EGALITARIAN GRADIENT DESCENT: A SIMPLE APPROACH TO ACCELERATED GROKKING

Ali Saheb Pasand[1,2] **Elvis Dohmatob**[2,3]

[1]McGill University [2]Mila–Quebec AI Institute [3]Concordia University

October 7, 2025

ABSTRACT

Grokking is the phenomenon whereby, unlike the training performance, which peaks early in the training process, the test/generalization performance of a model stagnates over arbitrarily many epochs and then suddenly jumps to usually close to perfect levels. In practice, it is desirable to reduce the length of such plateaus, that is to make the learning process "grok" faster. In this work, we provide new insights into grokking. First, we show both empirically and theoretically that grokking can be induced by asymmetric speeds of (stochastic) gradient descent, along different principal (i.e singular directions) of the gradients. We then propose a simple modification that normalizes the gradients so that dynamics along all the principal directions evolves at exactly the same speed. Then, we establish that this modified method, which we call egalitarian gradient descent (EGD) and can be seen as a carefully modified form of natural gradient descent, groks much faster. In fact, in some cases the stagnation is completely removed. Finally, we empirically show that on classical arithmetic problems such as modular addition and sparse parity problem which this stagnation has been widely observed and intensively studied, that our proposed method eliminates the plateaus.

1 Introduction

Neural networks sometimes exhibit a striking training dynamic known as *grokking*: after rapidly driving the training error to (near) zero, test performance can linger near chance for a long period before rising abruptly to near-perfect generalization—often without any change to the optimizer or explicit early stopping (Power et al., 2022). Grokking has been observed across architectures and tasks, from algorithmic problems such as modular arithmetic (Gromov, 2023; Doshi et al., 2024) and formal languages to more naturalistic settings (Liu et al., 2023; Nanda et al., 2023). Yet despite a rapidly growing body of empirical and mechanistic case studies, we still lack a principled account of *why* the delay occurs, *what* structural features crystallize at the transition, and *how* to predict or control it (see Section 2 for a detailed review of the existing literature).

In this work, we examine grokking through the lens of the dynamics of eigen-spectra of gradients during optimization, and propose a simple modification of (stochastic) gradient descent which provably reduces the length of the plateau without compromising the level of generalization of the model at the end of training.

Contributions. Our main contributions can be summarized as follows.

- *Egalitarian Gradient Descent (EGD).* We propose a novel and simple method to accelerate grokking and study its properties both theoretically and empirically. Our method operates by modifying the gradients at each layer so that the principal directions (also referred to as singular directions) are conserved, but the speed of evolution of the dynamics along each of these directions is the same. This stabilizes the training by reducing the effect of ill-conditioned loss landscapes (where gradients vary significantly in magnitude across different principal directions), leading to accelerated grokking.
- *An Effective Theory.* We develop a theory which shows that our proposed EGD method is guaranteed to rapidly grok, compared to vanilla methods such as (stochastic) gradient descent. We also make formal links to natural

gradient descent. In fact, we show that our proposed method (which can be seen as a carefully modified version of natural gradient descent) is a simplified version of Grokfast (which operates by low-pass filtering of the gradients to boost weaker components). At a high-level, we also show that both methods have the same inductive bias of making the scales of the dynamics of the evolution of the parameters comparable along different important directions.

– *Empirical Verification.* We run extensive experiments on toy problems (binary classification of anisotropic multivariate data) and arithmetic problems (sparse parity, modular addition, etc.) and show that our proposed method typically groks immediately at the very beginning of the learning curve.

