Hidden Progress in Deep Learning: SGD Learns Parities Near the Computational Limit

Anonymous Author(s)
Affiliation
Address
email

Abstract

Why is it that so many novel capabilities in deep learning are unlocked by scaling up datasets, model sizes, and training times? While there are some accounts of how these factors modulate statistical capacity, far less is known about how these factors affect the computational problem of model training. This work explores these computational issues through the lens of learning k-sparse parities of n bits, a canonical family of problems which pose computational barriers. Perhaps surprisingly, standard deep learning approaches, when applied to this problem, already exhibit a number of discontinuous impacts of increasing resources (such as dataset size, training time, or model size) on model performance, making it a natural candidate for rigorously studying these phase transitions. In particular, we empirically demonstrate that with standard initialization and training, a variety of architectures learn sparse parities in a number of iterations that nearly matches optimal lower bounds (in the SQ learning model), even without an explicit sparsity-promoting prior. Furthermore, we elucidate the mechanisms of these positive results with a theoretical analysis: we argue that the phase transition in performance is *not* due to SGD "stumbling in the dark" (with a Langevin-like mechanism) until it finds the hidden set of features; instead, we show that SGD gradually amplifies a Fourier gap (the hidden progress measure) in the population gradient.

1 Introduction

Neural networks perform better with more resources (data, model size, training time), but different tasks exhibit qualitatively different dependencies of performance on resources. In particular, while many learning tasks exhibit continuous improvement in performance with increasing resources, other cases show *discontinuous* improvement, where a capability emerges at a certain threshold . These *phase transitions* cannot be explained via statistical capacity alone. Indeed, in many cases we see a phase transition even when the amount of data remains fixed, with only model size or training time increasing. A timely example is the emergence of reasoning and few-shot learning capabilities when scaling up language models (Radford et al., 2019; Brown et al., 2020; Chowdhery et al., 2022), including computational resource only (Hoffmann et al., 2022). Power et al. (2022) give examples exhibiting discontinuous improvements in population accuracy ("grokking") when running time increases, while data and model size remain fixed.

In this work, we analyze the computational aspect of scaling in deep learning in an idealized problem setting which already exhibits discontinuous improvements. Specifically, we consider the *sparse parity problem*— where the concept is the parity of $k \ll n$ bits of an n-dimensional input. This is a canonical problem that is computationally difficult for a range of algorithms, including gradient-based methods (Kearns, 1998) and streaming algorithms (Kol et al., 2017). We focus on analyzing the resource measure of *training time*, and demonstrate that the loss curves for sparse parities display a

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

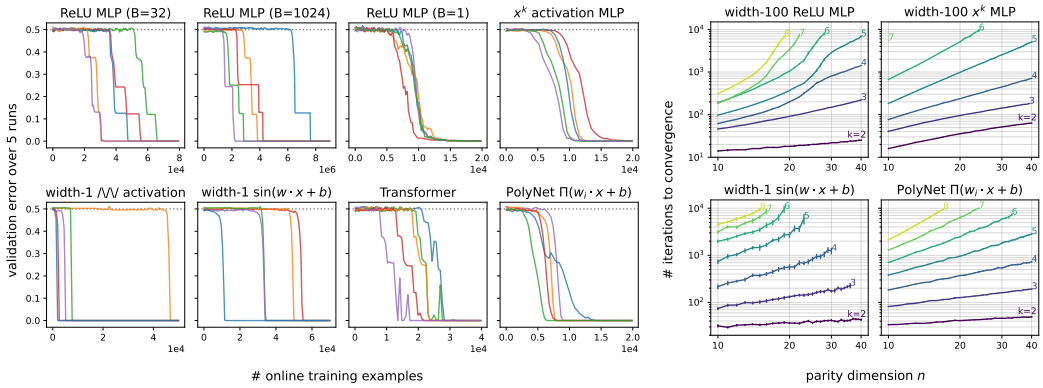

Figure 1: Main empirical findings at a glance. A variety of neural networks, with standard training and initialization, can solve the (n, k)-parity learning problem, with a number of iterations scaling as $n^{O(k)}$. *Left:* Training curves under various algorithmic choices (architecture, batch size, learning rate) on the $(n = 50, k = 3)$-parity problem. *Right:* Median convergence times for small (n, k).

phase transition across a variety of architectures and hyperparameters (see Figure 1, left). A striking observation is that SGD robustly finds the sparse subset (and hence, effectively, reaches 0 error) with a variety of activation functions, initialization schemes, and, as shown later on, even with *no* over-parameterization. This robust convergence is the starting point for our investigation.

Perhaps the most natural hypothesis to explain SGD's success in learning parities is that it simply "stumbles in the dark", essentially performing random search for the unknown target (e.g. through some stochastic Langevin dynamics). If that was the case, we may expect to observe a convergence time of $2^{O(n)}$ (which is consistent that of a naive parameter search). However, Figure 1, right, already provides some evidence against this "random search" hypothesis. This figure shows that the convergence time is closer to an $n^{O(k)}$ scaling. Notably, such a convergence rate implies SGD is closer to achieving the *optimal computation time* among a natural class of algorithms, namely SQ (statistical query) algorithms.

In this work, through an extensive empirical analysis of the scaling behavior of a variety of models, as well as theoretical analysis, we give strong evidence against the "stumbling in the dark" viewpoint. Instead, there is a *hidden progress measure* which under SGD is steadily improving. Furthermore, and perhaps surprisingly, we also give empirical and theoretical evidence that SGD is achieving a computational runtime that is much closer to the optimal SQ computational lower bound than simply doing (random) parameter search. More generally, our theoretical and empirical investigations reveal a number of notable phenomena regarding the dependence of SGD's performance on resources, and we provide further theoretical and empirical evidence of phase transitions with data, model size, and training time.

