The Benefits of Temporal Correlations: SGD Efficiently Learns k-Juntas from Random Walks

Elisabetta Cornacchia*
Department of Computing Sciences
Bocconi University
elisabetta.cornacchia@unibocconi.it

Dan Mikulincer
Department of Mathematics
University of Washington
danmiku@uw.edu

Elchanan Mossel
Department of Mathematics
MIT
elmos@mit.edu

Abstract

We study how temporal correlations in the data can make certain sparse learning problems efficiently learnable by gradient-based methods. Our focus is on Boolean k-juntas, a canonical sparse learning problem known to pose barriers for gradient-based methods under independent uniform samples. We show that this picture changes when the samples are generated by a lazy random walk on the hypercube. In this setting, the temporal dependencies can be exploited by a two-layer ReLU network trained using stylized-SGD with a temporal-difference loss, which compares target and predicted increments across consecutive samples. For every fixed k, the resulting sample complexity is essentially linear in the ambient dimension d. By contrast, we show that for large-batch gradient methods using standard convex pointwise losses, temporal correlations do not provide the same advantage.

1 Introduction

The empirical success of neural networks continues to outpace our theoretical understanding. One way to narrow this gap is to study simplified settings in which gradient-based methods can be shown to learn efficiently [DLL+19, DLS22, ABAM22, AGJ21]. Much of the existing work in this direction falls into one of two extremes: worst-case models, such as distribution-free PAC learning, where no statistical structure is assumed [Val84, SSBD14], and idealized stochastic models, where examples are drawn independently from a fixed distribution [AGJ21, BB20, AAM23, AZLS19, DTA+24, BBSS22, Gla23]. While both viewpoints have yielded important insights, they leave open an intermediate regime that is arguably closer to many real data sources: samples are neither adversarially chosen nor independent, but exhibit structured correlations.

Temporal correlations are a particularly common form of such structure. They arise in domains such as computer vision [RLRL24], healthcare [YCC+22], and environmental modeling [AGRK20], where data are generated sequentially and often display substantial temporal dependence or distribution shift. In this work, we study an idealized and tractable model of temporal dependence, focusing on the random walk on the hypercube. Our goal is to understand whether such temporal structure can itself influence the performance of gradient-based learning, complementing prior work that has focused on the role of spatial or geometric structure [GMKZ20, GLK+20].

*Authors are listed in alphabetical order.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

To study the effect of temporal correlations, we focus on learning Boolean k-juntas. A junta is a function $f : \{\pm 1\}^d \to \mathbb{R}$ that depends only on a small subset of the coordinates: that is, there exists a set $S \subseteq [d]$ with $|S| = k$, and $k = O(1)$, such that $f(x) = \tilde{f}(x_S)$, where x_S denotes the restriction of x to the coordinates in S. This problem has attracted significant recent attention as a canonical model of sparse feature learning, and has served as a useful testbed to rigorously study phenomena such as incremental feature learning [ABAM22, ABLR23], separations between loss functions [JMS24], curriculum learning [CM23, ACL23], and knowledge distillation [PLM+24]. In the *i.i.d.* uniform setting, previous works have established complexity *lower bounds* that scale unfavorably with k, and showed that several natural algorithms, including gradient-based training of standard neural networks, require $d^{\Omega(k)}$ time [DM20, AS20a, AAM23, SSSS17].

In contrast, we study learning f from samples generated by a random walk on the hypercube. The resulting temporal correlations allow gradient-based methods to bypass barriers that are specific to independent uniform samples. Our main result shows that, with an appropriate temporal-difference loss, a two-layer ReLU network trained by a stochastic gradient-based method can learn any k-junta using a number of samples essentially linear in the ambient dimension d, for every fixed k. Interestingly, our theory and experiments suggest that whether this advantage is realized depends crucially on both the choice of loss function and the batch size.

2 Summary of Contributions

To explain our results, it is helpful to begin with the special case of a k-parity, $f(x) = \prod_{i \in S} x_i$, where $S \subseteq [d]$, $|S| = k$, is the unknown support. Although parities are efficiently learnable by specialized algorithms, such as Gaussian elimination over $\mathbb{F}_2$, they pose a well-known barrier for gradient-based methods under *i.i.d.* uniform inputs. In particular, for gradient descent with limited gradient precision, $d^{\Omega(k)}$ steps are needed in this setting [DM20, AS20a, BEG+22, KCGK24]. The intuition is that, under the uniform distribution, $\mathbb{E}[f(x)x_j] = 0$ for every $j \in [d]$, so first-order pointwise correlations do not reveal the support of the target parity.

The situation changes when the samples are generated by a random walk on the hypercube. Since each step changes at most one coordinate, the finite difference

$$\Delta f^{(t)} := f(x_{t+1}) - f(x_t)$$

is nonzero only when the updated coordinate belongs to S. Thus, the support can be recovered by a simple procedure that records which coordinate flips produce nonzero label increments and a coupon-collector argument shows that this takes $\Theta(d \log k)$ steps. Our goal, however, is not to rely on this explicit problem-specific procedure. Rather, we ask whether gradient-based methods can extract the same information automatically.

Our main result shows that when a neural network is trained by a stochastic gradient-based algorithm with a suitable loss, the updates themselves pick up the information carried by the temporal differences, thus one does not need to recover the support by hand. To go beyond parities, we must consider the influences $\mathrm{Inf}_i(f)$ within the support when $i \in S$, to ensure the finite differences carry enough information, see Section 4 for the exact definition. We state the following informal result for k-juntas.

Theorem 1 (Informal)**.** *Let $f : \{\pm 1\}^d \to \mathbb{R}$ be a k-junta, supported on $S \subset [d]$, let $(x_t)_{t=1}^T$ be a lazy random walk on $\{\pm 1\}^d$, and let $\varepsilon > 0$. Suppose that $\min_{i \in S} \mathrm{Inf}_i(f) \geq \tau$, for some $\tau > 0$, and that $T = \tilde{\Omega}_{k,\varepsilon,\tau}(d)$. Then, a two-layer neural network, trained using layerwise SGD with suitable loss (see Algorithm 1), on the samples $(x_t, f(x_t))_{t=1}^T$, achieves*

$$\mathbb{E}_{x \sim \mathrm{Unif}(\{\pm 1\}^d)} \left[(\mathrm{NN}(x) - f(x))^2 \right] < \varepsilon.$$

The mechanism behind the result is a second-moment effect created by the random-walk dynamics. For a parity of degree at least two, the signed quantity

$$\Delta f^{(t)} \Delta x_j^{(t)},$$

where $\Delta x_j^{(t)} := x_j^{(t)} - x_j^{(t-1)}$, still has mean zero, so there is no informative first-order signal. However, its second moment distinguishes relevant and irrelevant coordinates: it is positive for $j \in S$ and zero for $j \notin S$. Thus, temporal differences induce a support-dependent anisotropy that is absent

under independent uniform sampling. The temporal-difference loss (see Def. 2) is designed to expose this anisotropy to the gradient updates.

In Section 4, we make Theorem 1 precise and give the full layerwise SGD procedure. Up to problem-dependent constants and lgoarithmic terms, the sample complexity of the resulting gradient-based algorithm matches the $\Theta(d \log k)$ scale of the explicit support-recovery procedure. Compared with the $d^{\Omega(k)}$ barriers known for independent uniform samples, this shows that temporal correlations can dramatically change the tractability of sparse learning problems.

Our second contribution is a complementary lower bound showing that temporal correlations are not automatically useful for all gradient-based methods. In Section 5, we prove that noisy gradient descent with large batch size and standard convex pointwise losses, such as the square or hinge loss, remains controlled by a cross-predictability barrier analogous to the *i.i.d.* setting. Thus, in the large-batch regime, pointwise losses do not substantially exploit the temporal structure. By contrast, this lower bound does not apply to small-batch SGD, and our experiments suggest that small-batch square-loss SGD may sometimes exploit temporal dependencies as well. We discuss this phenomenon in Section 6.

3 Related Works

Learning juntas from i.i.d. data. Several works considered the problem of learning juntas, and in particular sparse parities, under *i.i.d. uniform* inputs, as a testbed for understanding the computational limits of gradient descent methods. For parity functions, stochastic gradient descent (SGD) with small batch size can learn the target efficiently on an emulation network [AS20b]. By contrast, in the Statistical Query (SQ) model [Kea98], and similarly for gradient descent with limited gradient precision, any such learning procedure requires at least $\Omega\big(d^k\big)$ time [AS20b]. For standard shallow networks, positive results have been established for sparse parities, matching the SQ lower bound [BEG+22, Gla23, KCGK24]. For juntas, the complexity depends on the Fourier-Walsh structure of the target function, as captured by the *leap* [ABAM22, AAM23], and the *SQ leap* [JMS24] complexities. Other works have shown that under i.i.d. inputs from shifted product distributions, sparse parities [MKAS21, DM20] and juntas [CMM25] become significantly easier than under the uniform distribution.

Beyond i.i.d. data. Recent works show that gradient-based learnability of sparse Boolean functions can change beyond the static *i.i.d.* setting. Positive distribution shift (PDS), for example, shows that suitably shifted training distributions can help SGD perform better on the test distribution [MAC+26], while curriculum-learning results vary the training distribution over time to expose easier or more informative examples first [CM23, ACL23]. Our results fit into this broader line, but the mechanism is different: the learning advantage does not come from a hand-designed distribution shift, but from the temporal correlations generated by the sampling process itself. Gradient-based optimization with Markovian sampling scheme has also been studied from an optimization perspective [SSY18, Eve23]. These results primarily quantify how temporal dependence affects optimization rates for a fixed objective, whereas our focus is on showing that the temporal correlations can change the information exposed to the gradients, making certain sparse learning problems easier.

Random walk data as an extension of PAC learning. The problem of learning from correlated examples, particularly through the observation of a Markovian random walk on the Boolean hypercube, has emerged as a rich intermediate model between passive uniform PAC learning and active learning with membership queries. The idea of incorporating temporal differences goes back to Sutton [Sut88], while the present setting was initially proposed by Aldous and Vazirani [AV90], and later formalized computationally by Bartlett, Fischer, and Höffgen [BFH02]. A notable result in this setting is the result by Bshouty, Mossel, O'Donnell, and Servedio [BMOS05], who demonstrated that polynomial-size Disjunctive Normal Form (DNF) formulas can be efficiently learned from a uniform random walk. By bridging the random walk model with statistical query and noise sensitivity frameworks, their work established the viability of random walks to simulate models that rely on uniformly distributed but correlated data. A subsequent work [ELSW05] provided a rigorous separation, proving that under cryptographic assumptions, random walk learning is strictly more powerful than standard uniform PAC learning. For the specific problem of learning parities and juntas it was well known that these can be learned using random walks. Arpe and Mossel [AM08] proved that the class

of k-juntas can be *agnostically* learned from a uniform random walk on $\{\pm 1\}^d$. Their algorithm, which is combinatorial in nature, achieves this in time polynomial in d, 2^{k^2}, ε^{-k}, and $\log(1/\delta)$. This framework was later expanded by Jackson and Wimmer [JW14], who generalized the random walk model and corroborated the efficient agnostic learnability of juntas within these broader correlated spaces. More recent work on graphical models also showed the advantage of using dynamics such as Glauber dynamics to learn Markov Random Fields [BGS18, GMM24].

4 Efficient Learning with the Temporal Difference Loss

In this section, we formalize Theorem 1 by introducing a stochastic gradient-based algorithm to train a neural network, while leveraging temporal correlations in the data. As we will demonstrate, this algorithm can learn general k-juntas which satisfy a natural non-degeneracy condition. We begin by explaining the exact setting and the input to the algorithm, then we present Algorithm 1, and state its convergence and statistical guarantees.

Data. We consider learning a Boolean k-junta, i.e. $f : \{\pm 1\}^d \to \mathbb{R}$ such that $f(x) = \tilde{f}(x_S)$, where $S \subseteq [d]$ and with $|S| = k = O_d(1)$. We consider inputs generated from a *lazy* random walk on the hypercube.

Definition 1 (Lazy random walk on the hypercube with flip probability p)**.** *Let*

$$x^{(0)} \sim \mathrm{Unif}\{\pm 1\}^d,$$

and for each $t \geq 0$, define

$$x^{(t+1)} = x^{(t)} - 2Z_t x^{(t)}_{j_t} e_{j_t},$$

where $j_t \overset{iid}{\sim} \mathrm{Unif}\{1, ..., d\}$, and $Z_t \overset{iid}{\sim} \mathrm{Ber}(p)$, with some fixed $p \in (0, 1)$, and j_t, Z_t are independent of each other, and of $x^{(0)}$. $e_{j_t} \in \{0, 1\}^d$ here denotes the j_t-th standard basis vector (i.e., $(e_{j_t})_i = \mathbb{1}\{i = j_t\}$)

In other words, at each step with probability $1 - p$ the chain stays put, and with probability p it flips a uniformly random coordinate.

Architecture and algorithm. We consider a two-layered ReLU network:

$$\mathrm{NN}(x; \theta) = \sum_{i=1}^{N} a_i \,\mathrm{ReLU}(w_i^T x + b_i), \tag{1}$$

where for every $i \in [N]$, $w_i \in \mathbb{R}^d$, $a_i, b_i \in \mathbb{R}$, and $\theta = (w_i, a_i, b_i)_{i=1}^N$ denotes the set of all trainable parameters.

To train the network, we will take gradient steps to minimize the *temporal difference (TD) loss*, which is particularly suited to handle temporal correlations, and which we now present.

Definition 2 (Temporal Difference (TD) loss)**.** *Define the* temporal difference (TD) *loss with parameter $\alpha \in [0, 1]$ as follows. For an estimator $\hat{f} : \mathbb{R}^d \to \mathbb{R}$ relative to a target function $f : \mathbb{R}^d \to \mathbb{R}$ computed at samples $x^{(t-1)}$ and $x^{(t)}$ as:*

$$\mathcal{L}_\alpha^{\mathrm{TD}}(f, \hat{f}; x^{(t)}, x^{(t-1)}) := \frac{\alpha}{2}\left(\Delta f^{(t)} - \Delta \hat{f}^{(t)}\right)^2 + \frac{1-\alpha}{2}\left(f(x^{(t)}) - \hat{f}(x^{(t)})\right)^2, \tag{2}$$

where $\Delta f^{(t)} = f(x^{(t)}) - f(x^{(t-1)})$ and similarly $\Delta \hat{f}^{(t)} = \hat{f}(x^{(t)}) - \hat{f}(x^{(t-1)})$.