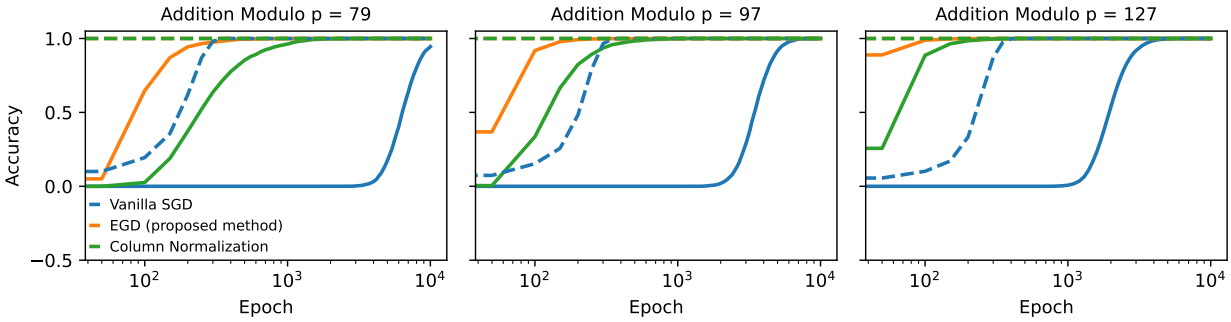

Figure 1: **Results on Modular Addition** for different values of the modulus p. Solid lines correspond to test accuracy and broken lines correspond to train accuracy. In all cases, our proposed EGD (*egalitarian gradient descent*) method groks after only a few epochs, while vanilla (stochastic) gradient descent stagnates for a long period before eventually grokking. We also include "Column Normalization", a simplification of EGD which simply rescales the columns of gradient matrices by dividing by their L_2 norm. Even this simplification seems to grok much faster than the baseline, vanilla (S)GD. Refer to Section 5 for details and to Appendix B for the hyper-parameters used.

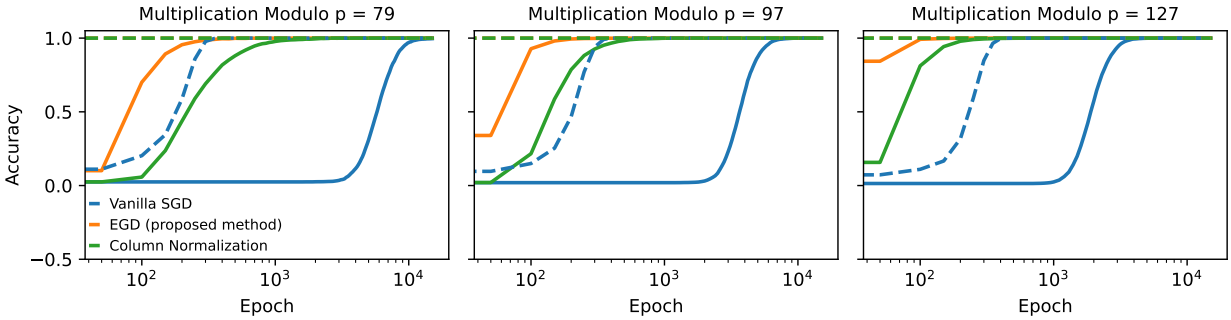

Figure 2: **Results on Modular Multiplication** for different values of the modulus p. Solid lines correspond to test accuracy and broken lines correspond to train accuracy. In all cases, our proposed EGD method groks after only a few epochs, while all the other methods stagnate a long period before eventually grokking. Refer to Section 5 for details and to Appendix B for the hyperparameters used.

2 Related Work

Grokking—the phenomenon where models first overfit the training set and only much later undergo a sharp jump in test accuracy—was first documented by Power et al. (2022). Subsequent empirical studies broadened the scope beyond purely algorithmic datasets and analyzed conditions under which grokking appears or disappears, e.g., via loss–norm tradeoffs, optimizer choice and dataset/regularization choices (Liu et al., 2023; Nanda et al., 2023; Notsawo et al., 2025; Thilak et al., 2022; Liu et al., 2022; Davies et al., 2023; Notsawo et al., 2023; Varma et al., 2023).

Plateau Phenomena and Singular Learning Theory. Long before grokking (Power et al., 2022), prolonged *training error* plateaus were analyzed in information geometry and statistical–mechanics treatments of neural nets. Plateaus arise near *singular* parameter regions—caused by permutation symmetries and redundancies—where the Fisher information degenerates and gradient flow becomes extremely slow (Wei et al., 2008; Amari et al., 2018). Classical