1.1 Our contributions

SGD efficiently learns sparse parities, in theory and practice. It is known that (a noisy SQ abstraction of) SGD on *any* architecture requires at least $n^{\Omega(k)}$ computational steps to learn k-sparse n-dimensional parities (for background, see Appendix A). However, with standard architectures and initialization, which do not encode the sparsity prior, one may expect SGD's performance to be much worse, on the order of $2^{\Omega(n)}$. We give extensive empirical evidence that this is not the case.

Empirical Finding 1. *For all small instances ($n \leq 30, k \leq 4$) of the sparse parity problem, architectures $\mathcal{A} \in \{$2-layer MLPs, Transformers[1], single neurons with ReLU or oscillating activations, PolyNets[2]$\}$, and initialization in $\{$uniform, Gaussian, Bernoulli$\}$, and for all batch sizes in $\{1, \ldots, 1024\}$, SGD on $\mathcal{A}$ solves the (n, k)-sparse parity problem with probability $\geq 1/2$ within at most $c \cdot n^{\alpha k}$ steps, for small architecture-dependent constants c, α.*

[1]With a smaller range of hyperparameters.

[2]A non-standard architecture introduced in this work; see Section 3 for the definition.

For a subset of these architectures, we performed more extensive training, to show scaling behaviors of the computation time.[3] See Figure 1, right.

The above results indicate that SGD succeeds in solving parities much faster than the $2^{\Omega(n)}$ steps that would be required by random search without a sparse prior. This suggests that despite progress seeming flat in the first phase of learning, SGD does in fact make progress "under the hood". We show that this is indeed the case by coming up with a *progress measure* that continuously improves throughout training. We can explain the loss and accuracy curves using this progress measure, by showing that they typically rise sharply once the measure passes a certain threshold.

Theoretical analysis. Our empirical results suggest that SGD is able to do *feature learning* for the parity problem, and identify the influential coordinates, even without a sparsity-inducing prior. We give a theoretical analysis that validates this claim. For MLPs, we prove that the *population gradient* contains enough signal on the influential coordinates to enable learning in the large-batch limit.

Informal Theorem 2. *For 2-layer MLPs, SGD in the large batch limit learns the relevant features using $n^{O(k)}$ samples.*

For an idealized architecture (which we call disjoint-PolyNet), we theoretically explain the qualitative behavior of the error curves.

Informal Theorem 3. *Gradient flow on disjoint-PolyNet exhibits two properties: (1) early progress is more visible in the weights than in the error, and (2) for a large fraction of the total running time (until perfect accuracy), accuracy remains nearly trivial.*

In addition to refuting a "random search" model, our work also poses a counterexample to other models for the computational mechanisms of deep learning; for instance, it provides a setting where deep neural nets successfully learn a concept to 100% accuracy while the corresponding Neural Tangent Kernel (NTK) only achieves trivial performance, hence showing the importance of *feature learning*. We also construct a counterexample to the *"deep only works if shallow is good"* principle of Malach and Shalev-Shwartz (2019), demonstrating a case where a deep network can get near-perfect accuracy even when greedy layerwise training (a la Belilovsky et al. (2019)) cannot beat trivial performance. By providing positive theory and empirics which elude these simplified explanations of SGD, we hope to point the way to a more complete understanding of learning dynamics in the challenging cases where no apparent progress is made for extended periods of time.

1.2 Related work

We present the most directly related work below, on feature learning and learning parities with neural nets. A broader discussion of related work can be found in Appendix A.3.

SGD and feature learning. Theoretical analysis of gradient descent over neural-networks is notoriously hard, due to the non-convex nature of the optimization problem. That said, it has been established that in some settings, the dynamics of GD keep the weights close to their initialization, thus behaving like a convex optimization over the Neural Tangent Kernel (see for example Jacot et al. (2018); Allen-Zhu et al. (2019); Du et al. (2018)). In contrast, it has been shown that in various tasks, moving away from the fixed features of the NTK is essential for the success neural networks trained with GD (for example Yehudai and Shamir (2019); Allen-Zhu and Li (2019); Wei et al. (2019) and the review in Malach et al. (2021)). These results demonstrate that feature learning is an important part of the GD optimization process. Our work also focuses on a setting where feature learning is essential for the target task. In our theoretical analysis, we show that the initial population gradient encodes the relevant features for the problem. The importance of the first gradient step for feature learning has been recently studied in Ba et al. (2022).

Learning parities with neural networks. The problem of learning parities using neural networks has been investigated in prior works from various perspectives. It has been demonstrated that parities are hard for gradient-based algorithms, using similar arguments as in the SQ analysis (Shalev-Shwartz et al., 2017; Abbe and Sandon, 2020). One possible approach for overcoming the computational hardness is to make favorable assumptions on the input distribution. Indeed, recent works show that under various assumptions on the input distribution, neural networks can be efficiently trained to

[3] While our focus is on the performance as a function of *training time*, we also performed some experiments on performance as a function of *model size*, see Section 5.

learn parities (XORs) (Daniely and Malach, 2020; Shi et al., 2021; Frei et al., 2022; Malach et al., 2021). In contrast to these results, this work takes the approach of intentionally focusing on a hard benchmark task, without assuming that the distribution has some favorable (namely, non-uniform) structure. This setting allows us to probe the performance of deep learning at a known computational limit. Notably, the work of Andoni et al. (2014) provides analysis for learning polynomials (and in particular, parities) under the uniform distribution. However, their main results require a network of size $n^{O(k)}$ (i.e., extremely overparameterized network), and provides only partial theoretical and empirical evidence for the success of smaller networks. Studying a related subject, some works have shown that neural networks display a spectral bias, learning to fit low-frequency coefficients before high-frequency ones (Rahaman et al., 2019; Cao et al., 2019).