If $\alpha = 0$, this reduces to the standard square loss. If $\alpha = 1$, the loss depends only on finite differences along the trajectory and is invariant to adding a constant to $\hat{f}$ (i.e., any $\hat{f} = f + c$ yields zero loss), so one cannot use $\alpha = 1$ if the goal is to learn the function itself. Nevertheless, in the algorithm below, we will start with $\alpha = 1$, since this is useful in order to extract information about the support of f. We will then switch to $\alpha < 1$, in order to remove the degeneracy of the loss, and learn the function while knowing its support. This choice is made mainly for convenience in the analysis and should not

Algorithm 1 *Layerwise SGD with temporal difference loss*
Init. scale $\kappa > 0$, bias range A, learning rates γ_1, γ_2, step counts T_1, T_2, batch size B, and regularization $\lambda > 0$.

Require: Training data $\{(x_t, y_t)\}_{t\geq 1}$, model: $\mathrm{NN}(x; (w, a, b)) = \sum_{i=1}^{N} a_i \sigma(w_i^\top x + b_i)$.

Initialize: weights $w^{(0)} = 0$, $a^{(0)} = \kappa$, and biases $b^{(0)} = 1/N$.

Phase 1: Train the first layer (second layer frozen)

for $t = 0$ **to** $T_1 - 1$ **: do**
  Take mini-batch $S^{(t)} = \{(x_s, y_s)\}_{s=1}^{B}$
  Update $w^{(t+1)} \leftarrow w^{(t)} - \gamma_1 \nabla_{w^{(t)}} \mathcal{L}_1^{\mathrm{TD}}(S^{(t)}; w^{(t)}, a^{(0)}, b^{(0)})$
end for

Phase 2: Train the second layer (first layer frozen)

Redraw random biases: $\hat{b}_1, \ldots, \hat{b}_N \sim \mathrm{Unif}[-A, A]^{\otimes N}$
for $t = T_1$ **to** $T_2 - 1$ **: do**
  Set $S^{(t)} = (x_t, y_t)$
  Update $a^{(t+1)} \leftarrow a^{(t)} - \gamma_2 \nabla_a \left(\mathcal{L}_0^{\mathrm{TD}}(S^{(t)}; w^{(T_1)}, a^{(t)}, \hat{b}) + \frac{\lambda}{2} \|a^{(t)}\|_2^2 \right)$
end for

Output: Trained model $\mathrm{NN}(x; (w^{(T_1)}, a^{(T_2)}, \hat{b}))$

be viewed as essential; In our experiments in Section 6, we choose α sufficiently close to 1, and we keep it fixed throughout training.

When using the TD loss, we always take mini-batches to consist of consecutive pairs $(x^{(t-1)}, x^{(t)})$ taken from a single random-walk trajectory. With these details in place, we implement SGD with the TD loss in Algorithm 1, which is carried out in two layer-wise phases.

- In the first phase, the first layer is trained for T_1 steps using a large mini-batch of size B, while the second layer is kept fixed. In practice, we show that as long as B is large enough, we can take $T_1 = 1$. The main point is that the TD loss picks up information concerning the support of f. As a result, the weights at the end of this phase carry a nontrivial signal on the support and little signal outside it. As explained, for convenience, we set $\alpha = 1$ in the TD loss for this step and perform SGD with respect to $\mathcal{L}_1^{\mathrm{TD}}$.
- In the second phase, the first layer is kept fixed and only a, the weights of the second layer, are trained using T_2 steps of SGD. At this point, once the relevant support information has been extracted, the remaining problem is essentially a low-dimensional regression problem. In this phase, it is important to take $\alpha < 1$ to ensure proper learning of the function f, and so we set $\alpha = 0$ and perform standard SGD steps for L_0^{TD}, which is the L_2 loss. To make sure the resulting regression problem is well-posed, we add the regularization term $\frac{\lambda}{2}\|a\|^2$. Remark that, as we explain below, we could remove the regularization if, instead of taking the terminal weight $a^{(T_2)}$, we would instead average the weights along the process.

Before the first phase, we initialize for all $i \in [N], j \in [d]$: $w_{ij}^{(0)} = 0,\ a_i^{(0)} = \kappa,\ b_i^{(0)} = 1/N$, for some $\kappa > 0$. After the first phase of training, we redraw the bias neurons from $b_i \overset{iid}{\sim} \mathrm{Unif}[-A, A]$, for an appropriate $A > 0$.

Main Result To state our main result for Algorithm 1 we need to introduce one final concept, the influences of f. If $f : \{\pm 1\}^d \to \{\pm 1\}$ the influence of f at direction i is $\mathrm{Inf}_i(f) := \mathbb{P}\left(f(x) \neq f(x^{(i)})\right)$, where the probability is over the uniform measure on $\{\pm 1\}^d$ and $x^{(i)}$ is obtained from x by flipping the ith bit. More generally, write any $f : \{\pm 1\}^d \to \mathbb{R}$ in the Fourier-Walsh monomial basis $f(x) = \sum_{A \subset [d]} \hat{f}_A x^A$. Then, $\mathrm{Inf}_i(f) = \sum_{i \in A} \hat{f}_A^2$ generalizes the previous definition from Boolean-valued to real-valued functions. In that sense, the influence of f in direction i measures the sensitivity of f to flipping the ith bit. In order to learn f from the random-walk trajectory, we require that the influences of f are not too small for directions in its support. Thus, if S is the support

of f, we will henceforth denote $\tau = \min_{i \in S} \mathrm{Inf}_i(f)$. All constants below will implicitly depend on τ as well as on $\|f\|_\infty$.

Theorem 2. *Let $\{(x^{(t)}, y^{(t)})\}_{t \in [T]}$, for $T \in \mathbb{N}$, be such that $\{x^{(t)}\}_{t \in [T]}$ is generated by the $\frac{1}{2}$-lazy random walk on the d-dimensional hypercube, as in Def. 1, and $y^{(t)} = f(x^{(t)})$, where f is a k-junta, with $k = O_d(1)$. Consider training a 2-layer* ReLU *network with $N = \Theta\left(\frac{\log(d) 2^k A}{\varepsilon}\right)$ hidden neurons using the layer-wise SGD with the finite difference loss, Algorithm 1. Then, there exists a constant $C > 0$, which can only depend on k, $\|f\|_\infty$, and $\min_{i \in S} \mathrm{Inf}_i(f)$, such that for all small enough $\varepsilon, \delta > 0$ if we set:*

- *At initialization, the scale $\kappa = \Theta_d(1)$ and the bias range $A = \Theta_k\left(\frac{\|f\|_\infty}{\varepsilon}\right)$.*
- *In Phase I, the batch size is $B = \Omega_k\left(\frac{d}{\varepsilon^2}\right)$, and the step size is $\gamma_1 = \frac{\sqrt{2Bd}}{\kappa\sqrt{k}}$, with $T_1 = 1$.*
- *In Phase II, the step size is $\gamma_2 = \tilde{O}_k\left(\frac{\delta^2 \varepsilon^4}{d}\right)$ with regularization $\lambda = \Theta_k\left(\delta\varepsilon^2\right)$ and $T_2 = \tilde{\Omega}_k\left(\frac{d}{\varepsilon^{18}\delta^3}\right)$.*

Then, with probability at least $1 - C\varepsilon$ over the randomness of Phase I and $\hat{b}$, the algorithm will output a network such that:

$$\mathbb{E}_{x \sim \mathrm{Unif}\{\pm 1\}^d}\left[\left(\mathrm{NN}(x; \theta^{(T_1+T_2)}) - f(x)\right)^2\right] < \delta. \tag{3}$$

To understand the sample complexity of Algorithm 1, we note that Phase I uses a single batch of size $B = \Omega_k(d)$, while Phase II requires $T_2 = \tilde{\Omega}_k(d)$ different steps of the random walk. Taken together, and ignoring the dependence on ε and δ, the algorithm requires running the random walk for $\tilde{\Omega}_k(d)$ steps. As mentioned above, a simple coupon-collector argument shows that this bound is near optimal. We also note that the dependence on k is exponential, as is natural for arbitrary k-juntas: even once the support is known, one must learn an arbitrary Boolean function on 2^k input patterns. This means that, in principle, we could allow $k = c\log(d)$, for a small enough constant $c > 0$, and still obtain effective results.

Ideas of the proof: The proof has three main components, corresponding to the two phases of the algorithm and to the non-degeneracy needed to connect them. To understand the first phase, consider the case in which the step from x_t to x_{t+1} flips a coordinate outside the support S. In this case, $f(x_{t+1}) - f(x_t) = 0$, and so the TD loss mostly reacts to moves in relevant coordinates. From this we show that the gradient updates $\nabla_{w^{(t)}} \mathcal{L}_1^{\mathrm{TD}}(S^{(t)}; w^{(t)}, a^{(0)}, b^{(0)})$ are essentially supported on S. Thus, at the end of Phase 1, the algorithm has extracted the relevant support information, since $w^{(T_1)}$ is concentrated on the coordinates in S.

Given the weights $w^{(T_1)}$ produced after Phase 1, the second layer is now a linear combination of neurons which essentially depend only on the coordinates in S. In light of this, the idea in Phase 2 is to train the second layer to solve a linear regression problem on the support variables. However, there is an important caveat here. For this regression problem to be well-defined, and for a solution to exist, we must be able to represent f using the weights $w^{(T_1)}$, suitable biases, and the ReLU function. Addressing this point is the most delicate and technical part of the proof, since, as we show, there exist possible choices of $w^{(T_1)}$ which, on one hand, reveal the relevant coordinates, yet cannot represent the function f. To circumvent this issue, we prove that these potential bad weights are only encountered with small probability. Thus, we establish an anti-concentration estimate for the batched first-layer update, viewed as an additive functional of the lazy random walk. Our key technical contribution, which may be of independent interest elsewhere, for this problem is a quantitative central limit theorem, based on [Klo19], which allows us to compare this update to a Gaussian and rule out the bad configurations in which the regression problem becomes ill-posed.

The final component of the proof focuses on efficiently solving the resulting linear regression problem. Most results in the literature assume that the samples are drawn i.i.d. from some underlying distribution, whereas in our setting, the samples still follow a random walk. Thus, even after establishing well-posedness, we still need to analyze the complexity of SGD in the presence of

temporal correlations. This is done by adapting recent arguments on SGD under Markovian sampling from [Eve23] for a strongly convex target function. This is facilitated by the strongly convex regularization we add in Phase II. Alternatively, we could appeal to the results of [SSY18], which does not require strong convexity, and so does not require the regularization, but only applies to the average weight, rather than the terminal point.

5 Noisy SGD Lower Bounds for Standard Losses

In this section, we prove a negative result for SGD with large batch size when training with any convex *pointwise* loss, such as the square loss or the hinge loss. This further highlights the role of the temporal-difference loss in Theorem 2. Here, by a pointwise loss, we mean a loss that evaluates the discrepancy between the predictor and the target on a single example. The lower bound applies to the class of *noisy* gradient-based methods, considered also in previous works [AS20a, MKAS21], which we formally introduce in Definition 3.

Definition 3 (Noisy-SGD)**.** *Let* $\mathrm{NN}(\cdot;\theta)$ *be a neural network with initialization* $\theta^{(0)}$*, and let* $\{\xi^{(t)}\}_{t\in[T]}$ *be a training sequence, where each* $\xi^{(t)}$ *is a sample (or tuple of samples) from which the loss is computed. This formulation covers both standard pointwise losses, where* $\xi^{(t)} = (x^{(t)}, f(x^{(t)}))$*, and temporal-difference losses, where* $\xi^{(t)}$ *may contain two consecutive samples, e.g.* $\xi^{(t)} = (x^{(t-1)}, x^{(t)}, f(x^{(t-1)}), f(x^{(t)}))$*. Let* $L(\theta;\xi)$ *be an almost surely differentiable loss evaluated at parameter* θ *and sample* ξ*, and let* $R(\theta)$ *be an almost surely differentiable regularizer. For* $\lambda \geq 0$*, define* $L_\lambda(\theta;\xi) := L(\theta;\xi) + \lambda R(\theta)$*. Given learning rates* $\{\gamma_t\}_{t\geq 0}$*, batch size* B*, gradient range* $A > 0$*, and noise level* $\tau \geq 0$*, the updates of noisy SGD are defined by*

$$\theta^{(t+1)} = \theta^{(t)} - \gamma_t \left(\frac{1}{B} \sum_{s=1}^{B} \left[\nabla_\theta L_\lambda(\theta^{(t)}; \xi^{(t,s)}) \right]_A + Z^{(t)} \right), \tag{4}$$

where for all $t \in \{0, \ldots, T-1\}$*,* $Z^t \sim \mathcal{N}(0, \tau^2 I_d)$*, for some* noise level τ*, and they are independent from other variables,* B *is the batch size, and* $[z]_A := \mathrm{argmin}_{|y|\leq A} |y - z|$*, i.e., whenever the gradient exceeds* A *(resp.* $-A$*) it is rounded to* A *(resp.* $-A$*).*

To establish our result, we extend the lower bound of [AS20a, Theorem 3], originally proved for i.i.d. data, to the case where the training samples are generated by a random walk. Such lower bound holds for *classes of functions*, which leads us to define the orbit of a fixed function. For a Boolean function $f : \{\pm 1\}^d \to \mathbb{R}$, and for a permutation $\pi \in \mathcal{S}_d$, define the action of π on $x \in \{\pm 1\}^d$ by $(\pi \cdot x)_i := x_{\pi(i)}$, $i \in [d]$. The *orbit* of f under coordinate permutations is:

$$\mathrm{orb}(f) := \{\, f \circ (\pi \cdot) \;:\; \pi \in \mathcal{S}_d \,\}.$$

We will also use the following notion of cross-predictability, introduced in [AS20a].

Definition 4 (Cross-Predictability (CP),[AS20a])**.** *Let* $P_{\mathcal{F}}$ *be a distribution over functions from* $\{\pm 1\}^d$ *to* $\mathbb{R}$ *and* $P_{\mathcal{X}}$ *be a distribution over* $\{\pm 1\}^d$*. Then,* $\mathrm{CP}(P_{\mathcal{X}}, P_{\mathcal{F}}) = \mathbb{E}_{F,F'\sim P_{\mathcal{F}}} \left[\mathbb{E}_{x\sim P_{\mathcal{X}}} [F(x)F'(x)]^2 \right]$.

We may now state our theorem.