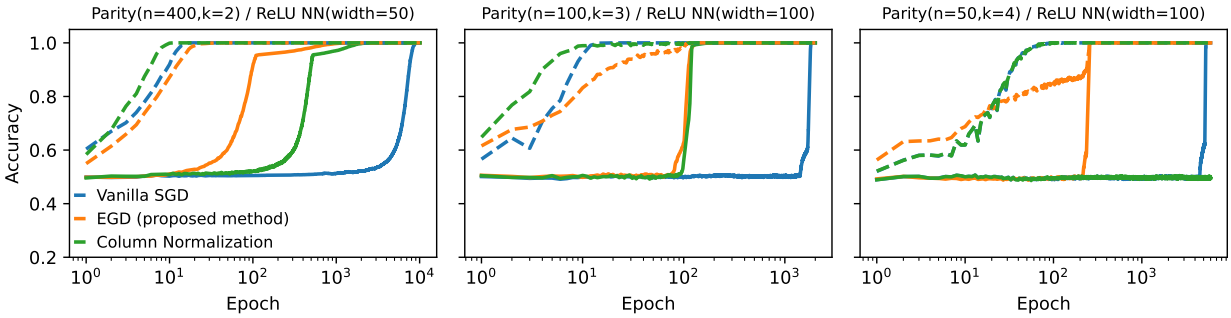

Figure 3: **Results on Sparse Parity Problem.** Solid lines correspond to test accuracy and broken lines correspond to train accuracy. All three plots show that our method (EGD) groks significantly faster than other methods. Refer to Section 5 for details on the experimental setup and to Appendix B for the hyperparameters used.

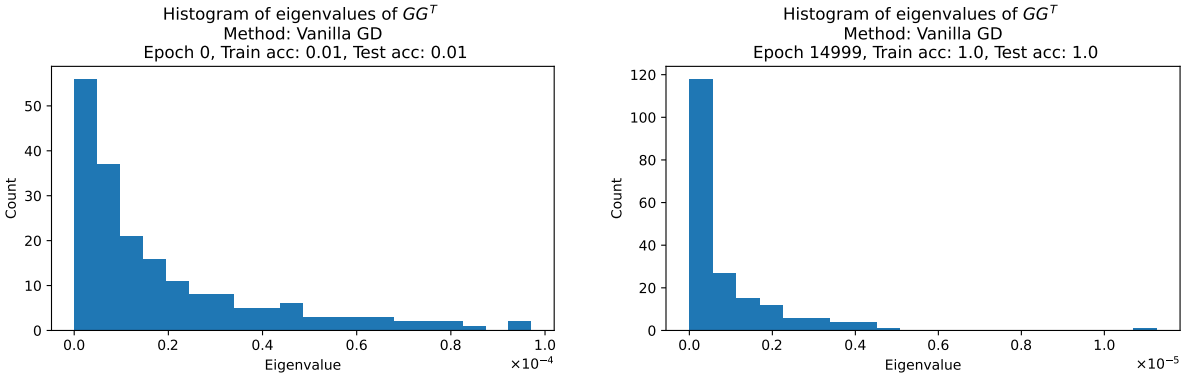

Figure 4: **Ill-conditioned Gradient Spectra** causes delayed generalization. We consider the problem of learning addition modulo 97 from data, with a two-layer ReLU neural network. At the start of optimization through to the end, the gradient matrix G for the hidden layer has a poor condition number. Here, we see that the largest singular-value (corresponding to a fast direction) is much larger than the smallest (corresponding to slow directions). This causes the overall dynamics of vanilla (S)GD to stall for arbitrarily long times, leading to delayed generalization (see Figure 1). Our proposed method, EGT (*egalitarian gradient descent*) forces all the singular values of G to be equal.

online-learning studies of multilayer (soft-committee) networks likewise reported long transients and quantified their dependence on teacher–student alignment (Saad and Solla, 1995). Singular learning theory gives a general account via algebraic geometry, relating generalization behavior and marginal likelihood to model singularities (Watanabe, 2009). More recently, statistical–mechanical analyses have shown how input-data spectra modulate whether plateaus appear prominently at all (Yoshida and Okada, 2019). While these plateaus need not coincide with delayed generalization as in grokking, the underlying mechanisms—degenerate Fisher spectra, symmetry breaking, and slow modes—are highly relevant to grokking.

Grokking in Modular Arithmetic. A rich line of mechanistic interpretability work reverse-engineers the circuits by which small transformers and MLPs implement modular addition. Notably, Zhong et al. (2023) identify complementary algorithmic mechanisms ("clock" and "pizza") and circular embeddings that emerge at the grokking transition. Complementary analyses and constructive solutions for modular addition are provided in Gromov (2023), while Doshi et al. (2024) extend these insights to modular polynomials (e.g., multi-term addition and related arithmetic), showing that trained networks converge toward these circuits near the onset of generalization.

