}(1-\alpha-\beta)\right)+(1-\varepsilon)\,\frac{1}{2}(1-\beta)\right)\\
&=(1-2m^{-3\theta})\left(\frac{1}{2}+\frac{\alpha}{2}\cdot\varepsilon-\frac{\beta}{2}\right)\\
&=\frac{1}{2}+\frac{\alpha}{2}\cdot\varepsilon-\frac{\beta}{2}-m^{-3\theta}-\alpha m^{-3\theta}\varepsilon+\beta m^{-3\theta}\\
&>\frac{1}{2}+\frac{m^{-2\theta}}{4}.
\end{aligned}$$

Thus, we can fix i, and the internal randomness of ϕ_x to get a predictor of size $O(m)$ with the same advantage that contradicts our generator. ■

10.1 A Motivating Example for Scorers: Minimum Spanning Tree

We present motivation for our notion of a scorer by considering the expected linear time algorithm for minimum spanning tree due to Karger, Klein, and Tarjan [KKT95]. We clarify that there already exist deterministic minimum spanning tree algorithms with nearly linear runtime [Cha00, PR02] that are thus more efficient than any derandomization we expect from our results. Thus, we discuss the algorithm at a high level mainly to motivate our notion of scorers.

On input a weighted undirected graph $G=(V,E)$, $w\colon E\to\mathbb{R}$, the algorithm's goal is to find a subtree $T=(V,E_T)$ of G with minimum overall weight $\sum_{e\in E_T} w(e)$. Without loss of generality (thanks to a transformation by Borůvka [Bor26]; see also [KKT95]), the number of edges can be made arbitrarily larger than the number of vertices.

The algorithm picks a random subgraph $G'=(V,E')$ of G, recursively finds a minimum spanning tree T' in G', and uses that tree to find edges in G that it can rule out since they are not part of a minimum spanning tree of G. The subgraph is chosen by including each edge in E independently with probability p. A common choice of p is $1/2$, and so the randomness uses $|E|$ uniform bits. The edges it keeps are those in T' and edges in E'', defined as edges $e\in E$ that are better than T' in the sense that T' is not a minimum spanning tree of $T'\cup\{e\}$. Thanks to an existing algorithm [Kin95, DRT92] one can find E'' in deterministic linear time. The algorithm rules out about half of the edges each time, and recursively finds a minimum spanning tree. Overall the runtime is linear with high probability.

We'll show that a single call to this algorithm is a scorer, with the caveat that the improvement property only holds for "nearly balanced" strings, ones for which the fraction of 1-s is in, say $\left[\frac{1}{2}-0.2,\frac{1}{2}+0.2\right]$. We note that the analysis of pseudorandomness for scorers above still holds (with a little bit extra error probability) even in such a case, because, according to Theorem 10.3 below, an unpredictability generator outputs nearly balanced strings with high probability.

Consider a single call of the algorithm (without recursion), so the randomness comes from choosing G'. Consider the score $\phi_{G,w}(G')=|E-E''|/|E|$, the fraction of edges that are not better than T'. For some constant $\tau\in[0,1]$, the goal of the call is to rule out at least τ fraction of the edges in G. The score can be computed in randomized linear time, by invoking the randomized linear

time minimum spanning tree algorithm on G'. We can consider this either a ZPP or BPP algorithm, depending on whether we allow it to run until completion or impose a fixed time cutoff.

To see that the score satisfies approximate monotonicity, observe that a random flip can only decrease the score if it takes out one of the tree edges of T'' from G'. This happens with probability at most $(|V|-1)/|E|$.

For the improvement property, consider when $\phi_{G,w}(G') < \tau$, and the randomness string is nearly balanced. Then with probability close to $1/2$ a random flip causes a random edge e to be *added* to G'. In this event, as long as e was not already ruled out by G' (there are at most τ fraction ruled out), at least one more edge is ruled out. In particular, e itself was an edge that was not ruled out by G', but is now included in the MST of the new G'. This happens with probability at least $\frac{1}{2} - 0.2 - \tau$.

We finally present a proof that unpredictability generators output nearly balanced strings.