2 Preliminaries

We define necessary notation here; see Appendix A for more background and technical ingredients.

Sparse parities. For integer $n \geq 1$ and non-empty $S \subseteq [n]$, the (n, S)-*parity function* $\chi_S : \{\pm 1\}^n \to \{\pm 1\}$ is defined as $\chi_S(x) = \prod_{i \in S} x_i$. We define the (n, S)-*parity distribution* $\mathcal{D}_S$ as the distribution over $(x, y = \chi_S(x))$[4] where x is drawn from the uniform distribution over $\{\pm 1\}^n$, which we denote by $\mathrm{Unif}(\{\pm 1\}^n)$. We define the (n, k)-*parity learning problem* as the task of recovering S using samples from $\mathcal{D}_S$, where S is chosen at random in $\binom{[n]}{k}$. Statistically, it is possible to do so using $\Theta(\log \binom{n}{k}) \approx k \log n$ samples. However, in the *statistical query* (SQ) model (Kearns, 1998), (which has been shown to encapsulate gradient-based methods such as GD or SGD (Abbe et al., 2021)), this task requires $\Omega(n^k)$ queries (assuming constant level of noise). While learning noiseless parities can be solved by Gaussian elimination using $O(n)$ samples, learning sparse *noisy* parities, even at a very small noise level (i.e., $o(1)$ or $n^{-\delta}$) is believed to inherently require $n^{\Omega(k)}$ computational steps and samples, or exponential computation with $n^{o(k)}$ samples. This was first explicitly conjectured by Alekhnovich (2003), and has been the basis for several cryptographic schemes (e.g., (Ishai et al., 2008; Applebaum et al., 2009, 2010; Bogdanov et al., 2019)).

Notation for neural networks and training. Our main results are presented in the online learning setting, with a stream of i.i.d. batches of examples. At each iteration $t = 1, \ldots, T$, a learning algorithm receives a batch of B examples $\{(x_{t,i}, y_{t,i})\}_{i=1}^B$ drawn i.i.d. from $\mathcal{D}_S$, then outputs a classifier $\widehat{y}_t : \{\pm 1\}^n \to \{\pm 1\}$. We say that the algorithm solves the parity task in t steps with ϵ error, if with probability at least $1 - \epsilon$ over both training and internal randomness, as well as $(x, y) \sim \mathcal{D}_S$, $\widehat{y}_t(x) = y$. We will focus on the case that $\widehat{y}_t = \mathrm{sign}(f(x; \theta_t))$ for some parameters θ_t in a continuous domain Θ and for a continuous function $f : \{\pm 1\}^n \times \Theta \to \mathbb{R}$.[5] A ubiquitous online learning algorithm is gradient descent (GD). For a choice of loss function $\ell : \{\pm 1\} \times \mathbb{R} \to \mathbb{R}$, initialization θ_0 (that are chosen from some distribution), learning rate schedule $\{\eta_t\}_{t=1}^T \subseteq \mathbb{R}$ and weight-decay schedule $\{\lambda_t\}_{t=1}^T \subseteq \mathbb{R}$, GD refers the standard iterative update rule using the regularized, empirical loss function, which is a function of *architecture* f. The learning rate η_t can also be a vector (e.g., allowing different rate schedules for different layers).

3 Empirical findings

3.1 SGD on neural networks learns sparse parities

The central phenomenon of study in this work is the empirical observation that neural networks, with standard initialization and training, can solve the (n, k)-parity problem in a number of iterations scaling as $n^{O(k)}$ on small instances. We observed robust positive results for randomly-initialized SGD on the following architectures, indexed by Roman numerals:

- *2-layer MLPs:* ReLU ($\sigma(z) = (z)_+$) or polynomial ($\sigma(z) = z^k$) activation, in a wide variety of width regimes $r \geq k$. Settings (i), (ii), (iii) (resp. (iv), (v), (vi)) use $r = \{10, 100, 1000\}$ ReLU

[4] Our theoretical analyses and experiments can tolerate noisy parities, that is, random flipping of the label. For ease of presentation, we state the non-noisy setting.

[5] When $f(x; \theta) = 0$ in practice (e.g. with sign initialization), we break the tie arbitrarily. We ensure in the theoretical analysis that this does not happen.