Grokking as Kernel Escape. Another perspective sees grokking as a dynamical transition from a lazy, kernel-like regime to a rich, feature-learning regime. Kumar et al. (2024); Walker et al. (2025) formalize conditions under which such a transition yields delayed generalization without requiring explicit regularization. For modular addition, Mohamadi et al. (2024) give a theoretical analysis explaining why early kernel-regime learning cannot generalize under symmetry constraints, while later training escapes the kernel and finds small-norm, generalizing solutions. Aligned with this view, Varma et al. (2023) study grokking through the lens of circuit efficiency. This study shows that networks

quickly learn memorizing solutions which generalize poorly, and the circuits that generalize well will take longer to form. When both circuits are formed, the generalizing circuits will be dominant in generating outputs.

Training Stability and the Optimization Lens. An orthogonal view emphasizes numerical stability and optimization dynamics as bottlenecks that can stall generalization; Prieto et al. (2025) argue that operating near the edge of numerical stability can induce grokking-like delays and propose remedies that restore or accelerate test performance. Similar to this perspective, Thilak et al. (2022) identify a mechanism called "Slingshot" in which cyclic phase transitions in adaptive optimizers co-occur with grokking. The early spectral characteristics of learning curves have also been proposed as predictors of grokking, reducing the need for long training (Notsawo et al., 2023). In a more general study, Liu et al. (2022) develops an effective theory of representation learning, showing that generalization emerges in a specific zone for the weights of a network called the "Goldilocks Zone". Grokking occurs when the weights enter this region, which is a narrow phase between memorization and confusion. In broader perspectives, grokking is linked to phenomena such as double descent and emergent abilities (Huang et al., 2024; Davies et al., 2023), suggesting that delayed generalization is caused by the competition between memorization and generalization circuits. These studies show that both data and representation dependent dynamics play roles in the emergence of grokking phenomena. As a result, grokking can be affected by architectural, optimization dynamics, and data-related interventions.

Grokking beyond arithmetic and Delay Mitigation. Grokking is not limited to algorithmic tasks. It has been observed in some computer vision and natural understanding tasks (Liu et al., 2023; Lee et al., 2024). Also, structural grokking has been shown to occur in language models, where models discover hierarchical sentence structures after extensive training (Murty et al., 2023; Zhu et al., 2024). As more practical tasks exhibit grokking behavior, an important and interesting research question is how we can reduce the delay between memorization and generalization. Practical interventions have been shown to be effective to shorten or remove the grokking delay (Lee et al., 2024; Lyle et al., 2025). *Grokfast* amplifies slow (low-frequency) gradient components via simple optimizer-side filters, consistently accelerating grokking across tasks and architectures (Lee et al., 2024). In Section 4.2, we discuss the algorithmic and conceptual benefits of our proposed method over Grokfast.

Also, accelerating grokking has shown to be beneficial in a practical scenario in which data distributions shift during training. To address this, Lyle et al. (2025) propose effective learning rate scaling and re-warming as a method to trigger and accelerate feature-learning dynamics during training which can both accelerate grokking and address the issue of primacy bias in continual learning. These dynamics-aware methods complement representation-side interventions and regularization/norm-control levers observed to modulate the phenomenon (Liu et al., 2023; Nanda et al., 2023).

3 warm-up: Motivation from a Simplified Setup

We start with a simple analytically solvable example where the structure of the data (relative strength of features, relative difficulty of training and test datasets) and optimization choices (learning rate, size of initialization) interact to provably produce grokking curves with arbitrarily long plateaus. The setting we consider here is directly motivated by the observations in Figure 4, where the delayed generalization in vanilla (S)GD is caused by ill-conditioned gradient matrices.

Data Distribution. Fix some small $\varepsilon > 0$ and let $z = (z^{(1)}, z^{(2)})$ be centered Gaussian random vector with covariance matrix $\Sigma = \begin{pmatrix} 1 & 0 \\ 0 & \varepsilon \end{pmatrix}$. Let $x_{train} \in \mathbb{R}^2$ be a folded version of z obtained by conditioning on the event $|z^\top e_1| = |z^{(1)}| \geq s$, where $e_1 = (1, 0)$ is the horizontal basis vector in $\mathbb{R}^2$. Let $x_{test} \in \mathbb{R}^2$ be a folded version of z obtained by conditioning on the event $(z^\top v)z^{(1)} \leq 0$. Here, $v \in \mathbb{R}^2$ is a fixed vector which makes an angle $\theta \in (0, \pi/2)$ with e_1. In this case, the condition number of the feature covariance matrix is $1/\varepsilon \gg 1$. This captures, albeit in a simplified way, what is happening in Figure 4.