Theorem 3 (Noisy-SGD with large random-walk batches)**.** *for* $T \in \mathbb{N}$ *let* $(x^{(s)}, y^{(s)})_{s\in[m]}$ *be such that* $\{x^{(s)}\}_{s\in[m]}$ *is generated by the* p*-lazy random walk on* $\{\pm 1\}^d$ *(see Def. 1), and* $y^{(s)} = f(x^{(s)})$*, for* $f : \{\pm 1\}^d \to \{\pm 1\}$ *a balanced Boolean function, i.e. such that* $\mathbb{E}_{x\sim\mathcal{U}_d}[f(x)] = o_d(1)$*, where* $\mathcal{U}_d := \mathrm{Unif}\{\pm 1\}^d$*. Consider training a fully connected neural network of size* M *and i.i.d. initialization with noisy-SGD (see Def. 3) with any pointwise convex and almost surely differentiable loss, noise level* $\tau > 0$*, gradient range* A*, and batch size* B*. Then, after* T *steps of training,*

$$\mathbb{P}\left(\mathrm{sgn}(\mathrm{NN}(x;\theta^{(T)})) = f(x) \right) \leq \frac{1}{2} + \frac{T\sqrt{M}A}{2\tau} \left(\mathrm{CP}(\mathcal{U}_d, \mathrm{orb}(f)) + \frac{d}{pB} \right)^{1/4}. \tag{5}$$

The theorem establishes the potential hardness of weak learning, i.e. of obtaining a predictor that achieves accuracy better than a random guess. The statement applies to arbitrary balanced Boolean functions, but it is especially meaningful when the function class is known to be hard to weakly learn,

i.e. when the cross-predictability of the orbit is small, such as for parities. In such cases, Theorem 3 states that if the batch size is large, so that the right-most factor in (5) is small, then noisy-SGD with bounded gradient precision A/τ needs either a large number of training iterations T, or a network of large size M, to perform better than random guess.

Remark 1 (Large batch size). The lower bound applies only in the large-batch regime. The intuition is that, when each SGD step averages gradients over many consecutive samples from the random-walk trajectory, the temporal correlations within the batch are largely washed out, provided they decay sufficiently fast. As a result, the averaged gradient becomes close to the one that would be obtained from the stationary distribution of the walk, which in our setting is the uniform distribution on the hypercube. The additional term $d/(pB)$ measures the residual dependence between samples within a batch drawn from the random walk, which is negligible only if B is large. Therefore, large-batch SGD with a pointwise loss does not significantly exploit temporal correlations, and one recovers essentially the same hardness barrier as in the i.i.d. uniform setting.

Let us now consider the case of parities. Recall that Theorem 2 shows that, for every fixed k, a two-layer ReLU network of size $M = \Theta(d)$, trained by Algorithm 1 with batch size $B = \Omega_k(d)$, learns any k-parity in $O_k(d)$ steps. In contrast, consider SGD with a standard pointwise loss on the same network of size $M = \Theta(d)$, with constant gradient precision $A/\tau = O_d(1)$, constant laziness parameter $p = \Theta(1)$, and batch size $B = d^c$, for some $c > 0$. Since the orbit of a fixed k-parity is the class of all k-parities, $\mathrm{CP}(\mathcal{U}_d, P_k) = \binom{d}{k}^{-1}$. Therefore, Theorem 3 implies that any such method achieving constant advantage $\eta > 0$ over random guessing must satisfy $T \geq \Omega_{k,\eta}\left(d^{\frac{\min\{k,c-1\}}{4} - \frac{1}{2}}\right)$. In particular, if $\min\{k, c-1\} > 6$, then pointwise-loss SGD requires superlinear time to beat random guessing by a constant amount.

6 Numerical Experiments

To demonstrate our theoretical results, we ran several experiments in dimension $d = 50$. In all experiments we train a 4-layer MLP with $512, 1024, 64$ hidden neurons and ReLU activation, by SGD with all layers trained jointly, batch size 1 and learning rate 0.005. We use either the temporal difference (TD) loss (Def. 2) with $\alpha = 0.9$ or the standard square loss. The inputs are either sampled online i.i.d. from the uniform distribution on the hypercube, or generated by a lazy random walk with $p = 0.9$. The test loss and accuracy are computed on the standard uniform distribution. We repeat each experiment for 5 seeds, and we report one standard deviation intervals.

In Figure 1, we consider learning the 5-parity $f(x) = x_1 x_2 \cdots x_5$. The left panel reports test accuracy as a function of the number of fresh samples observed during training. With random-walk data and the TD loss, the network reaches high test accuracy with moderate sample complexity, whereas the corresponding i.i.d. baselines fail to learn anything significant. The right panel tracks selected Fourier-Walsh coefficients of the learned predictor during training, for the random-walk experiment with TD loss. Specifically, we plot $\mathbb{E}_x[\mathrm{NN}(x;\theta^t)x_1]$ in orange, $\mathbb{E}_x[\mathrm{NN}(x;\theta^t)x_6]$ in green, and the target coefficient $\mathbb{E}_x[\mathrm{NN}(x;\theta^t)\prod_{i=1}^5 x_i]$ in blue, where expectations are taken over the uniform

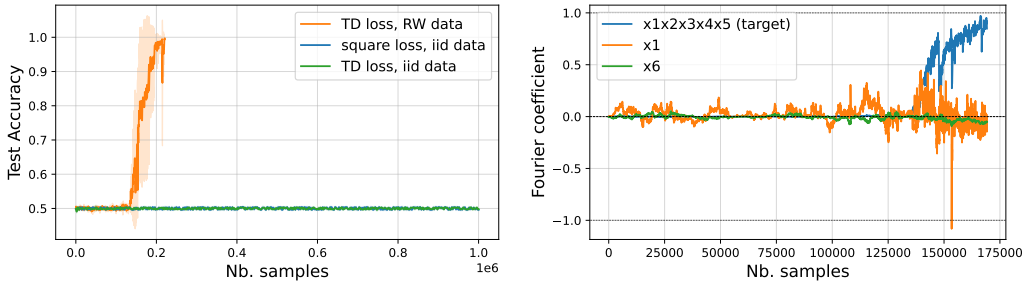

Figure 1: 5-parity with $d = 50$. We train a 4-layer MLP with batch size 1, learning rate 0.005, TD parameter $\alpha = 0.9$, and random-walk flip probability $p = 0.9$. Left: test accuracy for random-walk data with TD loss, compared with i.i.d. data using either TD loss or square loss. Right: selected Fourier-Walsh coefficients of the learned predictor for random-walk data with TD loss.

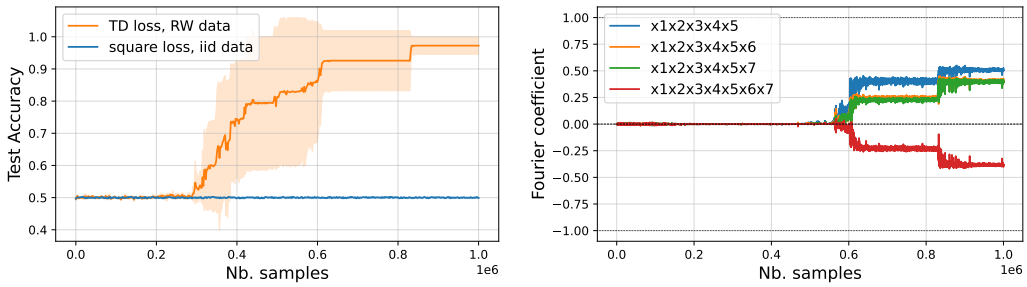

Figure 2: $f(x) = \frac{1}{2}x_1x_2...x_5(1 + x_6 + x_7 - x_6x_7)$ with $d = 50$. We train a 4-layer MLP with batch size 1, learning rate 0.005, TD parameter $\alpha = 0.9$, and random-walk flip probability $p = 0.9$. Left: test accuracy for random-walk data with TD loss, compared with i.i.d. data using square loss. Right: selected Fourier-Walsh coefficients of the learned predictor for random-walk data with TD loss.

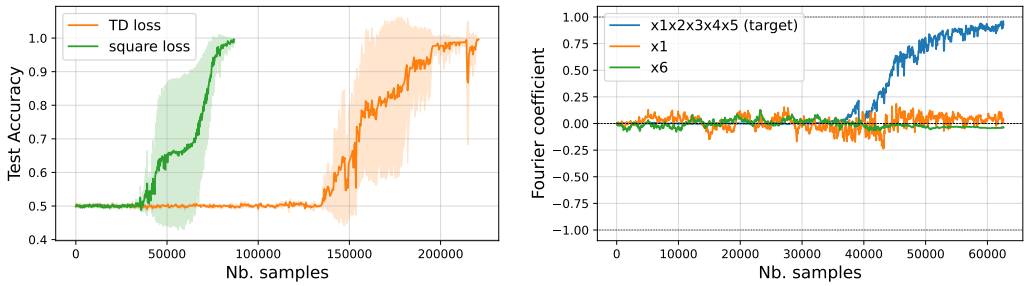

Figure 3: 5-parity with $d = 50$. We train a 4-layer MLP with batch size 1, learning rate 0.005, and random-walk flip probability $p = 0.9$. Left: test accuracy for random-walk data using TD loss versus square loss. Right: selected Fourier-Walsh coefficients of the learned predictor for random-walk data with square loss.

distribution. The first-order coefficients remain centered around zero, both for a coordinate inside the support (x_1) and outside the support (x_6). However, the in-support coefficient (orange) exhibits larger fluctuations than the out-of-support one (green). This variance anisotropy is consistent with our theoretical analysis: the support information is not visible through first-order means, but appears through second-moment effects induced by the random-walk dynamics.

Figure 2 repeats the same experiment for the 7-junta $f(x) = \frac{1}{2}x_1x_2 \cdots x_5(1 + x_6 + x_7 - x_6x_7)$, with the same qualitative conclusion. In the right panel, we observe that the relevant Fourier-Walsh coefficients of the learned predictor emerge at roughly the same time. This contrasts with the behavior typically seen under i.i.d. uniform sampling, where lower-order terms are learned before higher-order ones [AAM23].

In Figure 3, we replace the temporal-difference loss with the standard square loss. Since the experiment uses batch size one, our large-batch lower bound (Theorem 3) does not apply. In this regime, SGD with the square loss successfully learns the parity from random-walk data. This suggests that the small-batch square-loss regime can behave qualitatively differently from the large-batch regime covered by our theory. Interestingly, in these experiments, the square loss alone appears sufficient to exploit the temporal structure, and can even outperform the temporal-difference loss. Understanding this phenomenon theoretically is an interesting direction for future work.

7 Conclusion

In this work, we studied how temporal correlations can fundamentally change the complexity of learning some sparse tasks, such as k-juntas. Our main positive result shows that, when the data are generated by a lazy random walk on the hypercube, a two-layer ReLU network trained by SGD with a temporal-difference loss can learn Boolean k-juntas with linear sample complexity. This shows that

the temporal structure of the data can be exploited by SGD to bypass the hardness barriers known in the i.i.d. setting. On the negative side, we proved that this advantage does not extend to standard convex pointwise losses in the noisy large-batch regime. Our work leaves several important directions open. First, our lower bound for pointwise losses does not cover the small-batch regime, and our numerical experiments suggest that standard small-batch SGD with square loss may still exploit temporal correlations. Understanding this phenomenon theoretically, and more broadly clarifying how learnability depends on the batch size in the presence of temporal correlations, is an interesting next step. Secondly, it would be interesting to extend our results to more general forms of correlated data, beyond single-bit-flip dynamics, and to identify which qualitative properties of temporal correlations influence the learning dynamics.

8 Acknowledgments

DM was partially supported by the Brian and Tiffinie Pang Faculty Fellowship. EM was partially supported by NSF DMS-2031883, Bush Faculty Fellowship ONR-N00014-20-1-2826 and Simons Investigator award. This work started while EC was visiting MIT for a collaboration visit supported by NSF DMS-2031883.

References

[AAM23] Emmanuel Abbe, Enric Boix Adsera, and Theodor Misiakiewicz. SGD learning on neural networks: leap complexity and saddle-to-saddle dynamics. In *The Thirty Sixth Annual Conference on Learning Theory*, pages 2552–2623. PMLR, 2023.

[ABAM22] Emmanuel Abbe, Enric Boix-Adsera, and Theodor Misiakiewicz. The merged-staircase property: a necessary and nearly sufficient condition for sgd learning of sparse functions on two-layer neural networks. In *Conference on Learning Theory*, pages 4782–4887. PMLR, 2022.

[ABLR23] Emmanuel Abbe, Samy Bengio, Aryo Lotfi, and Kevin Rizk. Generalization on the unseen, logic reasoning and degree curriculum. In *International Conference on Machine Learning*, pages 31–60. PMLR, 2023.

[ACL23] Emmanuel Abbe, Elisabetta Cornacchia, and Aryo Lotfi. Provable advantage of curriculum learning on parity targets with mixed inputs. *Advances in Neural Information Processing Systems*, 36:24291–24321, 2023.

[AGJ21] Gerard Ben Arous, Reza Gheissari, and Aukosh Jagannath. Online stochastic gradient descent on non-convex losses from high-dimensional inference. *The Journal of Machine Learning Research*, 22(1):4788–4838, 2021.

[AGRK20] Federico Amato, Fabian Guignard, Sylvain Robert, and Mikhail Kanevski. A novel framework for spatio-temporal prediction of environmental data using deep learning. *Scientific reports*, 10(1):22243, 2020.

[AM08] Jan Arpe and Elchanan Mossel. Agnostically learning juntas from random walks. *arXiv preprint arXiv:0806.4210*, 2008.

[AS20a] Emmanuel Abbe and Colin Sandon. On the universality of deep learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 20061–20072, 2020.

[AS20b] Emmanuel Abbe and Colin Sandon. Poly-time universality and limitations of deep learning. arXiv:2001.02992, 2020.

[AV90] David Aldous and Umesh Vazirani. A markovian extension of valiant's learning model. *Information and Computation*, 85(2):127–133, 1990.

[AZLS19] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. In *International conference on machine learning*, pages 242–252. PMLR, 2019.

[BB20] Alberto Bietti and Francis Bach. Deep equals shallow for relu networks in kernel regimes, 2020.

[BBSS22] Alberto Bietti, Joan Bruna, Clayton Sanford, and Min Jae Song. Learning single-index models with shallow neural networks. *Advances in Neural Information Processing Systems*, 35:9768–9783, 2022.

[BEG+22] Boaz Barak, Benjamin L Edelman, Surbhi Goel, Sham Kakade, Eran Malach, and Cyril Zhang. Hidden progress in deep learning: Sgd learns parities near the computational limit. *arXiv preprint arXiv:2207.08799*, 2022.

[BFH02] Peter Bartlett, Paul Fischer, and Klaus-Uwe Höffgen. Exploiting random walks for learning. *Information and Computation*, 176(2):121–135, 2002.

[BGS18] Guy Bresler, David Gamarnik, and Devavrat Shah. Learning graphical models from the glauber dynamics. *IEEE Transactions on Information Theory*, 64:4072–4080, 2018.

[BMOS05] Nader H Bshouty, Elchanan Mossel, Ryan O'Donnell, and Rocco A Servedio. Learning dnf from random walks. *Journal of Computer and System Sciences*, 71(3):250–265, 2005.