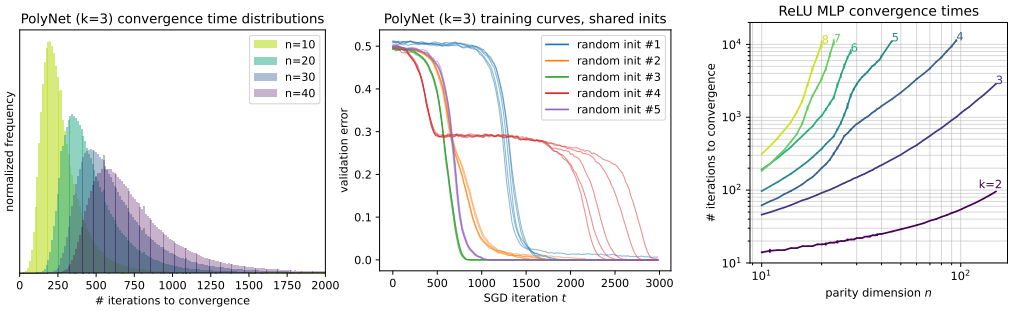

Figure 2: Black-box observations about the training dynamics. *Left:* Histograms of convergence times over 10^6 random trials, with heavy upper tails but no observed successes near $t = 0$ (unlike random search). *Center:* Optimization path (and convergence time) depend heavily on initialization, not the randomness of SGD; $B = 128, \eta = 0.01$ shown here. *Right:* The power-law exponent (α such that $t_c \propto n^\alpha$) eventually degrades on larger problem instances.

(resp. polynomial) activations. We also consider $r = k$ (exceptional settings (*i), (*ii)), the minimum width for representing a k-wise parity for both activations.

- *Single neurons:* Next, we consider non-standard activation functions σ which allow a one-neuron architecture $f(x; w) = \sigma(w^\top x)$ to realize k-wise parities. The constructions stem from letting $w^* = \sum_{i \in S} e_i$, and constructing $\sigma(\cdot)$ to interpolate (the appropriate scaling of) $\frac{k - w^{*\top} x}{2} \bmod 2$ with a piecewise linear k-*zigzag* activation (vii), or a degree-k polynomial (viii). Going a step further, a single ∞-*zigzag* (ix) or *sinusoidal* (x) neuron can represent *all* k-wise parities. We also removed the second trainable layer (setting $u = 1$), obtaining settings (xi), (xii), (xiii), (xiv). We found that wider architectures with these activations also trained successfully.

- *Transformers:* Motivated by recent theoretical and empirical work on the ability of self-attention to learn sparse functions and parities (Lu et al., 2021; Edelman et al., 2021; Hahn, 2020), we consider a simplified specialization of the Transformer architecture to this sequence classification problem. This is the less-robust setting (*iii); the architecture and optimizer are described in Appendix D.1.3.

- *PolyNets:* Our final setting (xv) introduces a non-standard architecture, which may be of independent interest. The PolyNet takes a product over k linear functions $f(x; w_{1:k}) := \prod_{i=1}^k w_i^\top x$; this can clearly realize k-way parities. Originally motivated by the search for an idealized setting where an end-to-end optimization trajectory analysis is tractable (see Section 4.1), we found in these experiments that this architecture trains very stably and sample-efficiently.

Robust space of positive results. All of the networks listed above were observed to successfully learn sparse parities in a variety of settings; it would be uninformative to enumerate all of them in a table or show training curves. Instead, we summarize our findings as follows: for all combinations of $n \in \{10, 20, 30\}$, $k \in \{2, 3, 4\}$, batch sizes $B \in \{1, 2, 4, \ldots, 1024\}$, initializations {uniform, Gaussian, Bernoulli}, loss functions {hinge, square, cross entropy}, and architecture configurations $\{(\text{i}), (\text{ii}), \ldots, (\text{xv})\}$, SGD solved the parity problem (with 100% accuracy, validated on a batch of 2^{13} samples) in at least 5 out of 10 random trials, for at least one choice of learning rate $\eta \in \{0.01, 0.1, 1\}$. The models converged in $t_c \leq c \cdot n^{\alpha k} \leq 10^5$ steps, for small architecture-dependent constants c, α, shown in Appendix C.1. Figure 1 *(left)* shows some representative training curves.

Less robust configurations. The settings (*i) and (*ii), where the MLP just barely represents a k-sparse parity, and the Transformer setting (*iii), were less robust to hyperparameter choices (especially the batch size B). We present these results in a few settings, described in Appendix D.1

3.2 Random search or hidden progress?

The remainder of this paper seeks to answer the question: *"By what mechanism does deep learning solve these emblematic computationally-hard optimization problems?"*

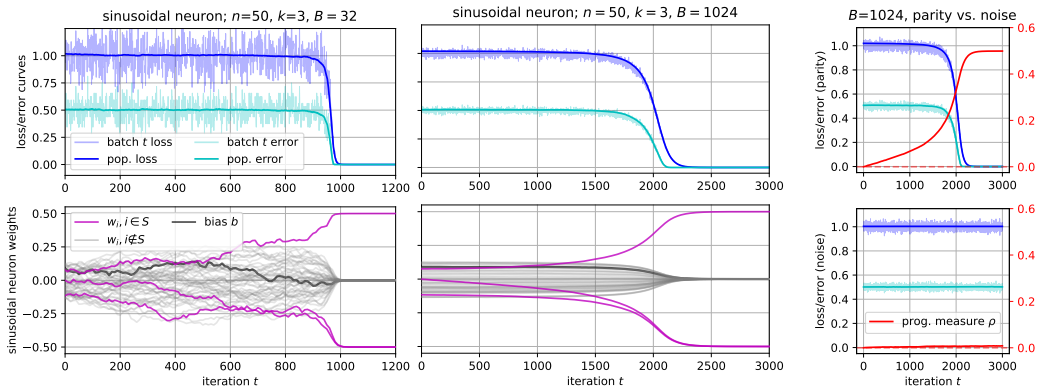

Figure 3: Hidden progress when learning parities with neural networks. *Left, center:* Black-box losses and accuracies exhibit a long plateau and sharp phase transition (top), hiding gradual progress in the SGD iterates (bottom). *Right:* A hidden progress measure which distinguishes gradual feature amplification (top) from training on noise (bottom).