Theorem 10.3. *Let $X = X_1, \ldots, X_m$ be $(\varepsilon = m^{-3\theta})$-symmetrically unpredictable for circuits of size $m \log m$ and for small constant θ. Let $|x|$ be the number of 1-s in x. Then $\Pr[|x| \notin [(\frac{1}{2}-\delta)m, (\frac{1}{2}+\delta)m] \leq \alpha$, for $\delta = \alpha = m^{-\theta}$.*

Proof: Suppose not. Then either $\Pr[|x| < (\frac{1}{2}-\delta)m] > \alpha/2$, or $\Pr[|x| > (\frac{1}{2}+\delta)m] > \alpha/2$. Without loss of generality, assume the latter. For any $m/2 \leq t \leq (\frac{1}{2}+\delta)m$, let the predictor $P_t(i, x_{-i})$ output 1 if $|x_{-i}| \geq t$ and output a random bit otherwise. There are three cases for such a P_t:

- If $|x| > t$, the predictor outputs 1 regardless of i. Thus whenever i is such that $x_i = 1$, the predictor is correct, and whenever $x_i = 0$, the predictor is incorrect. Overall, the probability of being correct is $|x|/m$.
- If $|x| = t$, then if $x_i = 1$, then $|x_{-i}| = t - 1 < t$, so the predictor guesses randomly. If $x_i = 0$, $|x_{-i}| = t$ so the predictor outputs 1 and is incorrect. Overall, the probability of being correct is $\frac{|x|}{2m}$.
- If $x < t$, the predictor always guesses randomly. So the probability of being correct is $1/2$.

Now consider the predictor P that on input i and X_{-i}, picks a random $m/2 \leq t \leq (\frac{1}{2}+\delta)m$ and outputs $P_t(i, x_{-i})$. Notice this predictor runs in time $O(m)$.

- Say X is *good* whenever $|X| > (\frac{1}{2}+\delta)m$. In this case, regardless of the choice of t, we have $|X| > t$. By assumption, X is good with probability $> \alpha/2$ over X, and given such an X, the probability of being correct is $> \frac{1}{2} + \delta$.
- Say X is *tricky* when $m/2 \leq |X| \leq (\frac{1}{2}+\delta)m$. Fix any such x. The probability $|x| = t$ is $\frac{1}{\delta m+1} \leq \frac{1}{\delta m}$. For any other choice of t, the probability of being correct is $\geq \frac{1}{2}$.
- Say X is *a wash* if $|X| < m/2$. In this case $|X| < t$ for any choice of t, and so the predictor always outputs a random bit.

Thus, the probability this predictor is correct (over i, X, t and internal randomness) is:

$$\begin{aligned}
&\Pr[\text{correct} \mid X \text{ is good}] \Pr[X \text{ is good}] \\
&+ \Pr[\text{correct} \mid X \text{ is tricky}] \Pr[X \text{ is tricky}] \\
&+ \Pr[\text{correct} \mid X \text{ is a wash}] \Pr[X \text{ is a wash}] \\
&> \left(\frac{1}{2} + \delta\right) \Pr[X \text{ is good}] + \frac{1}{2}\left(1 - \frac{1}{\delta m}\right) \Pr[X \text{ is tricky}] + \frac{1}{2} \Pr[X \text{ is a wash}] \\
&\geq \frac{1}{2} + \frac{\delta\alpha}{2} - \frac{1}{\delta m} \\
&\geq \frac{1}{2} + \frac{m^{-2\theta}}{2} - m^{-(1-\theta)} \\
&\geq \frac{1}{2} + m^{-3\theta}.
\end{aligned}$$

We can fix i and t to preserve this advantage and get a predictor for a specific i, contradicting the symmetric unpredictability of X. ∎

References

[ACR98] A. E. Andreev, A. E.F. Clementi, and J. D. P. Rolim. A new general derandomization method. *J. ACM*, 45(1):179–213, January 1998.

[AYZ95] N. Alon, R. Yuster, and U. Zwick. Color-coding. *Journal of the ACM*, 42(4):844–856, July 1995.

[BCT25] Marshall Ball, Lijie Chen, and Roei Tell. Towards free lunch derandomization from necessary assumptions (and OWFs). In *Computational Complexity Conference (CCC)*, pages 31–1. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2025.

[BGG+24] Vishwas Bhargava, Sumanta Ghosh, Zeyu Guo, Mrinal Kumar, and Chris Umans. Fast multivariate multipoint evaluation over all finite fields. *Journal of the ACM*, 71(3):1–32, 2024.