The training data is $(x_1, y_1), \ldots, (x_n, y_n)$ where $y_i := sign(x_i^{(1)})$, and the feature vectors x_i's are i.i.d. copies of x_{train}. Here, $x_i^{(j)} := x_i^\top e_j$ is the 1 component of the ith datapoint x_i. For the test data, the feature vectors are i.i.d. copies of x_{test} instead. The larger the value of s, the easier it is to quickly memorize the training data (perfect training accuracy). The situation is illustrated in Figure 5

Model. We consider a linear model $x \mapsto sign(x^\top \hat{w})$, where the weights vector $\hat{w}$ is obtained by minimizing the following quadratic loss function:

$$\ell(w) := \frac{1}{n} \sum_{i=1}^{n} (x_i^\top w - y_i)^2 = \frac{1}{n} \|Xw - Y\|^2, \tag{1}$$

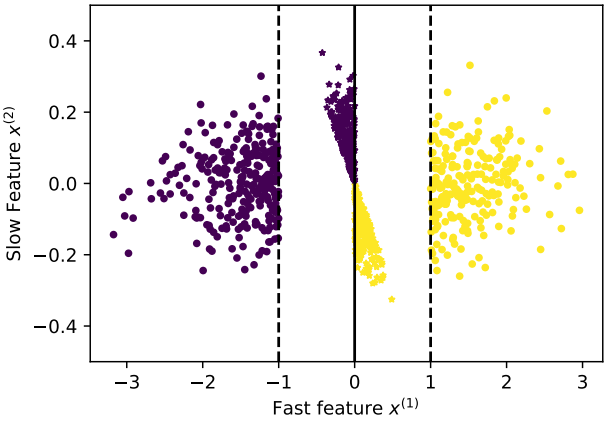

Figure 5: **A Toy Setup which Induces Stagnation in Gradient Descent (GD).** Training data points correspond to circles and test data points correspond to stars (middle region). The broken lines correspond to the large margin of the training data (their separation is $2s$), while the solid line is the ground-truth decision-boundary $x^{(1)} = 0$. The variance of the slow feature $x^{(2)}$ scales like $\varepsilon \ll 1$. GD would quickly find a linear model which perfectly separates the training data but will take a time of order $1/\varepsilon$ to find the ground-truth model which attains perfect test accuracy. See Figure 6.

where $X \in \mathbb{R}^{n\times d}$ is the design matrix with rows $x_1, \ldots, x_n$, and $Y = (y_1, \ldots, y_n) \in \{\pm 1\}^n$ is the response vector. We choose this loss function because it leads to tractable analysis while retaining the same phenomenology we would get using the logistic loss function, for example.

3.1 Vanilla Gradient-Descent Dynamics

With step size η, the vanilla gradient descent (GD) on loss (1) gives the following recursion:

$$w(k) = w(k-1) - \eta X^\top (Xw(k-1) - Y)/n = Aw(k-1) + b, \tag{2}$$

$$\text{with } b := \eta X^\top Y/n, \quad A := I - \eta\hat{\Sigma}, \quad \hat{\Sigma} := X^\top X/n. \tag{3}$$

Here $\eta > 0$ is a sufficiently small step-size / learning rate, k is the iteration (aka *epoch*), and $\hat{\Sigma}$ is the empirical covariance matrix. We can explicitly solve the above recursion to get (see Appendix A):

$$w(k) = A^k w(0) + (I - A^k)\hat{w}_{ols}, \tag{4}$$

where $\hat{w}_{ols} := X^+ Y = \hat{\Sigma}^{-1} X^\top Y/n$ is the ordinary least-squares solution. Thus, $w(k)$ interpolates between the initialization $w(0)$ and the least-squares solution $\hat{w}_{ols}$.

Define the constants $m_1 \in (0,1)$, $m_2 \in (1,\infty)$, $\alpha \in \mathbb{R}$, and $\beta \in (0,1)$ by

$$m_1 := \mathbb{E}[|x_i^\top e_1|] = \varphi(s)/Q(s),\ m_2 := \mathbb{E}[|x_i^\top e_1|^2] = 1 + sm_1,\ \alpha := 1 - \eta m_2,\ \beta := 1 - \eta\varepsilon, \tag{5}$$

where φ is the probability density function (PDF) of the standard Gaussian distribution, and $Q := 1 - \Phi$ is its survival function. Let the initialization be $w(0) = u = (u_1, u_2)$, and define the following dynamical quantities:

$$\mu_k := \alpha^k u_1 + (1-\alpha^k)\frac{m_1}{m_2},\ \nu_k := \beta^k u_2,\ L_k := \sqrt{\mu_k^2 + \varepsilon\nu_k^2}. \tag{6}$$

The evolution of the test error is given analytically by the following result.