A natural hypothesis would be that SGD somehow implicitly performs Monte Carlo random search, "bouncing around" the loss landscape in the absence of a useful gradient signal. Several empirical observations clash with this hypothesis:

- *Scaling of convergence times:* Without an explicit sparsity prior in the architecture or initialization, it is unclear how to account for the $n^{\Theta(k)}$ runtimes observed in experiments. The initializations, which certainly do not prefer sparse functions[6], are close to the correct solutions with probability $2^{-\Omega(n)} \ll n^{-k}$. On larger instances (n, k), the power-law exponents worsen; see Figure 2 *(right)*, and the discussion in Appendix C.4.
- *No early convergence:* Over a large number of random trials, no copies of this randomized algorithm get "lucky" (i.e. solve the problem in significantly fewer than the median number of iterations); see Figure 2 *(left)*. The success times of random exhaustive search would be distributed as $\mathrm{Geom}(1/\binom{n}{k})$; it is *more* likely to succeed (and terminate) at smaller t.
- *Sensitivity to initialization, not stochastic batches:* Running these training setups over multiple stochastic batches from a common initialization, we find that loss curves and convergence times are highly correlated with the architecture's random initialization; see Figure 2 *(center)*.

Even these observations, which do not probe the internal state of the algorithm, suggest exhaustive search is an insufficient picture of the training dynamics and a different mechanism is at play.

4 Theoretical analyses

4.1 Provable emergence of the parity indices in high-precision gradients

We now provide a theoretical account for the success of SGD in solving the (n, k)-parity problem. Our main theoretical observation is that, in many cases, the *population* gradient of the weights at initialization contains enough "information" for solving the parity problem. That is, given an accurate enough estimate of the initial gradient (by e.g. computing the gradient over a large enough batch size), the relevant subset S can be found.

As a warm-up example, consider training a single ReLU neuron $f(x; w) = \sigma(w^\top x)$ w.r.t. the correlation loss $\ell(y, \widehat{y}) = -y\widehat{y}$, from the initialization $w = [1, \ldots, 1]$. While a single neuron cannot *express* the parity, we observe that the population gradient can indicate what the correct subset is:

[6]Indeed, under all standard architectures and initialization, the probability that a random network is $\Omega(1)$-correlated with a sparse parity would be $2^{-\Omega(n)}$, since with that probability $1 - o(1)$ of the total influence would be accounted by the $n - k$ irrelevant features.

$\mathbb{E}_{(x,y)\sim\mathcal{D}_S}\left[\nabla_{w_i}\ell(y, f(x;w))\right] = \mathbb{E}_{(x,y)\sim\mathcal{D}_S}\left[-yx_i\,\sigma'(w^\top x)\right]$ which corresponds to either the order-$(k-1)$ Fourier coefficient $S \setminus \{i\}$ of the function $x \mapsto \sigma'(w^\top x)$ (if $i \in S$ is a relevant coordinate) or the order-$(k+1)$ coefficient $S \cup \{i\}$ (if $i \notin S$ is irrelevant). When σ is the ReLU function and $w = [1, \ldots, 1]$, $\sigma'(w^\top x) = \frac{\mathrm{sign}\left(\sum_i x_i\right)+1}{2}$ is just a shifted Majority function of x. For Majority, the Fourier spectrum is well understood: for even k, there is a gap between these Fourier coefficients that is detectable using $n^{O(k)}$ samples.

This analysis can be further extended to ReLU neuron initialized with weights $w \in \{\pm 1\}^n$ and a small bias. In fact, we can show that taking a single gradient step with large enough batch size on a ReLU network trained with the square-loss, already finds features that can solve the parity problem. Using this, we show that training a ReLU MLP with SGD solves the parity problem:

Theorem 4. *Fix some $\epsilon \in (0,1)$, let k be some even number, and assume that n is some odd number satisfying $n \geq \Omega(k^4 \log(k/\epsilon))$. There exists a symmetric random initialization scheme, learning rate and weight-decay schedules, s.t. for every $S \subseteq [n]$ of size k, training a ReLU MLP of size $r = O(2^k k \log(k/\epsilon))$ with batch-size $B = n^{O(k)} \log(1/\epsilon)$ for $T = \mathrm{poly}(k, r, 1/\epsilon)$ iterations on $\mathcal{D}_S$ w.r.t. the square-loss, finds a network $f(x;\theta_t)$ with expected loss: $\mathbb{E}\left[\ell(f(x;\theta_t), y)\right] \leq \epsilon$, where the expectation is over the randomness of initialization, training and sampling $(x,y) \sim \mathcal{D}_S$.*

The analysis presented above shows that when the *batch size* scales with $n^{O(k)}$, SGD over MLP solves the parity problem. However, in our experimental setting, the *number of steps*, and not the batch size, scales with $n^{O(k)}$. While we believe that running SGD with small batch size and small learning rate essentially amplifies the signal in the population gradient, behaving similarly to performing a large step over a large batch size, we do not have a complete analysis for training MLPs with SGD in the small batch size regime. To complement the above result, in Section 4.2 we analyze the trajectory for a variant of the PolyNet architecture trained with gradient flow.