[BHKT24] Guy Blanc, Alexandre Hayderi, Caleb Koch, and Li-Yang Tan. The sample complexity of smooth boosting and the tightness of the hardcore theorem. In *Annual Symposium on Foundations of Computer Science FOCS*, pages 1431–1450. IEEE, 2024.

[BM84] M. Blum and S. Micali. How to generate cryptographically strong sequences of pseudorandom bits. *SIAM Journal on Computing*, 13:850–864, 1984.

[Bor26] O. Borůvka. O jistém problému minimálním. *Práce Moravské přírodovědecké společnosti*, 3:37–58, 1926.

[BSW03] Boaz Barak, Ronen Shaltiel, and Avi Wigderson. Computational analogues of entropy. In *Approximation, Randomization, and Combinatorial Optimization – Algorithms and Techniques (RANDOM)*, pages 200–215. Springer, 2003.

[CH20] Kuan Cheng and William M. Hoza. Hitting sets give two-sided derandomization of small space. In *Computational Complexity Conference (CCC)*, pages 10–1. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020.

[Cha00] Bernard Chazelle. A minimum spanning tree algorithm with inverse-ackermann type complexity. *J. ACM*, 47(6):1028–1047, November 2000.

[CHR24] Lijie Chen, Shuichi Hirahara, and Hanlin Ren. Symmetric exponential time requires near-maximum circuit size. In *Annual Symposium on Theory of Computing (STOC)*, pages 1990–1999. ACM, 2024.

[CLO+23] Lijie Chen, Zhenjian Lu, Igor C. Oliveira, Hanlin Ren, and Rahul Santhanam. Polynomial-time pseudodeterministic construction of primes. In *Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1261–1270. IEEE, 2023.

[CT21] Lijie Chen and Roei Tell. Simple and fast derandomization from very hard functions: eliminating randomness at almost no cost. In *Annual Symposium on Theory of Computing (STOC)*, pages 283–291. ACM, 2021.

[CT24] Lijie Chen and Roei Tell. Hardness vs. randomness, revised: uniform, non-black-box, and instance-wise. *SIAM Journal on Computing*, pages FOCS21–323, 2024.

[DHIP26] Yotam Dikstein, Max Hopkins, Russell Impagliazzo, and Toniann Pitassi. High rate efficient local list decoding from HDX. *arXiv preprint arXiv:2601.22535*, 2026.

[DMOZ22] Dean Doron, Dana Moshkovitz, Justin Oh, and David Zuckerman. Nearly optimal pseudorandomness from hardness. *Journal of the ACM*, 69(6):1–55, 2022.

[DPTW25] Dean Doron, Edward Pyne, Roei Tell, and R. Ryan Williams. When connectivity is hard, random walks are easy with non-determinism. In *Annual Symposium on Theory of Computing (STOC)*, pages 1108–1117. ACM, 2025.

[DRT92] D. Dixon, M. Rauch, and R. E. Tarjan. Verification and sensitivity analysis of minimum spanning trees in linear time. *SIAM Journal on Computing*, 21(6):1184–1192, 1992.

[FSUV13] Bill Fefferman, Ronen Shaltiel, Christopher Umans, and Emanuele Viola. On beating the hybrid argument. *Theory of Computing*, 9(26):809–843, 2013.

[Gol22] O. Goldreich. Oded's choice number 324. https://www.wisdom.weizmann.ac.il/~oded/MC/324.pdf, 2022.

[GSV18] Aryeh Grinberg, Ronen Shaltiel, and Emanuele Viola. Indistinguishability by adaptive procedures with advice, and lower bounds on hardness amplification proofs. In *Annual Symposium on Foundations of Computer Science (FOCS)*, pages 956–966. IEEE, 2018.

[Har66] L.H. Harper. Optimal numberings and isoperimetric problems on graphs. *Journal of Combinatorial Theory*, 1(3):385–393, 1966.

[HILL99] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM Journal on Computing*, 28(4):1364–1396, 1999.

[Hir22] Shuichi Hirahara. NP-hardness of learning programs and partial MCSP. In *Annual Symposium on Foundations of Computer Science (FOCS)*, pages 968–979. IEEE, 2022.

[HLR07] Chun-Yuan Hsiao, Chi-Jen Lu, and Leonid Reyzin. Conditional computational entropy, or toward separating pseudoentropy from compressibility. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT)*, pages 169–186. Springer, 2007.