[CM23] Elisabetta Cornacchia and Elchanan Mossel. A mathematical model for curriculum learning for parities. In *International Conference on Machine Learning*, pages 6402–6423. PMLR, 2023.

[CMM25] Elisabetta Cornacchia, Dan Mikulincer, and Elchanan Mossel. Low-dimensional functions are efficiently learnable under randomly biased distributions. *arXiv preprint arXiv:2502.06443*, 2025.

[DLL+19] Simon Du, Jason Lee, Haochuan Li, Liwei Wang, and Xiyu Zhai. Gradient descent finds global minima of deep neural networks. In *International conference on machine learning*, pages 1675–1685. PMLR, 2019.

[DLS22] Alexandru Damian, Jason Lee, and Mahdi Soltanolkotabi. Neural networks can learn representations with gradient descent. In *Conference on Learning Theory*, pages 5413–5452. PMLR, 2022.

[DM20] Amit Daniely and Eran Malach. Learning parities with neural networks. *Advances in Neural Information Processing Systems*, 33:20356–20365, 2020.

[DTA+24] Yatin Dandi, Emanuele Troiani, Luca Arnaboldi, Luca Pesce, Lenka Zdeborová, and Florent Krzakala. The benefits of reusing batches for gradient descent in two-layer networks: Breaking the curse of information and leap exponents. *arXiv preprint arXiv:2402.03220*, 2024.

[ELSW05] Ariel Elbaz, Homin K Lee, Rocco A Servedio, and Andrew Wan. Separating models of learning from correlated and uncorrelated data. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC)*, pages 111–120, 2005.

[Eve23] Mathieu Even. Stochastic gradient descent under markovian sampling schemes. In *International Conference on Machine Learning*, pages 9412–9439. PMLR, 2023.

[Gla23] Margalit Glasgow. Sgd finds then tunes features in two-layer neural networks with near-optimal sample complexity: A case study in the xor problem. *arXiv preprint arXiv:2309.15111*, 2023.

[GLK+20] Federica Gerace, Bruno Loureiro, Florent Krzakala, Marc Mézard, and Lenka Zdeborová. Generalisation error in learning with random features and the hidden manifold model. In *International Conference on Machine Learning*, pages 3452–3462. PMLR, 2020.

[GMKZ20] Sebastian Goldt, Marc Mézard, Florent Krzakala, and Lenka Zdeborová. Modeling the influence of data structure on learning in neural networks: The hidden manifold model. *Physical Review X*, 10(4):041044, 2020.

[GMM24] Jason Gaitonde, Ankur Moitra, and Elchanan Mossel. Bypassing the noisy parity barrier: Learning higher-order markov random fields from dynamics. *arXiv*, 2024.

[JMS24] Nirmit Joshi, Theodor Misiakiewicz, and Nathan Srebro. On the complexity of learning sparse functions with statistical and gradient queries. *Advances in Neural Information Processing Systems*, 37:103198–103241, 2024.

[JW14] Jeffrey C Jackson and Karl Wimmer. New results for random walk learning. In *Conference on Learning Theory (COLT)*, pages 1051–1069, 2014.

[KCGK24] Yiwen Kou, Zixiang Chen, Quanquan Gu, and Sham M Kakade. Matching the statistical query lower bound for k-sparse parity problems with sign stochastic gradient descent. *Advances in Neural Information Processing Systems*, 37:113001–113037, 2024.

[Kea98] Michael Kearns. Efficient noise-tolerant learning from statistical queries. *Journal of the ACM (JACM)*, 45(6):983–1006, 1998.

[Klo19] Benoît Kloeckner. Effective berry–esseen and concentration bounds for markov chains with a spectral gap. *The Annals of Applied Probability*, 29(3):1778–1807, 2019.

[Klo21] Benoît Kloeckner. Toy examples for effective concentration bounds. *arXiv preprint arXiv:2102.09215*, 2021.

[MAC+26] Marko Medvedev, Idan Attias, Elisabetta Cornacchia, Theodor Misiakiewicz, Gal Vardi, and Nathan Srebro. Positive distribution shift as a framework for understanding tractable learning. *arXiv preprint arXiv:2602.08907*, 2026.

[MKAS21] Eran Malach, Pritish Kamath, Emmanuel Abbe, and Nathan Srebro. Quantifying the benefit of using differentiable learning over tangent kernels. In *International Conference on Machine Learning*, pages 7379–7389. PMLR, 2021.

[Oll09] Yann Ollivier. Ricci curvature of markov chains on metric spaces. *Journal of Functional Analysis*, 256(3):810–864, 2009.

[PGM+19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[PLM+24] Abhishek Panigrahi, Bingbin Liu, Sadhika Malladi, Andrej Risteski, and Surbhi Goel. Progressive distillation induces an implicit curriculum. *arXiv preprint arXiv:2410.05464*, 2024.

[RLRL24] Elham Ravanbakhsh, Yongqing Liang, J Ramanujam, and Xin Li. Deep video representation learning: a survey. *Multimedia Tools and Applications*, 83(20):59195–59225, 2024.

[SSBD14] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[SSSS17] Shai Shalev-Shwartz, Ohad Shamir, and Shaked Shammah. Failures of gradient-based deep learning. In *International Conference on Machine Learning*, pages 3067–3075. PMLR, 2017.

[SSY18] Tao Sun, Yuejiao Sun, and Wotao Yin. On markov chain gradient descent. *Advances in neural information processing systems*, 31, 2018.

[Sut88] Richard S Sutton. Learning to predict by the methods of temporal differences. *Machine learning*, 3(1):9–44, 1988.

[Val84] Leslie G Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

[YCC+22] Huaxiu Yao, Caroline Choi, Bochuan Cao, Yoonho Lee, Pang Wei W Koh, and Chelsea Finn. Wild-time: A benchmark of in-the-wild distribution shift over time. *Advances in Neural Information Processing Systems*, 35:10309–10324, 2022.

A Proof of Theorem 2

Without loss of generality we assume that the support $S = [k]$, i.e. that the target junta is supported on the first k coordinates.

A.1 Initial gradient - results of Phase I

The training of the first layer is performed via batched SGD with respect to the temporal difference loss. That is, let $B \in \mathbb{N}$ and let $\{x^{(t)}\}_{t=0}^{B}$ be the initial segment of the random walk. Define, $\mathcal{L}_\alpha^{(t)}(\theta^{(0)}) := \mathcal{L}_\alpha^{TD}\left(f, \mathrm{NN}(.;\theta^{(0)}), x^{(t)}, x^{(t-1)}\right)$. Then,

$$\begin{aligned} w^{(1)} &= w^{(0)} - \frac{\gamma_1}{B}\sum_{t=1}^{B} \nabla_w \mathcal{L}_\alpha^{(t)}(\theta^{(0)}) \\ &= w^{(0)} + G(B), \end{aligned}$$

where we defined:

$$G(B) = -\frac{\gamma_1}{B}\sum_{t=1}^{B} \nabla_w \mathcal{L}_\alpha^{(t)}(\theta^{(0)}). \tag{6}$$

Note that by our initialization, $G(B)_{ij}$ does not depend on the hidden unit $i \in [N]$. In the following, we drop the dependence on i, to simplify our notation. Note that:

$$\begin{aligned} \nabla(\Delta\,\mathrm{NN}(.;\theta^{(0)})^{(t)}) &= a^{(0)}(\mathbb{1}((w^{(0)})^T x^{(t)} + b \geq 0)x^{(t)} - \mathbb{1}((w^{(0)})^T x^{(t-1)} + b \geq 0)x^{(t-1)}) \\ &= \kappa(x^{(t)} - x^{(t-1)}). \end{aligned}$$

Further, since $w_i^{(0)} = 0$,

$$\forall x \in \{-1,1\}^d, \quad \mathrm{NN}(x;\theta^{(0)}) = \kappa\sum_{i=1}^{N}\mathrm{ReLU}(b_i) \implies \Delta\,\mathrm{NN}(\cdot;\theta^{(0)})^{(t)} = 0.$$

Recall that to simplify matters, we chose $\alpha = 1$ for the first part of training on the first layer, which will nullify the gradient in the irrelevant coordinates. Note that we could have just as well chosen α sufficiently close to 1 at the cost of complicating the proof. In the case of $\alpha = 1$ we then have:

$$\begin{aligned} \nabla_w \mathcal{L}_1^{(t)}(\theta^{(0)}) &= -\kappa(x^{(t)} - x^{(t-1)})\Big(\Delta f^{(t)} - \Delta\,\mathrm{NN}(.;\theta^{(0)})^{(t)}\Big) \\ &= -\kappa(x^{(t)} - x^{(t-1)})\Delta f^{(t)} \end{aligned}$$

Thus,

$$G(B) = \frac{\kappa\gamma_1}{B}\sum_{t=1}^{B}(x^{(t)} - x^{(t-1)})\Delta f^{(t)}. \tag{7}$$

Because $x^{(t)}$ evolves via the lazy hypercube walk (Def. 1), the increment is $x^{(t+1)} - x^{(t)} = -2Z_t x_{j_t}^{(t)} \cdot e_{j_t}$. Note that if $j_t \notin [k]$, then f does not depend on that coordinate, thus $\Delta f^{(t)} = 0$. Therefore, the weights update only when the walk flips a relevant coordinate, and in that case, the update affects only coordinates in the support of the k-junta:

$$w_j^{(1)} = (G(B))_j = \begin{cases} \dfrac{\kappa\gamma_1}{B}\displaystyle\sum_{t=1}^{B}\Delta f^{(t)}\big(x_j^{(t)} - x_j^{(t-1)}\big), & \text{if } j \in [k], \\ 0, & \text{if } j \notin [k]. \end{cases} \tag{8}$$

A.1.1 Non-degeneracy of the initial gradient

With these choices, it should be rather clear that as long as the batch size B is sufficiently large, $w^{(1)}$ will precisely recover the support of f. Thus, let $x \to E(x) := \{\sigma((w^{(1)})^T x + b_i)\}_{i=1}^N$ be the embedding from $\mathbb{R}^d$ to $\mathbb{R}^N$ obtained by the first layer of the network. The above observation on the gradient can be rephrased as saying that E induced a map from functions $\{-1,1\}^d$ to functions on $\{-1,1\}^k$. Thus, to fully recover f it should be enough to optimize over the second layer of $\mathrm{NN}(\cdot;\theta^{(1)})$, which reduces to a linear regression problem. Concretely, we shall need to find some $a \in \mathbb{R}^N$ such that for $\theta = (w^{(1)}, a)$ we have

$$\forall x \in \{-1,1\}^d \qquad \mathrm{NN}(x,\theta) = a^T E(x) = f(x). \tag{9}$$

To make this linear problem well-posed though, we require a non-degeneracy condition on $\mathrm{NN}(\cdot;\theta^{(1)})$. Suppose we have two points $s, r \in \{-1,1\}^k$ such that $\tilde{f}(s) \neq \tilde{f}(r)$ yet $(w^{(1)})^T s = (w^{(1)})^T r$. Then, for any choice of $a \in \mathbb{R}^N$ we will have $a^T E(s) = a^T E(r)$, making the above program unfeasible. Our main technical condition for the first stage of the algorithm is hence the following condition on the initial gradient. Recall below that $\tau = \min_{j\in[k]} \mathrm{Inf}_j(f)$ is the minimal influence on the support.

Proposition 1 (Non-degeneracy of the initial gradient)**.** *Assume the data is generated by the lazy random walk of Definition 1 with $p \in [0,1/2]$ and that $f : \{\pm 1\}^d \to \mathbb{R}$ is a k-junta supported on $[k]$. For $B \geq 1$, let $G(B) = (G_j(B))_{j\in[d]}$ be the gradient defined in (8). Let $\eta = \frac{\kappa\gamma_1}{B}$, so that*

$$G(B) = \eta \sum_{t=1}^{B} \Delta f^{(t)}(x^{(t)} - x^{(t-1)}).$$

Then there exists a universal constant $C > 0$ such that, as long as,

$$B \geq B_0(\varepsilon) := \frac{C}{\varepsilon^2} \frac{dk^{16}\|f\|_\infty^6}{p^7\tau^3} \quad \text{and} \quad \eta \geq \sqrt{\frac{2d}{Bk}},$$

for every pair $s, r \in \{\pm 1\}^k$ with $\tilde{f}(s) \neq \tilde{f}(r)$, we have

$$\mathbb{P}\Big(\Big|\sum_{j=1}^{k} G_j(B)\,(s_j - r_j)\Big| > \varepsilon\Big) \;\geq\; 1 - \varepsilon\sqrt{\frac{Ck}{p\tau}}, \tag{10}$$

where the probability is over the randomness of the random walk.

Our proof of Proposition 1 goes through a quantitative version of the central limit theorem for Markov chains, introduced in [Klo19]. To apply this result, we consider the set

$$\mathcal{V} := \{\, v = s - r : s, r \in \{\pm 1\}^k,\ \tilde{f}(s) \neq \tilde{f}(r) \,\} \subseteq \{0, \pm 2\}^k,$$

and fix $s, r \in \{\pm 1\}^k$ such that $v = s - r \in \mathcal{V}$. Define the scalar projection of the gradient:

$$\Pi_B(v) := \sum_{j=1}^{k} G_j(B) v_j. \tag{11}$$

By (8),

$$\Pi_B(v) = \eta \sum_{u=1}^{B} \Delta f^{(u)} \sum_{j=1}^{k} (x_j^{(u)} - x_j^{(u-1)}) v_j$$

Further define the process

$$Y_u := \Delta f^{(u)} \sum_{j=1}^{k} (x_j^{(u)} - x_j^{(u-1)}) v_j, \qquad \overline{Y}_B := \eta \sum_{u=1}^{B} Y_u,$$

for which

$$\Pi_B(v) = \overline{Y}_B. \tag{12}$$

We rewrite $\overline{Y}_B$ as an additive functional of simple Markov chain as follows. Consider the Markov chain

$$Z_u := (x^{(u-1)}, x^{(u)}) \in \Omega := \{\pm 1\}^d \times \{\pm 1\}^d.$$

Since $(x^{(u)})_{u\geq 0}$ is a homogeneous lazy random walk with stationary law $\mathrm{Unif}\{\pm 1\}^d$ and we start from $x^{(0)} \sim \mathrm{Unif}\{\pm 1\}^d$, the chain (Z_u) is an *edge random walk*, stationary with respect to some measure μ_0 on Ω. Define the observable

$$\varphi_v(x,y) := \big(f(y) - f(x)\big) \cdot \sum_{j=1}^{k} (y_j - x_j) v_j, \qquad (x,y) \in \Omega.$$

Then $Y_u = \varphi_v(Z_u)$ and $\overline{Y}_B = \eta \sum_{u=1}^{B} \varphi_v(Z_u)$. To avoid having to depend on the ambient dimension d, let us note that

$$\mathrm{support}(x^{(u)} - x^{(u-1)}) \cap [k] = \emptyset \implies \varphi_v(Z_u) = 0.$$

We can thus couple $\overline{Y}_B$ with an additive function of a k-dimensional random walk as follows: Let Z'_u be an edge random walk on $\{\pm 1\}^k \times \{\pm 1\}^k$, and set $Y'_u = \varphi_v(Z'_u)$. By the above observation $Y'_u = Y_u$ conditioned on $\mathrm{support}(x^{(u)} - x^{(u-1)}) \cap [k] \neq \emptyset$ while also accounting for lazy steps in this direction, and so Y'_u is well-defined. For $B \in \mathbb{N}$ we are thus led to consider the normalized lower-dimensional function

$$\overline{Y}'_B = \frac{1}{\sqrt{B}} \sum_{u=1}^{B} Y'_u.$$

Further, recall from Definition 1 that j_u is the proposed flip of the random walk at step u, and define the random variable

$$B_k := \#\{0 \leq u \leq B \mid j_u \in [k]\}.$$

That is, the number of steps in which Z_u evolved along the first k coordinates, where we also include the lazy steps that did not materialize to a flip. With respect to these counts, we now have the following equality in law

$$\overline{Y}_B = \eta \sqrt{B_k} \overline{Y}'_{B_k}.$$

Let

$$m_v := \mathbb{E}[\varphi_v(Z'_1)], \qquad \sigma_v^2 := \sigma^2(\varphi_v),$$

where $\sigma^2(\cdot)$ is the dynamical variance, given by

$$\sigma_v^2 = \lim_{T\to\infty} \mathrm{Var}(\overline{Y}'_T).$$

The proof of Proposition 1 crucially relies on the following fact, that for B large enough (and hence also for B_k large enough) $\overline{Y}'_{B_k}$ is essentially Gaussian.