Theorem 1. *For large n, it holds w.h.p that: for any iteration $k \geq 1$,*

$$E(w(k)) \simeq \min(1, \arccos(r_k)/\arccos(r)),\ \textit{with } r := \rho/\gamma, \quad r_k := \mu_k/L_k, \tag{7}$$

$$\textit{where } \rho := \cos\theta, \quad \gamma := \sqrt{\rho^2 + \varepsilon\cdot(1-\rho^2)}, \quad L_k := \sqrt{\mu_k^2 + \varepsilon\nu_k^2}. \tag{8}$$

Theorem 1 reveals that: the test error plateaus at value 100%, until k is sufficiently large that r_k decreases below r, at which point the error starts to decrease at the rate $\arccos(r_k)/\arccos(r)$.

We have the following important corollary.

Corollary 1. *(A) **Large Initialization.** If $|u_2|$ is large in the sense that $\tau := |u_2||\tan\theta| m_2 > m_1$, then the plateau length of the test error (the time to grok) is of order $k_* \asymp \frac{\log\tau}{\eta\varepsilon} \asymp \frac{1}{\varepsilon}$.*

*(B) **Small Initialization.** If $|u_2|$ is small, then the plateau length is of order $k_* \asymp \frac{1}{\eta}\log\frac{1}{\tau\varepsilon} \asymp \log\frac{1}{\varepsilon}$.*

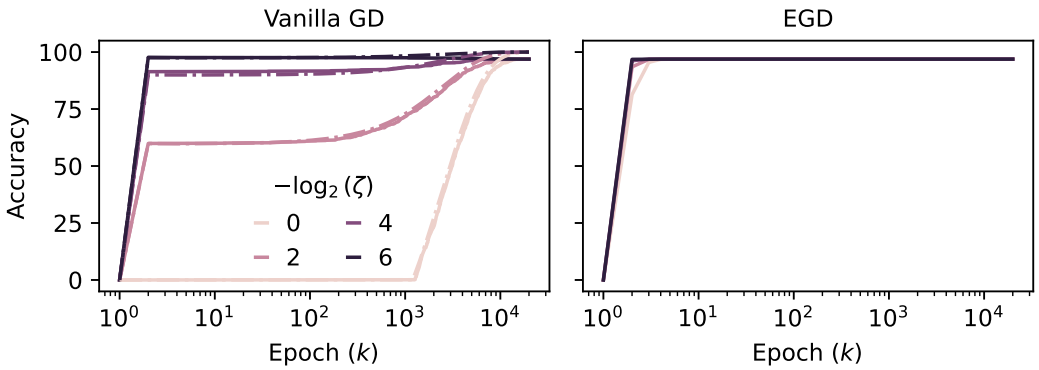

Figure 6: **Grokking on the Toy Problem.** Solid lines correspond to experimental results, while broken lines correspond to our theory (Theorem 1). The initialization is $w(0)$ is such that $\|w(0)\| = \zeta$. Thus, the scalar $\zeta > 0$ controls the size of the initialization. **Left.** As predicted by Theorem 1 and Corollary 1, large initialization leads to delayed generalization in vanilla GD, i.e. long plateaus (of length $k_* \asymp 1/\varepsilon$) of stagnation in the test performance, while small initialization attenuates it ($k_* \asymp \log 1/(\tau\varepsilon)$, where $\tau \propto \zeta$). Note that for this problem, the ratio $1/\varepsilon \gg 1$ represents the condition number of the covariance matrix of the features. **Right.** Our proposed EGD (*egalitarian gradient descent*) method groks immediately: the generalization error jumps to a perfect value after only a few iterations. Moreover, EGD appears to be completely insensitive to the scaling of the initialization, which is a desirable property in real-world optimization.

Corollary 1 shows that a model trained via vanilla gradient decent will quickly converge to a decision boundary which memorizes the training data (100% training accuracy, but poor test accuracy), but it will take a long time of order $1/\varepsilon$ or $\log 1/\varepsilon$ (depending on the size of initialization) before it converges to the ground truth decision boundary (Theorem 1), and only then does the test error abruptly increase to 100%. This result can be extended to vanilla stochastic gradient descent (SGD) as well.