[HMS23] Iftach Haitner, Noam Mazor, and Jad Silbak. Incompressiblity and next-block pseudoentropy. In *Innovations in Theoretical Computer Science Conference (ITCS)*, pages 66–1. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2023.

[IW97] Russell Impagliazzo and Avi Wigderson. P = BPP if E requires exponential circuits: Derandomizing the XOR lemma. In *Annual Symposium on Theory of Computing (STOC)*, pages 220–229. ACM, 1997.

[Kin95] V. King. A simpler minimum spanning tree verification algorithm. In *International Workshop on Algorithms and Data Structures (WADS)*, page 440–448. Springer, 1995.

[KKT95] D. R. Karger, P. N. Klein, and R. E. Tarjan. A randomized linear-time algorithm to find minimum spanning trees. *Journal of the ACM*, 42(2):321–328, March 1995.

[KPWW16] Stephan Krenn, Krzysztof Pietrzak, Akshay Wadia, and Daniel Wichs. A counterexample to the chain rule for conditional HILL entropy. *computational complexity*, 25(3):567–605, 2016.

[LPT24] Jiatu Li, Edward Pyne, and Roei Tell. Distinguishing, predicting, and certifying: On the long reach of partial notions of pseudorandomness. In *Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1–13. IEEE, 2024.

[NW94] Noam Nisan and Avi Wigderson. Hardness vs randomness. *Journal of Computer and System Sciences*, 49(2):149–167, 1994.

[OS25] Justin Oh and Ronen Shaltiel. Extractors for samplable distributions from the two-source extractor recipe. *ECCC*, TR25-107, 2025.

[PF79] Nicholas Pippenger and Michael J. Fischer. Relations among complexity measures. *Journal of the ACM*, 26(2):361–381, 1979.

[PR02] Seth Pettie and Vijaya Ramachandran. An optimal minimum spanning tree algorithm. *J. ACM*, 49(1):16–34, January 2002.

[PRZ23] Edward Pyne, Ran Raz, and Wei Zhan. Certified hardness vs. randomness for log-space. In *Annual Symposium on Foundations of Computer Science (FOCS)*, pages 989–1007. IEEE, 2023.

[Rei03] Omer Reingold. Personal Communication, 2003.

[SGP15] Maciej Skorski, Alexander Golovnev, and Krzysztof Pietrzak. Condensed unpredictability. In *International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 1046–1057. Springer, 2015.

[Spi25] Daniel A. Spielman. Spectral and algebraic graph theory. http://cs-www.cs.yale.edu/homes/spielman/sagt/sagt.pdf, 2025. Lecture notes.

[STV01] Madhu Sudan, Luca Trevisan, and Salil Vadhan. Pseudorandom generators without the XOR lemma. *Journal of Computer and System Sciences*, 62(2):236–266, 2001.

[SU05] Ronen Shaltiel and Christopher Umans. Simple extractors for all min-entropies and a new pseudorandom generator. *Journal of the ACM*, 52(2):172–216, 2005.

[SV22] Ronen Shaltiel and Emanuele Viola. On hardness assumptions needed for "extreme high-end" PRGs and fast derandomization. In *Innovations in Theoretical Computer Science Conference (ITCS)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2022.

[Tre01] Luca Trevisan. Extractors and pseudorandom generators. *Journal of the ACM*, 48(4):860–879, 2001.

[Tre13] Luca Trevisan. Expansion, sparsest cut, and spectral graph theory. https://lucatrevisan.github.io/books/expanders.pdf, 2013. Lecture notes.

[Uma03] Christopher Umans. Pseudo-random generators for all hardnesses. *Journal of Computer and System Sciences*, 67(2):419–440, 2003.

[Vad12] Salil Vadhan. Pseudorandomness. *Foundations and Trends® in Theoretical Computer Science*, 7(1–3):1–336, 2012.

[Wil16] Ryan Williams. Strong ETH breaks with Merlin and Arthur: Short non-interactive proofs of batch evaluation. In *Conference on Computational Complexity (CCC)*, pages 2:1–2:17. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2016.

[Yao82] A. C. Yao. Theory of cryptographic protocols and pseudo-random generators. *Annual Symposium on Foundations of Computer Science (FOCS)*, pages 100–109, 1982.

ECCC ISSN 1433-8092

https://eccc.weizmann.ac.il