Lemma 1. *Let G be a standard Gaussian variable. There exists a universal constant $C > 0$ such that for any $v \in \mathcal{V}$ and $x \in \mathbb{R}$,*

$$\sup_{x\in\mathbb{R}} \left| \mathbb{P}\left(\sigma_v^{-1} \left(\overline{Y}'_T - \mathbb{E}[\overline{Y}'_T] \right) \leq x \right) - \mathbb{P}\left(G \leq x\right) \right| \leq C \frac{k^7 \|f\|_\infty^3}{p^2 \min(\sigma_v^3, \sigma_v)\sqrt{T}}.$$

We will prove Lemma 1 in Section B below. For now, to properly apply the Lemma, we will need to estimate the first moments of $\overline{Y}'_T$.

Lemma 2. *Let Y'_u be as above. Then,*

$$\mathbb{E}[Y'_u] = \frac{4p}{k} \sum_{j=1}^{k} v_j \hat{f}_j,$$

where $\hat{f}_j$ are the linear Fourier coefficients of f. In particular $\mathbb{E}[Y'_u] = O(p)$.

Proof. Let $Z' = (u, v)$ be drawn according to invariant measure, that is (u, v) is a random edge of $\{\pm 1\}^k \times \{\pm 1\}^k$. We have,

$$\mathbb{E}\left[\varphi_v(Z')\right] = \frac{2p}{k} \sum_{j=1}^{k} \mathbb{E}\left[(f(u) - f(u^{(j)}))u_j \right] v_j = \frac{4p}{k} \sum_{j=1}^{k} \mathbb{E}\left[D_j(v_j f(u)) u_j \right],$$

where u is uniform and $u^{(j)}$ is obtained from u by flipping the j'th coordinate. Integrating by parts on the discrete cube, we see

$$\sum_{j=1}^{n} \mathbb{E}\left[D_j(v_j f(u))u_j\right] = \sum_{j=1}^{d} \mathbb{E}\left[v_j f(u) u_j\right] = \sum_{j=1}^{d} v_j \hat{f}_j,$$

□

Lemma 3. *Suppose that $p \leq \frac{1}{2}$. Then, for any v, it holds that*

$$\mathrm{Var}(\varphi_v(Z_1')) \leq \sigma_v^2 \leq 2\frac{k}{p}\mathrm{Var}(\varphi_v(Z_1')).$$

Proof. We have that

$$\begin{aligned}\sigma_v^2 = \lim_{T\to\infty} \frac{1}{T}\mathrm{Var}\left(\sum_{u=1}^{T} \varphi_v(Z_u')\right) &= \mathrm{Var}(\varphi(Z_1')) + 2\sum_{u=1}^{\infty} \mathrm{Cov}(\varphi_v(Z_1'), \varphi_v(Z_u')) \\ &= \mathrm{Var}(\varphi(Z_1')) + 2\sum_{u=1}^{\infty} \mathrm{Cov}(\varphi_v(Z_1'), P_u\varphi_v(Z_1')).\end{aligned}$$

We shall start with the lower bound, which is the more difficult part. We claim that for any $u \geq 1$, $\mathrm{Cov}(\varphi_v(Z_1')\varphi_v(Z_u')) \geq 0$, the bound will immediately follow. Indeed, for z' and edge write $z' = (x, x')$, and similarly for the random walk $Z_u = (X_{u-1}, X_u)$. We first note that

$$h_v(x, x') := P_1\varphi_v(z) = \sum_{x''} \mathbb{P}\left(X_2 = x'' | X_1 = x'\right) \varphi_v(x', x'').$$

In particular, h_v is only a function of x' and is independent from x, and so we will instead write $h_v(x')$. Moreover, it is immediate to verify that $\varphi_v(x, x') = \varphi_v(x', x)$, and so

$$\mathbb{E}\left[\varphi_v(Z_1')|X_1\right] = \mathbb{E}\left[\varphi_v(X_0, X_1)|X_1\right] = \mathbb{E}\left[\varphi_v(X_0, X_1)|X_0\right],$$

i.e. conditioning on the past and on the present has the same effect. Now, for $u \geq 0$, since X_u is Markovian

$$\mathrm{Cov}(\varphi_v(Z_1'), \varphi_v(Z_u')) = \mathrm{Cov}(\mathbb{E}\left[\varphi_v(X_0, X_1)|X_1\right], \mathbb{E}\left[\varphi_v(X_{u-1}, X_u)|X_1\right]).$$

Because of the symmetry, as above, we have $\mathbb{E}\left[\varphi_v(X_0, X_1)|X_1\right] = h(X_1)$. Furthermore, by definition,

$$\mathbb{E}\left[\varphi_v(X_{u-1}, X_u)|X_1\right] = P_u(X_0, X_1) = P_{u-1}h(X_1).$$

Combining, we get

$$\mathrm{Cov}(\varphi_v(Z_1'), \varphi_v(Z_u')) = \mathrm{Cov}(h(X_1), P_{u-1}h(X_1)) \geq 0.$$

In the inequality, we have used that on functions that only depend on X_1, P_{u-1} identifies with the usual semigroup of the lazy random walk on $\{\pm 1\}^k$. Since we take $p \leq \frac{1}{2}$, it is positive semi-definite, which leads to the inequality.

For the upper bound, let τ be the spectral gap of Z_t', so that $\mathrm{Var}\left(P_u\varphi_v(Z_1')\right) \leq (1-\tau)\mathrm{Var}\left(\varphi_v(Z_1')\right)$. Thus, with the Cauchy Schwartz inequality, we get,

$$|\mathrm{Cov}(\varphi_v(Z_1')\varphi_v(Z_u'))| \leq (1-\tau)^u \mathrm{Var}(\varphi_v(Z_1')).$$

Since $\tau \geq \frac{p}{k}$, we obtain the result by summing the geometric series. □

We can now prove Proposition 1.

Proof of Proposition 1. With Lemma 1 and Lemma 3 we get

$$\sup_{x\in\mathbb{R}} \left|\mathbb{P}\left(\sigma_v^{-1}\left(\overline{Y}_T' - \mathbb{E}[\overline{Y}_T']\right) \leq x\right) - \mathbb{P}\left(G \leq x\right)\right| \leq C\frac{k^7}{p^2}\frac{\|f\|_\infty^3}{\mathrm{Var}(\varphi_v(Z_1'))^{\frac{3}{2}}\sqrt{T}}. \tag{13}$$

We begin by bounding $\mathrm{Var}(\varphi_v(Z_1'))$ from below by decomposing the functional as $\varphi_v = \sum_{j=1}^{k} \varphi_{v,j}$, where for each $i \in [k]$,

$$\varphi_{v,j}(x,y) = (f(x) - f(y))(x_j - y_j)v_j,$$

with $\varphi_{v,j}(Z_1')\varphi_{v,j'}(Z_1') = 0$ almost surely whenever $j \neq j'$. Thus, since $v_j \in \{-2, 0, 2\}$

$$\mathbb{E}\left[\varphi_v^2(Z_1')\right] = \sum_{j=1}^{k} \mathbb{E}\left[\varphi_{v,j}^2(Z_1')\right] = \frac{16p}{k} \sum_{v_j \neq 0} \mathbb{E}\left[(f(X) - f(X^{(j)}))^2\right],$$

where X is uniform on $\{\pm 1\}^k$ and $X^{(j)}$ is obtained from X by flipping the j'th coordinate. Observe that

$$\mathbb{E}\left[(f(X) - f(X^{(j)}))^2\right] = 4\mathrm{Inf}_j(f) = 4\sum_{j \in S} \hat{f}_S^2,$$

and

$$\mathbb{E}\left[\varphi_v^2(Z_1')\right] = \frac{64p}{k} \sum_{v_j \neq 0} \mathrm{Inf}_j(f).$$

On the other hand, by Lemma 2 we have

$$\mathbb{E}\left[\varphi_v(Z_1')\right]^2 = \frac{16p^2}{k^2}\left(\sum_{j=1}^{k} \hat{f}_j v_j\right)^2 \leq \frac{16p^2}{k^2} \sum_{j=1}^{k} v_j^2 \sum_{v_j \neq 0} \hat{f}_j^2 \leq \frac{64p^2}{k} \sum_{v_j \neq 0} \mathrm{Inf}_j(f),$$

where we have used the Cauchy-Schwartz inequality and the fact that $v_j \in \{-2, 0, 2\}$. Combining, and using the fact that $p \leq \frac{1}{2}$, we obtain

$$\mathrm{Var}(\varphi_v(Z_1') \geq \frac{32p}{k} \sum_{v_j \neq 0} \mathrm{Inf}_j(f) \geq \frac{32p}{k} \min_{j \in [k]} \mathrm{Inf}_j(f).$$

Plugging his into (13), we get,

$$\sup_{x \in \mathbb{R}} \left| \mathbb{P}\left(\sigma_v^{-1}\left(\overline{Y}_T' - \mathbb{E}[\overline{Y}_T']\right) \leq x\right) - \mathbb{P}\left(G \leq x\right)\right| \leq C' \frac{k^{\frac{17}{2}}}{p^{\frac{7}{2}}} \frac{\|f\|_\infty^3}{\min_{j \in [k]} \mathrm{Inf}_j(f)^{\frac{3}{2}} \sqrt{T}}. \tag{14}$$

Recall that for $B \in \mathbb{N}$, we've defined the random variable

$$B_k := \#\{0 \leq u \leq B \mid j_t \in [k]\},$$

counting the number of potential flips for the first k coordinates. Consider the event $A := \{\frac{k}{2d} \leq \frac{B_k}{B} \leq \frac{3k}{2d}\}$. As a sum of independent indicator variables, by Chernoff's inequality, we have

$$\mathbb{P}\left(A\right) \geq 1 - 2e^{-\frac{k}{12d}B}.$$

Further note that under A we can invoke (13) for $T = B_k$ and choose B to force the error bound to be of order ε. Thus, as long as $B \geq B_0(\varepsilon) = 4\frac{C^2}{\varepsilon^2}\frac{dk^{16}}{p^7}$, since the Gaussian density is bounded uniformly by $\frac{1}{\sqrt{2\pi}}$, we get

$$\begin{aligned}
\mathbb{P}\Big(\Big|\sum_{j=1}^{k} G_j(B)\left(s_j - r_j\right)\Big| \leq \varepsilon\Big) = \mathbb{P}\left(|\overline{Y}_B| \leq \varepsilon\right) &= \mathbb{P}\left(\eta\sqrt{B_k}\sigma_v^{-1}|\overline{Y}_{B_k}| \leq \sigma_v^{-1}\varepsilon\right) \\
&\leq \mathbb{P}\left(\eta\sqrt{B_k}\sigma_v^{-1}|\overline{Y}_{B_k}| \leq \sigma_v^{-1}\varepsilon \Big| A\right) + \mathbb{P}(A^c) \\
&\leq 10\sigma_v^{-1}\varepsilon \leq 10\varepsilon\sqrt{\frac{k}{p \min \mathrm{Inf}_j(f)}}
\end{aligned}$$

where in the inequality before last we have used that $\eta \geq \sqrt{\frac{2d}{Bk}}$, and so under A, $\frac{1}{\eta\sqrt{B_k}} \leq 1$. $\square$

A.2 Markov chain linear regression - results of Phase II

Given the non-degeneracy promised by Proposition 1, the main idea is that the second phase of Algorithm 1 now solves a well-posed linear regression problem. The main technical detail that requires accounting for is that the samples are not i.i.d., and so we need to appeal to results concerning Markov chains. We shall first prove that there is indeed a solution to the linear regression presented in (9), and then show that SGD performed on the random walk can find this solution.

A.2.1 Certificate

As explained, we begin by showing that Proposition 1 implies the existence of a solution to (9).