We note that, while the above analysis applies for ReLU MLPs with a specific initialization scheme, a similar feature emergence phenomenon can be observed in a broader set of architectures and setting. Indeed, for feature emergence to occur, we only require that there is a "gap" between the relevant and irrelevant Fourier coefficients. Formally, denote $\widehat{f}(S) := \mathbb{E}\left[f(x)\chi_S(x)\right]$ the Fourier coefficient of f at S, and observe the following definition:

Definition 1 (Fourier gap)**.** *For some function $f : \{\pm 1\}^n \to \mathbb{R}$ and some subset S of size k, we say that f has a γ-*Fourier gap *at S if (1) for every $(k-1)$-element subset $S' \subseteq S$, it holds that $|\widehat{f}(S')| \geq \gamma$, and (2) for every subset $S' \subseteq [n]$ of size $k+1$ it holds that $|\widehat{f}(S')| \leq \gamma/2$.*

Now, given a network architecture where some neuron has a γ-Fourier gap with respect to the target subset S, we can generalize the result of the ReLU neuron. That is, we show that the subset S can be determined by observing an estimate of the population gradient at initialization:

Proposition 5. *Let σ be some activation function and let ℓ be some loss function. Denote $f(x;w) = \sigma(w^\top x)$. Fix some subset $S \subseteq [n]$. Let $g \in \mathbb{R}^n$ be an estimate of the population gradient such that $\left\|g - \mathbb{E}_{(x,y)\sim\mathcal{D}_S}\left[\nabla_w \ell(y, f(x;w))\right]\right\|_\infty \leq \gamma/4$. Then, for every w s.t. $\sigma'(w^\top x)$ has a γ-Fourier gap w.r.t. to S and $\ell'(f(x;w), y) = -y$ for all x, the target subset S is detected by g, namely $S = \{i \in [n] \,:\, |g_i| \geq 3\gamma/4\}$.*

Comparison with NTK analysis. In recent years, many theoretical works have studied the behavior of SGD on neural networks through the lens of the neural tangent kernel (NTK) (Jacot et al., 2018). It is therefore important to highlight the fact that the NTK (or, in fact, any kernel) cannot solve the sparse parity problem. The following result shows that no kernel can achieve small error on the sparse parity problem, unless the size of the kernel is $n^{\Omega(k)}$:

Theorem 6. *Let $\Psi : \{\pm 1\}^n \to \mathbb{R}^D$ be some D-dimensional embedding. If $D < 2\epsilon \cdot \binom{n}{k}$, then there exists some (n,k)-parity distribution $\mathcal{D}_S$ s.t. $\inf_{w \in \mathbb{R}^D} \mathbb{E}_{(x,y)\sim\mathcal{D}_S}\left[\ell(\Psi(x)^\top w, y)\right] > 1/2 - \epsilon$.*

The above is a corollary of the results in (Kamath et al., 2020; Malach and Shalev-Shwartz, 2022).

4.2 PolyNet: an idealized architecture for trajectory analyses

In this section, we present an idealized architecture (a version of PolyNets (xv)) that exhibits similar behavior as MLPs (experimentally) and is technically easier to analyze. More specifically, we

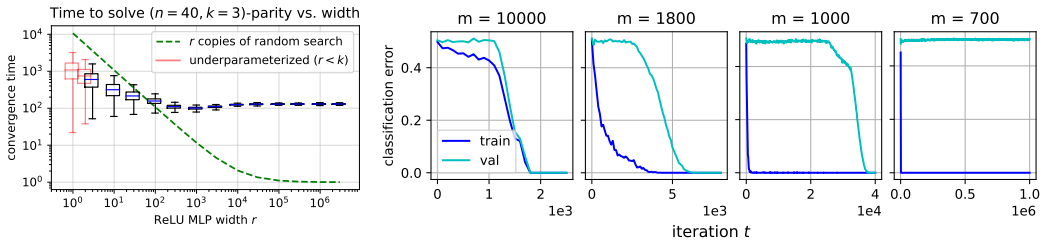

Figure 4: Parity as a sandbox for understanding the effects of model size and dataset size. *Left:* Success times vs. network width r on a fixed $(40, 3)$-parity task: in accordance with the theory, parallelization experiences diminishing returns (unlike expected success times for random search, shown in green). Underparameterized models ($r = 1, 2$ were considered successful upon reaching 55% accuracy). *Right:* Training curves for an identical setup ($(50, 3)$-parity task, architecture, and training algorithm), varying only the sample size m. The two center panels display *"grokking"*: a large gap between the time to zero train error vs. zero test error.

consider the *disjoint PolyNet* which takes a product over k linear functions where the linear functions are computed on k disjoint partitions of the input $P_1, \ldots, P_k$ with each $P_i = \{(i-1)n'+1, \ldots, im\}$ with $n' = n/k$[7], that is, $f(x; w_{1:k}) := \prod_{i=1}^{k} w_i^\top x_{P_i}$.

In order for the disjoint-PolyNet to be able to express the class of parity problems, we assume that the set S of size k in the (n, k)-parity problem is selected such that exactly one index belongs to each disjoint partition, that is, for all $i \in [k]$, $S \cap P_i = 1$. We refer to this problem as the (n, k)-disjoint parity problem. Note that there are still $m^k = (n/k)^k$ different possibilities for set S under this restriction. For fixed k, these represent a constant portion of the $\binom{n}{k} \approx (ne/k)^k$ (by Stirling's approximation) possibilities for S in the general non-disjoint case.