The Issue with Vanilla Gradient Descent. One can show that for large sample size ($n \to \infty$), the empirical covariance matrix $\hat{\Sigma}$ which modulates the dynamics (2) has the following approximation.

$$\hat{\Sigma} \simeq \begin{pmatrix} m_2 & 0 \\ 0 & \varepsilon \end{pmatrix},$$

Where m_2 is as defined in (5). Since m_2 is a fixed positive constant, the condition number of the RHS is of the order $1/\varepsilon$. This controls the rate at which the GD iterates $w(k)$ converge to the least-squares solution via (4), which is in fact optimal (perfect test accuracy) here since we are in the infinite-sample regime). Therefore,

Insight #1. *Grokking for this toy problem is therefore sole due to a delayed convergence least-squares solution caused by ill-conditioned gradients.*

3.2 Accelerated Grokking via a Modified Gradient Scheme

Consider the following modified dynamics, which will later form the basis for our proposed method:

$$w(k) = w(k-1) - \eta\hat{\Sigma}^{-1}X^\top(Xw(k-1) - Y)/n, \tag{9}$$

with $\hat{\Sigma} := X^\top X/n$ as before. Expanding the above equation, the dynamics become

$$\begin{aligned} w(k) &= w(k-1) - \eta\hat{\Sigma}^{-1}((X^\top X/n)w(k-1)) - X^\top Y/n) \\ &= w(k-1) - (\eta w(k-1) - \eta\hat{\Sigma}^{-1}X^\top Y/n) = aw(k-1) + \eta\hat{w}_{ols}, \end{aligned}$$

with $a := 1 - \eta$. What has happened is that the inverse *Fisher information matrix (FIM)* has killed the ill-conditioned $\hat{\Sigma}$ matrix, which was multiplicatively damping the iterations. Solving the above recurrence gives

$$w(k) = a^k w(0) + (\sum_{j=0}^{k-1} a^j)\eta\hat{w}_{ols} = a^k w(0) + \frac{1-a^k}{1-a}\eta\hat{w}_{ols} = a^k w(0) + (1-a^k)\hat{w}_{ols}. \tag{10}$$

The troublesome dependence on ε has disappeared completely from the picture. Moreover, we see that the convergence to $\hat{w}_{ols}$ is now significantly faster than with the vanilla dynamics (4), namely $w(k) = A^k w(0) + (I - A^k)\hat{w}_{ols}$, where

$A := I - \eta\hat{\Sigma}$. Thus, in (10) an isotropic matrix aI (spectral radius = $a = 1 - \eta$, a fixed positive constant less than 1 for any step-size $\eta \in (0, 1)$) replaces

$$A = I - \eta\hat{\Sigma} \simeq \begin{pmatrix} 1 - \eta m_2 & 0 \\ 0 & 1 - \eta\varepsilon \end{pmatrix}$$

which has spectral radius = $1 - \eta\epsilon \approx 1$, in (4). Therefore, under the modified gradient descent update rule (9), the iterations $w(k)$ converge to $\hat{w}_{ols}$ at an exponential rate independent of ε. This leads to grokking after only a few iterations, as seen in Figure 6.

Insight #2. *The modified GD update rule* (9) *induces the desirable normalizing effect whereby the optimization dynamics has exactly the same speed along all principal directions.*

4 Proposed Method: Egalitarian Gradient Descent

Motivated by the insights from Section 3.2, we now consider a general non-linear model (neural network, etc.) on a general problem and let G be the gradient matrix for an arbitrary layer. Thus, G has the shape $m \times p$, where m is the fan-out and p is the fan-in width for that layer. For example, if the layer is the hidden layer in a two-layer feedforward full-connected neural network, then m is the number of hidden neurons and p is the input dimension. Consider the following transformation of the gradient matrix G:

$$\textbf{(EGD)} \qquad \tilde{G} := F^{-1/2}G = (GG^\top)^{-1/2}G, \tag{11}$$

where $(GG^\top)^{-1/2}$ denotes the matrix square root of the inverse of the $GG^\top$. We call this transformation *egalitarian gradient descent (EGD)*, a name that will become clear shortly.