Lemma 4 (Certificate)**.** *Under the same conditions of Proposition 1, let* $b_i \overset{iid}{\sim} \mathrm{Unif}[-A, A]$*, with* $A \geq \|w^{(1)}_{[k]}\|_1 + \varepsilon$ *and* $i \in [N]$*, with* $N = \Omega\left(\frac{A}{\varepsilon}\left(k + \log(1+\varepsilon^{-1})\right)\right)$*. Then, there exists a constant* $C = C(k, \min_{i\in[k]} \mathrm{Inf}_i(f))$*, such that with probability at least* $1 - C\varepsilon$*, there exists* $a^* \in \mathbb{R}^N$*, with* $\|a^*\|_\infty = O(\|f\|_\infty/\varepsilon)$ *such that*

$$f(x) = \sum_{i=1}^{N} a^*_i \,\mathrm{ReLU}(\langle w^{(1)}_i, x\rangle + b_i), \tag{15}$$

for all $x \in \{\pm 1\}^d$*.*

Proof. Fix $\varepsilon > 0$. By applying the union bound to Proposition 1, there exists a constant $C = C(k, \min_{i\in[k]} \mathrm{Inf}_i(f))$ such that with probability at least $1 - C\varepsilon$, $w^{(1)}$ is such that, for every $s, t \in \{\pm 1\}^k$ with $\tilde{f}(s) \neq \tilde{f}(t)$:

$$\left|\sum_{j=1}^{k} w^{(1)}_j (s_j - t_j)\right| > \varepsilon. \tag{16}$$

Define the projected values

$$v(s) := \sum_{j=1}^{k} s_j w^{(1)}_j, \qquad s \in \{\pm 1\}^k. \tag{17}$$

Note that $v(s)$ takes at most $M \leq 2^k$ different values, which we order as $U = \{u_1 < \cdots < u_M\}$. By (16), all inputs with the same projected value have the same target value. We therefore define $c_j = \tilde{f}(s)$ whenever $v(s) = u_j$. Let $J = \{j \in [M-1] : c_{j+1} \neq c_j\}$. For every $j \in J$, (16) gives $u_{j+1} - u_j > \varepsilon$. Consider the interval $I = [-u_M - \varepsilon, -u_1 + \varepsilon]$. Note that $I \subseteq [-A, A]$. For every $j \in J$, consider the bias intervals

$$I^-_j = [-u_j - \varepsilon/4, -u_j - \varepsilon/8], \qquad I^+_j = [-u_{j+1} + \varepsilon/8, -u_{j+1} + \varepsilon/4].$$

Also define

$$I^-_0 = [-u_1 + 3\varepsilon/8, -u_1 + \varepsilon/2], \qquad I^+_0 = [-u_1 + \varepsilon/8, -u_1 + \varepsilon/4].$$

These $2(|J|+1) \leq 2M$ intervals are disjoint sub-intervals of I, each of width $\varepsilon/8$. If N biases are sampled uniformly from $[-A, A]$, the probability that any given interval is missed is at most

$$\left(1 - \frac{\varepsilon}{16A}\right)^N \leq \exp\left(-\frac{\varepsilon N}{16A}\right).$$

By a union bound, the probability that every interval is hit is at least

$$1 - 2M \exp\left(-\frac{\varepsilon N}{16A}\right) \geq 1 - \varepsilon,$$

provided that $N \geq \frac{16A}{\varepsilon}\log\left(2M(1+\varepsilon^{-1})\right)$, as ensured by the assumed lower bound on N. Together with the non-degeneracy event, this gives probability at least $1 - C\varepsilon$, after adjusting C. For each

$j \in J \cup \{0\}$, choose sampled biases $b_{h_j^-} \in I_j^-$ and $b_{h_j^+} \in I_j^+$, and set $r_j^- = -b_{h_j^-}$ and $r_j^+ = -b_{h_j^+}$. Consider the linear combination of these sampled neurons

$$H_j(x) = \frac{\mathrm{ReLU}(\langle w^{(1)}, x\rangle - r_j^-) - \mathrm{ReLU}(\langle w^{(1)}, x\rangle - r_j^+)}{r_j^+ - r_j^-}.$$

For $j \in J$, we have $r_j^+ - r_j^- \geq \varepsilon/2$. If $\langle w^{(1)}, x\rangle = u_\ell$, then $H_j(x) = 0$ when $\ell \leq j$, and $H_j(x) = 1$ when $\ell \geq j+1$. Moreover, $r_0^+ - r_0^- \geq \varepsilon/8$ and $r_0^+ < u_1$, so $H_0(x) = 1$ on all projected inputs. We therefore obtain the exact telescoping representation

$$f(x) = c_1 H_0(x) + \sum_{j \in J} (c_{j+1} - c_j) H_j(x), \qquad x \in \{\pm 1\}^d.$$

This construction thus constitutes the network promised in the statement of the lemma. Since the chosen intervals are disjoint, the sampled neurons are distinct. Set all unused coefficients to zero. The nonzero coefficients have absolute values

$$\frac{|c_1|}{r_0^+ - r_0^-} \quad \text{or} \quad \frac{|c_{j+1} - c_j|}{r_j^+ - r_j^-}, \qquad j \in J,$$

each bounded by $10\|f\|_\infty/\varepsilon$, as claimed.

□

A.2.2 Second Layer Training

We now analyze Phase II using the strongly convex Markov-chain SGD bound of [Eve23, Cor. 1]. We shall first explain how to set up the problem as a strongly convex and smooth linear regression problem and bound its condition number. Since $w_j^{(1)} = 0$ for all $j > k$, both f and the feature map induced by the frozen first layer depend only on the first k coordinates. It is therefore enough to work with the projected chain,

$$v^{(t)} := x_{[k]}^{(t)} \in \{\pm 1\}^k.$$

Let $\hat{b} \in \mathbb{R}^N$ be a vector of biases with coordinates i.i.d. from $\mathrm{Unif}[-A, A]$, as in Lemma 4. Using these biases, for $x \in \{\pm 1\}^k$, define the feature map

$$\Phi(x) := \Big(\mathrm{ReLU}(\langle w^{(1)}, x\rangle + \hat{b}_i)\Big)_{i=1}^N \in \mathbb{R}^N, \qquad R_\Phi := \max_{x \in \{\pm 1\}^k} \|\Phi(x)\|_2.$$

Then

$$\mathrm{NN}(x; (w^{(1)}, a, \hat{b})) = \sum_{i=1}^N a_i \,\mathrm{ReLU}\left(\langle w^{(1)}, x\rangle + \hat{b}_i\right) = a^\top \Phi(x_{[k]}).$$

Moreover, $(v^{(t)})_{t \geq 0}$ is a Markov chain on $\{\pm 1\}^k$ with transition kernel

$$P_k(s, s') = \begin{cases} p/d, & \text{if } s' = s^{(j)} \text{ for some } j \in [k], \\ 1 - kp/d, & \text{if } s' = s, \\ 0, & \text{otherwise,} \end{cases}$$

where $s^{(j)}$ denotes s with the jth coordinate flipped. This chain is reversible with stationary law $\pi = \mathrm{Unif}(\{\pm 1\}^k)$, and its characters are eigenfunctions with eigenvalues

$$\lambda_S = 1 - \frac{2p|S|}{d}, \qquad S \subseteq [k].$$

Hence the spectral gap is $2p/d$. We denote by

$$\tau_{\mathrm{mix}} := \inf\Big\{t \geq 1 : \ \forall \nu, \ d_{\mathrm{TV}}(\nu P^t, \pi) \leq 1/4\Big\},$$

the total-variation mixing time of the sampling chain, where P is the transition kernel and π its stationary distribution. In our case,

$$\tau_{\mathrm{mix}} \leq \frac{d}{2p} \log(2^{2k+1}) = O\left(\frac{dk}{p}\right). \tag{18}$$

For $\lambda > 0$, define the regularized per-state loss

$$\ell_\lambda(s;a) := \frac{1}{2}\big(a^\top \Phi(s) - \tilde{f}(s)\big)^2 + \frac{\lambda}{2}\|a\|_2^2,$$

and the total loss

$$L_\lambda(a) := 2^{-k} \sum_{s\in\{\pm1\}^k} \ell_\lambda(s;a).$$

Let $a_\lambda^\star = \arg\min L_\lambda(a)$. For each $s \in \{\pm1\}^k$,

$$\nabla_a \ell_\lambda(s;a) = \big(a^\top \Phi(s) - \tilde{f}(s)\big)\Phi(s) + \lambda a, \qquad \nabla_a^2 \ell_\lambda(s;a) = \Phi(s)\Phi(s)^\top + \lambda I.$$

Therefore $\ell_\lambda(s;\cdot)$ is λ-strongly convex and M_λ-smooth, with $M_\lambda := R_\Phi^2 + \lambda$ and condition number $\kappa_\lambda := \frac{M_\lambda}{\lambda}$. Let a^* be the vector promised by Lemma 4, so that

$$a^{*\top}\Phi(s) = \tilde{f}(s) \qquad \forall s \in \{\pm1\}^k.$$

By optimality of $a_\lambda^\star$,

$$L_\lambda(a_\lambda^\star) \le L_\lambda(a^*) = \frac{\lambda}{2}\|a^*\|_2^2. \tag{19}$$

In particular, since both terms are positive:

$$2^{-k} \sum_{s\in\{\pm1\}^k} \big(a_\lambda^{\star\top}\Phi(s) - \tilde{f}(s)\big)^2 \le \lambda\|a^*\|_2^2, \qquad \text{and,} \qquad \|a_\lambda^\star\|_2 \le \|a^*\|_2. \tag{20}$$

Define

$$\sigma_{\lambda,\star}^2 := \max_{s\in\{\pm1\}^k} \|\nabla_a \ell_\lambda(s;a_\lambda^\star)\|_2^2.$$

Then, using (20),

$$\begin{aligned}\max_{s\in\{\pm1\}^k} \big|a_\lambda^{\star\top}\Phi(s) - \tilde{f}(s)\big| &\le 2^{k/2}\left(2^{-k}\sum_{u\in\{\pm1\}^k}\big(a_\lambda^{\star\top}\Phi(u) - \tilde{f}(u)\big)^2\right)^{1/2} \\ &\le 2^{k/2}\sqrt{\lambda}\,\|a^*\|_2.\end{aligned}$$

Hence

$$\sigma_{\lambda,\star}^2 \le \Big(2^{k+1}R_\Phi^2\lambda + 2\lambda^2\Big)\|a^*\|_2^2 \le \big(2^{k+1}+2\big)M_\lambda\lambda\|a^*\|_2^2. \tag{21}$$

Finally, note that

$$R_\Phi \le \sqrt{N}\big(\|w^{(1)}\|_1 + A\big), \qquad M_\lambda \le N\big(\|w^{(1)}\|_1 + A\big)^2 + \lambda. \tag{22}$$

Proposition 2 (Phase II: convergence of the second layer)**.** *Assume the event of Proposition 1. Fix $\lambda > 0$ and run Phase II with the update*

$$a^{(t+1)} = a^{(t)} - \gamma_2 \nabla_a \ell_\lambda(v^{(t)}; a^{(t)}),$$

with step size

$$\gamma_2 = \min\left\{\frac{1}{M_\lambda},\ \frac{1}{\lambda T_2}\log\left(\frac{\lambda^2 T_2\,\|a^{(0)} - a_\lambda^\star\|_2^2}{M_\lambda\,\tau_{\mathrm{mix}}\,\sigma_{\lambda,\star}^2}\right)\right\}.$$

Then,

$$\mathbb{E}\big[\|a^{(T_2)} - a_\lambda^\star\|_2^2\big] \le 2e^{-T_2/\kappa_\lambda}\|a^{(0)} - a_\lambda^\star\|_2^2 + \widetilde{\mathcal{O}}\left(\frac{M_\lambda\tau_{\mathrm{mix}}\sigma_{\lambda,\star}^2}{\lambda^3 T_2}\right). \tag{23}$$

Consequently,

$$\begin{aligned}&\mathbb{E}_{x\sim\mathrm{Unif}(\{\pm1\}^d)}\Big[\big(\mathrm{NN}(x;(w^{(1)},a^{(T_2)},\hat{b})) - f(x)\big)^2\Big] \\ &\qquad \le \lambda\|a^*\|_2^2 + 2L_\lambda e^{-T_2/\kappa_\lambda}\|a^{(0)} - a_\lambda^\star\|_2^2 + \widetilde{\mathcal{O}}\left(\frac{L_\lambda^3\tau_{\mathrm{mix}}\|a^*\|_2^2}{\lambda^2 T_2}\right).\end{aligned} \tag{24}$$

Proof. Apply Corollary 1 of [Eve23] to the finite family of functions

$$\{\ell_\lambda(s;\cdot) : s \in \{\pm 1\}^k\},$$

using the facts established above: each $\ell_\lambda(s;\cdot)$ is λ-strongly convex and L_λ-smooth, the sampling chain is $(v^{(t)})_{t\geq 0}$, and $\sigma_{\lambda,\star}$ is bounded by (21). This gives (23).

For the population error, define

$$\mathcal{E}(a) := 2^{-k} \sum_{s\in\{\pm 1\}^k} \left(a^\top \Phi(s) - \tilde{f}(s)\right)^2.$$

Since both f and Φ depend only on the first k coordinates, $\mathcal{E}(a)$ is exactly the uniform $\{\pm 1\}^d$-risk of the network with frozen first layer. Further, since $\nabla L_\lambda(a_\lambda^\star) = 0$ and L_λ is M_λ-smooth,

$$L_\lambda(a) \leq L_\lambda(a_\lambda^\star) + \frac{M_\lambda}{2}\|a - a_\lambda^\star\|_2^2.$$

Also, by definition of L_λ,

$$\frac{1}{2}\mathcal{E}(a) \leq L_\lambda(a).$$

Combining the last two displays with (19), we obtain

$$\mathcal{E}(a) \leq \lambda\|a^*\|_2^2 + M_\lambda\|a - a_\lambda^\star\|_2^2.$$

Taking $a = a^{(T_2)}$, expectations, and using (23) together with (21) gives (24). □

We now choose λ and T_2. Set

$$\lambda := \frac{\delta}{3\|a^*\|_2^2}. \tag{25}$$

Then the first term on the right-hand side of (24) is at most $\delta/3$. It is therefore enough to choose T_2 so that

$$2M_\lambda e^{-T_2/\kappa_\lambda}\|a^{(0)} - a_\lambda^\star\|_2^2 \leq \frac{\delta}{3} \qquad \text{and} \qquad \widetilde{\mathcal{O}}\left(\frac{M_\lambda^3 \tau_{\mathrm{mix}}\|a^*\|_2^2}{\lambda^2 T_2}\right) \leq \frac{\delta}{3}. \tag{26}$$

Equivalently, one may take

$$T_2 = \widetilde{\mathcal{O}}\left(\kappa_\lambda \log\left(\frac{M_\lambda\|a^{(0)} - a_\lambda^\star\|_2^2}{\delta}\right) + \frac{M_\lambda^3\tau_{\mathrm{mix}}\|a^*\|_2^6}{\delta^3}\right).$$

Since $\tau_{\mathrm{mix}} = O(dk/p)$ by (18), and since by the certificate lemma

$$\|a^*\|_2^2 \leq 2^k\|a^*\|_\infty^2 = O_k(\varepsilon^{-2}),$$

this yields a linear bound in d and polynomial bound in $1/\delta$.