Consider training a disjoint-PolyNet w.r.t. the correlation loss. WLOG, let $S = \{1, n'+1, \ldots, (k-1)n'+1\}$ and $e_1 = (1, 0, ..0)$. The population gradient is non-zero at i iff $i \in S$: $g_i(w_{1:k}) = \mathbb{E}\left[\nabla_{w_i} \ell(f(x; w_{1:k}), y)\right] = -\mathbb{E}\left[y \left(\prod_{j \neq i} w_j^\top x_{P_j}\right) x_{P_i}\right] = -\left(\prod_{j \neq i} w_{j,1}\right) e_1$.

Now we consider the gradient flow dynamics of disjoint-PolyNet, which provide a mathematically tractable case study for the trajectory of sparse parity learning. For each $i \in [k]$, in this section we treat w_i as a function from $\mathbb{R}_{\geq 0} \to \mathbb{R}^m$ which satisfies the following differential equation $\dot{w}_i = -g_i(w_{1:k}(t))$. For clarity of exposition, assume all-ones initialization.[8] Then all of the relevant weights $\{w_{i,1} : i \in [k]\}$ follow the same trajectory, which we denote by $v : \mathbb{R}_{\geq 0} \to \mathbb{R}$.

Theorem 7. *The error at time t is $1/2 - (v(t)/\sqrt{m})^{ck}$ for some constant $c > 0$.*

Thus, the trajectory of error appears to display a sharper "phase transition" than that of $v(t)$.

Theorem 8. *For $k \geq 3$ and $\epsilon > 0$, the fraction of the time it takes for the error to fall below $\frac{1}{2} - \epsilon$ is $1 - \tilde{O}\left(\frac{1}{\epsilon^{1-2/k} m^{k/2-1}}\right)$ of the running time required to achieve zero error.*

In other words, for sufficiently large m, almost no progress is visible in the accuracy for most of the running time.

5 Hidden progress: discussion and additional experiments

In this section, we advocate for sparse parities as an idealized testbed for understanding algorithms and phenomena in modern deep learning. These are accompanied by experimental vignettes which are auxiliary to the core results from Section 3, with more systematic studies deferred to future work. Details are given in Appendix D.

[7] We assume for simplicity that n is divisible by k.

[8] Results for Gaussian initialization are qualitatively similar and can be found in the Appendix. By analyzing the dependence of running time on initialization, we can also explain the "no early convergence" empirical observation that it is extremely rare for the parity bits to be learned in significantly fewer than the median number of iterations.

A progress measure for parity. To begin, the black-box experiments in Section 3 suggest that random search is the incorrect model of SGD's behavior in this setting. Using the theoretical insight that amplifying a precise initial population gradient suffices for learning parities, we construct one possible progress measure, which is a function of the sequence of weights $w_0, \dots w_t \in \mathbb{R}^n$ so far: $\rho(w_{0:t}) := \|w_t - w_0\|_\infty$, using the fact that $w_t - w_0$ is an estimate for the initial population gradient in the linearized setting. Figure 3 shows how gradual weight movement (and thus, progress) can be hidden behind plateauing losses.

Roles of overparameterization vs. *oversampling*. An interesting consequence of our analysis is that it illuminates scaling behaviors with respect to a third fundamental resource parameter: *model size*, which we study in terms of network width r. If SGD operated by a "random search" mechanism, one would expect width to provide a parallel speedup. Instead, we SGD sequentially amplifies progress. The sharp lower tails in Figure 2 *(left)* imply that running r identical copies of SGD does *not* give $\times 1/r$ speedups; more directly, in Figure 4 *(left)*, convergence times for sparse parities empirically plateau at large model sizes.

It is a significant challenge to understand the interactions between network *depth* and computation, and largely outside the scope of this work. However, in Appendix C.3, we provide a brief note on using parities and polynomial-activation MLPs to construct a simple counterexample to the *"deep only works if shallow is good"* principle of Malach and Shalev-Shwartz (2019), demonstrating a case where a deep network can get near-perfect accuracy even when greedy layerwise training (e.g. (Belilovsky et al., 2019)) cannot beat trivial performance.

Learning and grokking in the finite-sample multi-pass setting. The main theoretical and empirical results in this work consider online learning algorithms which couple the resources of training time and independent samples. However, due to the computational-statistical gap in parity learning, these positive results are suboptimal in terms of sample efficiency. We find that minibatch SGD (with weight decay) can empirically solve sparse parities, even from a sample of size $m \ll n^k$. For small values of m, we reliably observe the *grokking* phenomenon (Power et al., 2022): overfitting for a long time, then a *delayed* phase transition for the generalization error; see the two center panels of Figure 4 *(right)*.

6 Conclusion

This work puts forward parity learning as a stylized test case to explore some of the puzzling features of the role of computational (as opposed to statistical) resources in deep learning. These include discontinuous improvements (a.k.a. *emergent capabilities*), feature learning, and universality of architectures. In particular, we show that deep learning on parities exhibits a phase transition behavior, that it is successfully learned by a variety of deep-net architectures, and that this success cannot be explained as a "random exhaustive search", nor through frameworks such as the neural tangent kernel or layer-by-layer learning.

However, there are more experimental and theoretical questions, even for this simplified case of parity learning. Our focus in this work was on the online learning case, where training time and samples arise in tandem. However, we believe it would be instructive to investigate parity learning when three resources of samples, time, and model size are scaled separately. Some very preliminary findings along these lines are presented in Section 3.

Extending our theoretical results to the small-batch setting, as well as to more architectures, is an open problem. Resolving it would require a better understanding of anti-concentration (lower bound on deviation from mean) of Fourier coefficients, a phenomenon that is much less studied than the concentration of these coefficients. We would also want to extend the analysis beyond parities to tasks that are not aligned with the elementary basis such as low-rank tensor recovery.