A.3 Finishing the Proof

Proof of Theorem 2. We work with $p = \frac{1}{2}$ in the lazy random walk. Thus, taking $B \geq C\frac{k^{15}d}{\varepsilon^2\tau^3}$ and $\gamma_1 = \frac{\sqrt{2Bd}}{\kappa\sqrt{k}}$, Proposition 1 applies and at the end of Phase I, the weights vector $w^{(1)}$ satisfies (10) with the given probability. Moreover, the expression 8, and the variance bound from Lemma 3, make it clear that with high probability $\|w^{(1)}\|_1 = O_k\left(\frac{\|f\|_\infty}{\varepsilon}\right)$, and so with our choice of N and A, Lemma 4 applies as well. So, after sampling the biases $\hat{b}$, with probability $1 - C\varepsilon$, there exists a^*, a weights vector for the second layer, satisfying (15). The proof now concludes by applying Proposition 2 and using the choices for T_2, γ_2, and λ. □

B Anti-concentration for additive functionals of Markov chains (Proof of Lemma 1)

We consider the *p-lazy edge random walk* of the discrete cube, defined as follows: The state space is $\Omega_k = \{(u,v) \in \{-1,1\}^k \times \{-1,1\}^k : \|u-v\|_1 \le 2\}$, i.e. the set of all oriented edges, augmented by self-loops, of the discrete k-dimensional cube. For the evolution of the chain, assume that for some $t \ge 0$, $Y_t = (u_t, v_t)$, then Y_{t+1} is obtained by performing one step of a lazy random walk starting from v_t. Call the new vertex v_{t+1} and set $Y_{t+1} = (v_t, v_{t+1})$. In other words Y_t is a random walk on the (directed) edge graph of $\{-1,1\}^k$, where the laziness is represented by the probability to move to, or stay at, a self-loop. With that perspective if $(u,v) \sim \pi_E$, the unique invariant measure, then $u \sim \mathrm{uniform}(\{-1,1\}^k)$ and v is obtained from u via a one-step p-lazy random walk.

We shall require the following definitions: Let $C(\Omega_k)$ be the space of real-valued functions over Ω_k. For $t \ge 0$, define the Markov semi-group P_t by

$$P_t f(y) = \mathbb{E}\left[f(Y_t) | Y_0 = y\right] \qquad \forall f \in C(\Omega_k).$$

Further, if $\|\cdot\|$ is any norm on $C(\Omega_k)$, we say that P_t is ρ-contracting with respect to $\|\cdot\|$, if for every $f \in C(\Omega_k)$ such that $\int f \pi_E = 0$,

$$\|P_1 f\| \le \rho \|f\|,$$

which also implies for every $t \ge 0$,

$$\|P_t f\| \le \rho^t \|f\|.$$

For $v, v' \in \{-1,1\}^k$, let

$$D_H(v,v') := \frac{1}{2}\|v - v'\|_1$$

denote the Hamming distance. Equip

$$\Omega_k = \left\{(u,v) \in \{-1,1\}^k \times \{-1,1\}^k : D_H(u,v) \le 1\right\}$$

with the metric

$$d_E\big((u,v),(u',v')\big) := D_H(v,v') + \frac{1}{2}\mathbf{1}_{\{v=v',\, u \ne u'\}},$$

which satisfies,

$$\mathrm{diam}(\Omega_k, d_E) = k.$$

For a function $f : \Omega_k \to \mathbb{R}$, define

$$\mathrm{Lip}_E(f) := \sup_{x \ne y} \frac{|f(x) - f(y)|}{d_E(x,y)}$$

and

$$\|f\|_{\infty,L} := \|f\|_\infty + \mathrm{Lip}_E(f),$$

where $\|f\|_\infty = \max_{x \in \Omega_k} |f(x)|$.

Lemma 5. *Suppose that $0 < p \le \frac{1}{2}$. Then, for every $f : \Omega_k \to \mathbb{R}$ satisfying*

$$\int_{\Omega_k} f \, d\pi_E = 0,$$

one has

$$\|P_1 f\|_{\infty,L} \le \left(1 - \frac{p}{2k^2}\right) \|f\|_{\infty,L}.$$

Proof. The main observation utilized in the proof is that $P_1 f(u,v)$ is a function only of v, as in Lemma 3. Using this observation, we first prove that

$$\mathrm{Lip}_E(P_1 f) \le \left(1 - \frac{p}{k}\right) \mathrm{Lip}_E(f). \tag{27}$$

As in [Klo21, Theorem 3.2], this contractive property essentially follows from the results in [Oll09]. For completeness, we include the full argument. Let $v' = v^{(j)}$ be obtained from v by flipping its j'th

coordinate, and let W and W' have the one-step transition laws of the p-lazy random walk started from v and v', respectively. Couple them by setting

$$\mathbb{P}\big((W,W')=(v,v)\big)=\frac{p}{k},\qquad \mathbb{P}\big((W,W')=(v',v')\big)=\frac{p}{k},\qquad \mathbb{P}\big((W,W')=(v,v')\big)=1-p-\frac{p}{k},$$

and, for every $i\neq j$,

$$\mathbb{P}\big((W,W')=(v^{(i)},(v')^{(i)})\big)=\frac{p}{k}.$$

The probabilities are nonnegative because $p\leq\frac{1}{2}$, and their two marginals are precisely the required one-step transition laws. On each of the first two events, the states (v,W) and (v',W') have the same terminal vertex and different incoming vertices, and hence their d_E-distance is $1/2$. On every other event, their terminal vertices are adjacent, and hence their d_E-distance is 1. Therefore,

$$\mathbb{E}\left[d_E\big((v,W),(v',W')\big)\right]=\frac{2p}{k}\cdot\frac{1}{2}+\left(1-\frac{2p}{k}\right)=1-\frac{p}{k}.$$

Consequently,

$$\begin{aligned}|P_1f(u,v)-P_1f(u',v')|&=\left|\mathbb{E}\big[f(v,W)-f(v',W')\big]\right|\\&\leq \mathrm{Lip}_E(f)\mathbb{E}\left[d_E\big((v,W),(v',W')\big)\right]\\&\leq\left(1-\frac{p}{k}\right)\mathrm{Lip}_E(f).\end{aligned}$$

Concatenating this estimate along a distance-minimizing path yields, for arbitrary $v\neq v'$,

$$|P_1f(u,v)-P_1f(u',v')|\leq\left(1-\frac{p}{k}\right)\mathrm{Lip}_E(f)D_H(v,v').$$

On the other hand, when $v=v'$, as we observed $P_1f(u,v)=P_1f(u',v)$ and so the above left-hand side vanishes. Hence we have shown (27). Furthermore, since P is a Markov operator, $\|P_1f\|_\infty\leq\|f\|_\infty$, and, since $\int f\,d\pi_E=0$ and $\mathrm{diam}(\Omega_k,d_E)=k$, we have $\|f\|_\infty\leq k\mathrm{Lip}_E(f)$. Thus [Klo21, Lemma 1.1] applies with $C=k$, and $\theta=1-\frac{p}{k}$. Thus, whenever $\int f\,d\pi_E=0$,

$$\|P_1f\|_{\infty,L}\leq\left(1-\frac{1-\theta}{1+C\theta}\right)\|f\|_{\infty,L}=\left(1-\frac{p}{k(k+1-p)}\right)\|f\|_{\infty,L}\leq\left(1-\frac{p}{2k^2}\right)\|f\|_{\infty,L}.$$

□

Now, fix $f\in C(\Omega_k)$ and initialize $Y_1\sim\pi_E$ according to the invariant measure. For $T\in\mathbb{N}$ we define the additive functional

$$S_T:=\frac{1}{\sqrt{T}\sigma_f}\sum_{t=1}^{T}\left(f(Y_t)-\mathbb{E}\left[f(Y_1)\right]\right),$$

where σ_f^2 is the *dynamical variance* defined as

$$\sigma_f^2:=\lim_{T\to\infty}\frac{1}{T}\mathrm{Var}\left(\sum_{t=1}^{T}f(Y_t)\right)=\mathrm{Var}(f(Y_1))+2\sum_{t=1}^{\infty}\mathrm{Cov}(f(Y_1),f(Y_{1+t})).$$

Supposing that $0<\sigma_f<\infty$, since the process Y_t is stationary we see that for every $T\in\mathbb{N}$, $\mathbb{E}[S_T]=0$ and $\lim_{T\to\infty}\mathrm{Var}(S_T)=1$. We should thus expect to have a central limit theorem for S_T. Indeed, this is the content of [Klo19, Theorem C] with a quantitative error bound.

Proposition 3. *Let $f\in C(\Omega_k)$ satisfy $\|f\|_\infty\leq M$, for some $M\geq 1$ and let G be a standard Gaussian random variable. Then,*

$$\sup_{x\in\mathbb{R}}\left|\mathbb{P}\left(S_T\leq x\right)-\mathbb{P}\left(G\leq x\right)\right|\leq C\frac{k^4M^3}{p^2\min(\sigma_f^3,\sigma_f)\sqrt{T}}.$$

Proof. First, define the function $\tilde{f}=\frac{f-\mathbb{E}[f(Y_1)]}{\sigma_f}$. According to [Klo19, Theorem C] there exists a universal constant $C>0$, such that

$$\sup_{x\in\mathbb{R}}\left|\mathbb{P}\left(S_T\leq x\right)-\mathbb{P}\left(G\leq x\right)\right|\leq C\frac{1}{\sqrt{T}(1-\rho)^2}\max\left(\|\tilde{f}\|_{\infty,L},\|\tilde{f}\|_{\infty,L}^3\right),$$

where ρ is the contraction coefficient with respect to $\|\cdot\|_{\infty,L}$. By Lemma 5, we may take $\rho = 1 - \frac{p}{2k^2}$, so that $1 - \rho = \frac{p}{2k^2}$. Further since $|\mathbb{E}[f(Y_1)]| \le \|f\|_\infty$ and the minimum nonzero d_E-distance is $1/2$, we have $\mathrm{Lip}_E(f) \le 4\|f\|_\infty$. Centering does not change the Lipschitz seminorm, so $\|\tilde{f}\|_{\infty,L} \le \frac{6\|f\|_\infty}{\sigma_f} \le \frac{6M}{\sigma_f}$. Absorbing numerical factors into C, the proof is complete. □

Since the standard Gaussian density is bounded by $\frac{1}{\sqrt{2\pi}}$, the following corollary is now immediate.

Corollary 1. *Under the same conditions of Proposition 3, let $\varepsilon > 0$. If*

$$T \ge C\varepsilon^{-2}\frac{M^6 k^8}{p^4 \min(\sigma_f^6, \sigma_f^2)},$$

for some universal constant $C > 0$, then for any $a \in \mathbb{R}$,

$$\mathbb{P}\left(|S_T - a| \le \varepsilon\right) \le \varepsilon.$$

Let us also apply Proposition 3 to the function φ_v from Lemma 1.

Proof of Lemma 1. The lemma is an immediate consequence of Proposition 3 by noting that since $v_j \in \{-2, 0, 2\}$,

$$|\varphi_v(x,y)| = \left|\left(f(x^{(u)}) - f(x^{(u-1)})\right) \cdot \sum_{j=1}^{k}(x_j^{(u)} - x_j^{(u-1)})v_j\right| \le 8\|f\|_\infty k,$$

and so we can substitute $M \le 8\|f\|_\infty k$. □

C Proof of Theorem 3

For fully connected architectures with i.i.d. initialization, and for SGD updates that treat all input coordinates symmetrically, the training dynamics are equivariant under permutations of the input coordinates. Consequently, all targets in the orbit $\mathrm{orb}(f)$ induce the same law for the trained predictor up to the corresponding input permutation, and are therefore equally hard to learn. It is thus enough to bound the average performance over $F \sim P_{\mathrm{orb}(f)}$.

Therefore, fix $F \sim P_{\mathrm{orb}(f)}$. As in the proof of Theorem 3 of [AS20a], we compare the output law under true labels and under random labels. Given a target function $F : \{\pm 1\}^d \to \{\pm 1\}$, define the update of the noisy-SGD algorithm (see Def. 3)

$$\theta_F^{(t)} = \theta_F^{(t-1)} - \frac{1}{B}\sum_{s=1}^{B} G_{t-1}(\theta_F^{(t-1)}, x_{(t-1)B+s}, F(x_{(t-1)B+s})) + Z^{(t)},$$

where $G_{t-1} = \gamma[\nabla L]_A$ is the clipped gradient vector associated with the loss, $Z^{(t)} \sim \mathcal{N}(0, \gamma^2\tau^2 I_M)$ is the gradient noise and $\{x_s\}_{s\ge 1}$ is generated by a p-lazy random walk. Let $\theta_*^{(t)}$ denote the *label-averaged* reference process, defined by the same recursion as $\theta_F^{(t)}$ but with each mini-batch gradient replaced by its conditional expectation over a uniform label $y \in \{\pm 1\}$. By construction $\theta_*^{(t)}$ is independent of F.

For each $t \in [T]$ and $H \in \{F, *\}$, let $Q_H^{(t)}$ denote the conditional law of $\theta_H^{(t)}$ given the target function F and the batches $(S_B^{(1)}, \dots, S_B^{(t)})$. By the standard total variation coupling argument,

$$\Pr\left(\mathrm{sgn}(\mathrm{NN}(x;\theta_F^{(T)})) = F(x)\right) \le \Pr\left(\mathrm{sgn}(\mathrm{NN}(x;\theta_*^{(T)})) = F(x)\right) + \mathbb{E}_{F,S_B^T}\,\mathrm{TV}\left(Q_F^{(T)}, Q_*^{(T)} \mid F, S_B^T\right).$$

Since under the junk process the final predictor depends only on junk labels and the input law is balanced,

$$\Pr_{x,F}\left(\mathrm{sgn}(\mathrm{NN}(x;\theta_*^{(T)})) = F(x)\right) \le \frac{1}{2} + o_d(1).$$

It therefore suffices to bound the total variation term. Fix $t \in [T]$. Define the one-step mean gradients

$$\Gamma_F^{(t)} := \frac{1}{B}\sum_{i=1}^{B} G_{t-1}(\theta_*^{(t-1)}, x_i^{(t)}, F(x_i^{(t)})), \qquad \Gamma_*^{(t)} := \frac{1}{B}\sum_{i=1}^{B} \mathbb{E}_{Y\sim \mathrm{Unif}(\{\pm 1\})} G_{t-1}(\theta_*^{(t-1)}, x_i^{(t)}, Y),$$

where $x_i^{(t)} = x_{(t-1)B+i}$ and for $h \in \{F, *\}$ we denote by $Q_{*,h}^{(t)}$ the distribution, given F and $S_B^{(t)}$, of $\theta_{*,h}^{(t)}$ defined as:

$$\theta_{*,h}^{(t)} = \theta_*^{(t-1)} - \Gamma_h^{(t)} + Z^{(t)}.$$

Using the triangle inequality and data-processing inequality, as in [AS20a][Proof of Theorem 3], we can write:

$$\mathrm{TV}\Big(Q_F^{(t)}, Q_*^{(t)} \mid F, S_B^t\Big) \le \mathrm{TV}\Big(Q_F^{(t-1)}, Q_*^{(t-1)} \mid F, S_B^{t-1}\Big) + \mathrm{TV}\Big(Q_{*,F}^{(t)}, Q_{*,*}^{(t)} \mid F, S_B^{(t)}\Big).$$

Conditioned on F and $S_B^{(t)}$, the one-step updates differ only by the shift in the Gaussian mean, hence

$$\mathrm{TV}\Big(Q_{*,F}^{(t)}, Q_{*,*}^{(t)} \mid F, S_B^{(t)}\Big) \le \frac{1}{2\gamma\tau}\|\Gamma_F^{(t)} - \Gamma_*^{(t)}\|_2,$$

exactly as in [AS20a, Section 5.1].

Taking expectation over $F, S_B^{(t)}$ and using Cauchy–Schwarz,

$$\mathbb{E}_{F,S_B^{(t)}} \mathrm{TV}\Big(Q_{*,F}^{(t)}, Q_{*,*}^{(t)} \mid F, S_B^{(t)}\Big) \le \frac{1}{2\gamma\tau}\left(\mathbb{E}_{F,S_B^{(t)}}\|\Gamma_F^{(t)} - \Gamma_*^{(t)}\|_2^2\right)^{1/2}.$$

We now bound the squared norm on the right-hand side. Since the samples within each batch are correlated, the tensorization argument of [AS20a] does not yield a useful bound in our setting. We therefore adopt a different approach, based on a direct kernel-norm estimate. For each coordinate $e \in [M]$, write

$$g_e(x, y) := \big(G_{t-1}(\theta_*^{(t-1)}, x, y)\big)_e.$$

Since $y \in \{\pm 1\}$ is uniform, set

$$\tilde{g}_e(x) := \frac{1}{2}\big(g_e(x, 1) - g_e(x, -1)\big).$$

Then

$$g_e(x, F(x)) - \mathbb{E}_{y\sim \mathrm{Unif}(\{\pm 1\})} g_e(x, y) = F(x)\tilde{g}_e(x),$$

so

$$\Gamma_{F,e}^{(t)} - \Gamma_{*,e}^{(t)} = \frac{1}{B}\sum_{i=1}^{B} F(x_i^{(t)})\tilde{g}_e(x_i^{(t)}).$$

Therefore

$$\mathbb{E}_{F\sim P_{\mathrm{orb}(f)}}\big(\Gamma_{F,e}^{(t)} - \Gamma_{*,e}^{(t)}\big)^2 = \mathbb{E}\left[\frac{1}{B^2}\sum_{i,j=1}^{B} K(x_i^{(t)}, x_j^{(t)})\tilde{g}_e(x_i^{(t)})\tilde{g}_e(x_j^{(t)})\right],$$

where we introduced the kernel

$$K(x, x') := \mathbb{E}_{F\sim P_{\mathrm{orb}(f)}}[F(x)F(x')].$$

Fix the batch $(x_i^{(t)})_{i=1}^B$, and write

$$a_i := \tilde{g}_e(x_i^{(t)}), \qquad K_{ij} := K(x_i^{(t)}, x_j^{(t)}).$$

Then

$$\frac{1}{B^2}\sum_{i,j=1}^{B} K_{ij}a_i a_j = \frac{1}{B^2}\langle aa^\top, K\rangle_F,$$

where $\langle \cdot, \cdot \rangle_F$ denotes the Frobenius inner product. By Frobenius Cauchy–Schwarz,

$$|\langle aa^\top, K \rangle_F| \le \|aa^\top\|_F \, \|K\|_F.$$

Since $\|aa^\top\|_F = \sum_{i=1}^B a_i^2$, we obtain

$$\frac{1}{B^2} \sum_{i,j=1}^B K_{ij} a_i a_j \le \left(\frac{1}{B} \sum_{i=1}^B a_i^2 \right) \left(\frac{1}{B^2} \sum_{i,j=1}^B K_{ij}^2 \right)^{1/2}.$$

Taking expectation and applying Cauchy–Schwarz once more yields

$$\mathbb{E}_F \big(\Gamma_{F,e}^{(t)} - \Gamma_{*,e}^{(t)} \big)^2 \le \left(\mathbb{E}\Big[\Big(\frac{1}{B} \sum_{i=1}^B \tilde{g}_e(x_i^{(t)})^2 \Big)^2 \Big] \right)^{1/2} \cdot \big(\mathrm{CP}_B^{\mathrm{rw}}(\mathcal{U}_d, P_{\mathrm{orb}(f)}) \big)^{1/2},$$

where we denoted by

$$\begin{aligned} \mathrm{CP}_B^{\mathrm{rw}}(\mathcal{U}_d, P_{\mathrm{orb}(f)}) &:= \mathbb{E}_{(x_i^{(t)})_{i=1}^B} \left[\frac{1}{B^2} \sum_{i,j=1}^B K(x_i^{(t)}, x_j^{(t)})^2 \right] \\ &= \mathbb{E}_{(x_i^{(t)})_{i=1}^B} \left[\frac{1}{B^2} \sum_{i,j=1}^B \Big(\mathbb{E}_{F \sim P_{\mathrm{orb}(f)}} [F(x_i^{(t)}) F(x_j^{(t)})] \Big)^2 \right], \end{aligned}$$

with $(x_i^{(t)})_{i=1}^B$ generated by the lazy random walk with stationary distribution $\mathcal{U}_d$ (Def. 1). Defining

$$\widetilde{G}_{t-1}(x) := \frac{1}{2} \Big(G_{t-1}(\theta_*^{(t-1)}, x, 1) - G_{t-1}(\theta_*^{(t-1)}, x, -1) \Big).$$

we have

$$\mathbb{E}_{F, S_B^{(t)}} \|\Gamma_F^{(t)} - \Gamma_*^{(t)}\|_2^2 \le \left(\mathbb{E}\Big[\Big(\frac{1}{B} \sum_{i=1}^B \|\widetilde{G}_{t-1}(X_i^{(t)})\|_2^2 \Big)^2 \Big] \right)^{1/2} \cdot \big(\mathrm{CP}_B^{\mathrm{rw}}(\mathcal{U}_d, P_{\mathrm{orb}(f)}) \big)^{1/2}.$$

Hence

$$\mathbb{E}_{F, S_B^{(t)}} \mathrm{TV}\Big(Q_{*,F}^{(t)}, Q_{*,*}^{(t)} \mid F, S_B^{(t)} \Big) \le \frac{1}{2\gamma\tau} \left(\mathbb{E}\Big[\Big(\frac{1}{B} \sum_{i=1}^B \|\widetilde{G}_{t-1}(X_i^{(t)})\|_2^2 \Big)^2 \Big] \right)^{1/4} \cdot \big(\mathrm{CP}_B^{\mathrm{rw}}(\mathcal{U}_d, P_{\mathrm{orb}(f)}) \big)^{1/4}.$$

Now use the clipping bound. Since

$$\|G_{t-1}(\theta, x, y)\|_2 \le \gamma A \sqrt{M} \qquad \text{for all } (\theta, x, y),$$

we also have

$$\|\widetilde{G}_{t-1}(x)\|_2 \le \frac{1}{2} \Big(\|G_{t-1}(\theta_*^{(t-1)}, x, 1)\|_2 + \|G_{t-1}(\theta_*^{(t-1)}, x, -1)\|_2 \Big) \le \gamma A \sqrt{M}.$$

Therefore

$$\left(\mathbb{E}\Big[\Big(\frac{1}{B} \sum_{i=1}^B \|\widetilde{G}_{t-1}(X_i^{(t)})\|_2^2 \Big)^2 \Big] \right)^{1/4} \le \gamma A \sqrt{M},$$

and thus

$$\mathbb{E}_{F, S_B^{(t)}} \mathrm{TV}\Big(Q_{*,F}^{(t)}, Q_{*,*}^{(t)} \mid F, S_B^{(t)} \Big) \le \frac{A\sqrt{M}}{2\tau} \cdot \big(\mathrm{CP}_B^{\mathrm{rw}}(\mathcal{U}_d, P_{\mathrm{orb}(f)}) \big)^{1/4}.$$

Summing over $t = 1, \dots, T$ and iterating the total variation recursion as in [AS20a, Proof of Theorem 3], we obtain

$$\mathbb{E}_{F, S_B^T} \mathrm{TV}\Big(Q_F^{(T)}, Q_*^{(T)} \mid F, S_B^T \Big) \le \frac{TA\sqrt{M}}{2\tau} \big(\mathrm{CP}_B^{\mathrm{rw}}(\mathcal{U}_d, P_{\mathrm{orb}(f)}) \big)^{1/4}.$$

Thus, to prove our result it remains to show that for large enough batch size B, $\mathrm{CP}_B^{\mathrm{rw}}$ is close to the standard CP as defined in Def. 4.

Lemma 6 (Comparison with standard cross-predictability)**.** *Let $\{x_t\}_{t\geq 1}$ be the stationary p-lazy random walk on $\{\pm 1\}^d$. Then,*

$$\mathrm{CP}_B^{\mathrm{rw}}(\mathcal{U}_d, P_{\mathrm{orb}(f)}) \leq \mathrm{CP}(\mathcal{U}_d, P_{\mathrm{orb}(f)}) + \frac{d}{pB}.$$

Proof. Fix F, F' and define $h(x) := F(x)F'(x) \in \{\pm 1\}$. Then

$$\Big(\mathbb{E}_{F\sim P_{\mathrm{orb}(f)}}[F(x)F(x')]\Big)^2 = \mathbb{E}_{F,F'\sim P_{\mathrm{orb}(f)}}[h(x)h(x')].$$

Therefore,

$$\mathrm{CP}_B^{\mathrm{rw}}(\mathcal{U}_d, P_{\mathrm{orb}(f)}) = \mathbb{E}_{F,F'}\mathbb{E}\left[\frac{1}{B^2}\sum_{i,j=1}^{B} h(x_i)h(x_j)\right] = \mathbb{E}_{F,F'}\mathbb{E}\left[\left(\frac{1}{B}\sum_{i=1}^{B} h(x_i)\right)^2\right].$$

Let

$$\bar{h}_B := \frac{1}{B}\sum_{i=1}^{B} h(x_i).$$

Since the random walk is stationary,

$$\mathbb{E}[\bar{h}_B^2] = \big(\mathbb{E}[\bar{h}_B]\big)^2 + \mathrm{Var}(\bar{h}_B) = \big(\mathbb{E}_{\mathcal{U}_d} h\big)^2 + \mathrm{Var}(\bar{h}_B).$$

Averaging over F, F' gives

$$\mathrm{CP}_B^{\mathrm{rw}}(\mathcal{U}_d, P_{\mathrm{orb}(f)}) = \mathrm{CP}(\mathcal{U}_d, P_{\mathrm{orb}(f)}) + \mathbb{E}_{F,F'}\,\mathrm{Var}(\bar{h}_B).$$

It remains to bound the variance term. The p-lazy random walk on $\{\pm 1\}^d$ is reversible with spectral gap $2p/d$. Since the walk is reversible, for every bounded observable h, decomposing $h = \mathbb{E}_{\mathcal{U}_d}[h] + h_0$ with $\mathbb{E}_{\mathcal{U}_d}[h_0] = 0$ gives

$$\mathrm{cov}(h(x_0), h(x_k)) = \langle h_0, P^k h_0\rangle_{\mathcal{U}_d} \leq (1-\mathrm{gap})^k\,\mathrm{Var}(h(x_0)),$$

where P denotes the transition kernel of the p-lazy random walk. Therefore

$$\begin{aligned}
\mathrm{Var}(\bar{h}_B) &= \frac{1}{B^2}\sum_{i,j=1}^{B}\mathrm{cov}(h(x_i), h(x_j)) \\
&\leq \frac{\mathrm{Var}(h(x_0))}{B^2}\left(B + 2\sum_{k=1}^{B-1}(B-k)(1-\mathrm{gap})^k\right) \\
&\leq \frac{\mathrm{Var}(h(x_0))}{B}\left(1 + 2\sum_{k=1}^{\infty}(1-\mathrm{gap})^k\right) \\
&\leq \frac{2}{B\cdot\mathrm{gap}}\,\mathrm{Var}(h(x_0)).
\end{aligned}$$

Since $|h| \leq 1$, we have $\mathrm{Var}(h(x_0)) \leq 1$, and thus

$$\mathrm{Var}(\bar{h}_B) \leq \frac{2}{B\cdot\mathrm{gap}} = \frac{d}{pB}.$$

This proves the result. □

D Experiment Details

All experiments were performed using the PyTorch framework ([PGM+19]) and they were executed on NVIDIA Volta V100 GPUs. Each experiment is repeated 5 times and we plot the mean; shaded regions denote ± 1 standard deviation.

Architecture. For the results presented in the main, we used a 4-layer MLP trained by SGD with all layers trained jointly (no layerwise training). In particular, we choose a fully-connected architecture of 3 hidden layers of neurons of size $512, 512, 64$, and ReLU activation. We use PyTorch's default initialization, which initializes weights of each layer with $\mathrm{Unif}[\frac{1}{\sqrt{\dim_{\mathrm{in}}}}, -\frac{1}{\sqrt{\dim_{\mathrm{in}}}}]$, where $\dim_{\mathrm{in}}$ is the input dimension of the corresponding layer.

Training procedure. We consider the square loss: $L_{\ell_2}(\hat{y}, y) := (\hat{y} - y)^2$ or the TD loss, defined in Def 2, with $\alpha = 0.9$. In the iid data case, we sample fresh batches of samples at each iterations from the uniform distribution on the hypercube. In the RW data case, we take samples from the trajectory of a lazy random walk with lazyness parameter $p = 0.9$. We stop training either when the training loss is less than 0.01, or when 10^6 iterations are performed. We select test samples from $\mathrm{Unif}\{\pm 1\}^d$, for both parities and juntas. In all experiments, the test-set is of size 8192.

Hyperparameter tuning. The primary goal of our experiments is to conduct a fair comparison between RW and iid data, with either the TD loss or the square loss. Thus, we did not engage in extensive hyperparameter tuning. We tried different learning rates, and we did not observe significant qualitative differences. We chose to report the experiments obtained for batch size of 1 and a learning rate of 0.005.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We provide formal proofs for the results in the appendix.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We made sure to explicitly state the setting in which our results hold.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

Justification: The assumptions appear in Sections 4 and 5.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide details of the experiments in Section 6 and Appendix D.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: The code used to run the experiments is submitted as a zip folder with the supplementary material.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The training and test details are in Section 6 and Appendix D.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: All experiments contain shaded one standard deviation intervals.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We specified details on the computer resources used in Appendix D.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have followed all guidelines in the attached link.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: This is a purely theoretical work.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [N/A]

    Justification: The paper does not use existing assets.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The use of LLM does not impact the core methodology used in this paper.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.