Another important follow-up direction is understanding the extent that these insights extend from parity learning to real-world problems, as well as the extent into which non-synthetic tasks (in, e.g., natural language processing and program synthesis) embed within them parity-like subtasks of exhaustive combinatorial search.

Broader impact. This work seeks to contribute to the foundational understanding of computational scaling behaviors in deep learning. Our theoretical and empirical analyses are in a heavily-idealized synthetic problem setting. Hence, we see no direct societal impacts of these results in this study.

References

Emmanuel Abbe and Colin Sandon. Poly-time universality and limitations of deep learning. *arXiv preprint arXiv:2001.02992*, 2020.

Emmanuel Abbe, Pritish Kamath, Eran Malach, Colin Sandon, and Nathan Srebro. On the power of differentiable learning versus pac and sq learning. *Advances in Neural Information Processing Systems*, 34, 2021.

Michael Alekhnovich. More on average case vs approximation complexity. In *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings.*, pages 298–307. IEEE, 2003.

Zeyuan Allen-Zhu and Yuanzhi Li. What can resnet learn efficiently, going beyond kernels? *Advances in Neural Information Processing Systems*, 32, 2019.

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.

Alexandr Andoni, Rina Panigrahy, Gregory Valiant, and Li Zhang. Learning polynomials with neural networks. In *International conference on machine learning*, pages 1908–1916. PMLR, 2014.

Benny Applebaum, David Cash, Chris Peikert, and Amit Sahai. Fast cryptographic primitives and circular-secure encryption based on hard learning problems. In *Annual International Cryptology Conference*, pages 595–618. Springer, 2009.

Benny Applebaum, Boaz Barak, and Avi Wigderson. Public-key cryptography from different assumptions. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 171–180, 2010.

Jimmy Ba, Murat A Erdogdu, Taiji Suzuki, Zhichao Wang, Denny Wu, and Greg Yang. High-dimensional asymptotics of feature learning: How one gradient step improves the representation. *arXiv preprint arXiv:2205.01445*, 2022.

Eugene Belilovsky, Michael Eickenberg, and Edouard Oyallon. Greedy layerwise learning can scale to imagenet. In *International conference on machine learning*, pages 583–593. PMLR, 2019.

Andrej Bogdanov, Manuel Sabin, and Prashant Nalini Vasudevan. Xor codes and sparse learning parity with noise. *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 986–1004, 2019.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

Yuan Cao, Zhiying Fang, Yue Wu, Ding-Xuan Zhou, and Quanquan Gu. Towards understanding the spectral bias of deep learning. *arXiv preprint arXiv:1912.01198*, 2019.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

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

Simon S Du, Xiyu Zhai, Barnabas Poczos, and Aarti Singh. Gradient descent provably optimizes over-parameterized neural networks. *arXiv preprint arXiv:1810.02054*, 2018.

Benjamin L Edelman, Surbhi Goel, Sham Kakade, and Cyril Zhang. Inductive biases and variable creation in self-attention mechanisms. *arXiv preprint arXiv:2110.10090*, 2021.

Spencer Frei, Niladri S Chatterji, and Peter L Bartlett. Random feature amplification: Feature learning and generalization in neural networks. *arXiv preprint arXiv:2202.07626*, 2022.

Michael Hahn. Theoretical limitations of self-attention in neural sequence models. *Transactions of the Association for Computational Linguistics*, 8:156–171, 2020.

Godfrey Harold Hardy, John Edensor Littlewood, George Pólya, György Pólya, et al. *Inequalities*. Cambridge university press, 1952.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Cryptography with constant computational overhead. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 433–442, 2008.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31, 2018.

Pritish Kamath, Omar Montasser, and Nathan Srebro. Approximate is good enough: Probabilistic variants of dimensional and margin complexity. In *Conference on Learning Theory*, pages 2236–2262. PMLR, 2020.

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

Gillat Kol, Ran Raz, and Avishay Tal. Time-space hardness of learning sparse parities. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1067–1080, 2017.

Kevin Lu, Aditya Grover, Pieter Abbeel, and Igor Mordatch. Pretrained transformers as universal computation engines. *arXiv preprint arXiv:2103.05247*, 2021.

Eran Malach and Shai Shalev-Shwartz. Is deeper better only when shallow is good? *Advances in Neural Information Processing Systems*, 32, 2019.

Eran Malach and Shai Shalev-Shwartz. When hardness of approximation meets hardness of learning. *Journal of Machine Learning Research*, 23(91):1–24, 2022.

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.

Ryan O'Donnell. *Analysis of boolean functions*. Cambridge University Press, 2014.

Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets. *arXiv preprint arXiv:2201.02177*, 2022.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *International Conference on Machine Learning*, pages 5301–5310. PMLR, 2019.

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.

Zhenmei Shi, Junyi Wei, and Yingyu Liang. A theoretical analysis on feature learning in neural networks: Emergence from inputs and advantage over fixed features. In *International Conference on Learning Representations*, 2021.

Colin Wei, Jason D Lee, Qiang Liu, and Tengyu Ma. Regularization matters: Generalization and optimization of neural nets vs their induced kernel. *Advances in Neural Information Processing Systems*, 32, 2019.

Gilad Yehudai and Ohad Shamir. On the power and limitations of random features for understanding neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...

(a) Did you state the full set of assumptions of all theoretical results? [Yes]
(b) Did you include complete proofs of all theoretical results? [Yes]

3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [N/A]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]