Observe that the above transformation leaves the left and right singular vectors of G unchanged but makes all the singular values equal. Indeed, consider the singular value decomposition (SVD) of G:

$$G = USV^\top = \sum_j s_j u_j v_j^\top, \tag{12}$$

where $U = (u_j)_j$ and $V = (v_j)_j$ contain the left and right singular vectors (aka principal directions) of G, and $S = \mathrm{diag}(s_1, s_2, \ldots)$ contains its singular values. Then, $(GG^\top)^{-1/2} = US^{-1}U^\top$ and so

$$\tilde{G} = (GG^\top)^{-1/2}G = US^{-1}U^\top USV^\top = US^{-1}SV^\top = UV^\top,$$

which has the same left and right singular vectors as G but singular values all equal to 1. This means that *all through the optimization process, no principal direction will evolve faster/slower than another.* This justifies the name EGD (egalitarian gradient descent) given to (11), and we shall show that it drastically accelerates grokking.

Remark 1. *Note that in formula* (11)*, we have implicitly assumed that G is full-rank. In the case of rank deficiency (which will happen if $m > p$ for example), we simply replace $(GG^\top)^{-1}$ and S^{-1} by their Moore-Penrose pseudo-inverses.*

Practical Considerations. The gradient $\tilde{G} = UV^\top$ for our proposed EGD method is obtained by performing SVD on the original gradient matrix G. This is the main computational cost incurred by our method. In practice, we turn off EGD and switch it for vanilla (S)GD once we detect grokking has occured[1]. Moreover, experiments suggest that the SVD does not have to be precise, and randomized approximations thereof work just fine.

4.1 Connection to Natural Gradient Descent

The empirical *Fisher information matrix (FIM)* for any layer with gradient matrix G is given by $F = GG^\top$, the same matrix that appears in (11). It is then easy to see that

$$\frac{1}{m}\|\tilde{G}\|_F^2 = \frac{1}{m}\,\mathrm{tr}[\tilde{G}^\top F\tilde{G}] = \frac{1}{m}\,\mathrm{tr}[(GG^\top)^{-1/2}G^\top GG^\top G(GG^\top)^{-1/2}] = \frac{1}{m}\,\mathrm{tr}\,I = 1.$$

Thus, the Fisher-norm of the modified gradient matrix G is a constant of motion of the dynamics induced by our proposed transformation (11); however, note that our EGD proposed method is not equivalent to *natural gradient descent (NGD)*

[1]This is detected by monitoring validation loss.

(Amari, 1998; Pascanu and Bengio, 2013; Ollivier et al., 2017), which would correspond to $\bar{G} := F^{-1}G = (GG^\top)^{-1}G$. Despite their differences, the two methods are connected in the following manner:

$$\underbrace{\tilde{G}}_{\text{EGD}} = \underbrace{F}_{\text{FIM}}^{1/2} \underbrace{\bar{G}}_{\text{NGD}}. \tag{13}$$

Therefore, our proposed EGD corresponds to a whitened version of NGD. The effect of this whitening is precisely to equalize the singular values of the gradient matrix, leading to accelerated grokking.

4.2 Comparison with Gradient-Filtering (Grokfast)

The *Grokfast* method proposed by Lee et al. (2024) induces fast grokking as follows: each row of the gradient matrix G is replaced by $g + F(g)$, where $F(g)$ is a low-pass filtered version of g, computed by aggregating with a large buffer of the past history of gradients. This has the desirable effect of boosting the low-frequency components of the gradient and attenuating the high-frequency components thereof.

Now, let $c_j := g^\top u_j$ be the jth component of g measured in the eigenbasis for $G^\top G$. From (12),

$$(GG^\top)^{-1/2} g = \sum_j (c_j/s_j) u_j. \tag{14}$$

This downweights the components of g aligned with large s_j—the "high-frequency" directions—so our EGD update inherits the filtering inductive bias of Grokfast (Lee et al., 2024) as a by-product. Crucially, unlike Grokfast, EGD *equalizes* the optimization speed across principal directions, yielding isotropic progress in the eigenspace. It also enjoys the following important properties:

- *Memory.* In contrast to Grokfast (Lee et al., 2024), which maintains a large buffer of past gradients, our proposed EGD method incurs **no** additional memory overhead beyond the current gradient (and, if used, a running spectral estimate).
- *Simple and Hyperparameter-free.* EGD introduces no extra tuning knobs. It therefore avoids the task-dependent, time-consuming hyperparameter sweeps that are important for Grokfast. The formula for EGD (11) is a lightweight, drop-in modification of (stochastic) gradient descent with a closed-form, per-step rescaling in the principal basis; it requires no schedulers, no momentum variants, and no buffering.
- *Theoretical foundations.* EGD comes with a spectral analysis guaranteeing equalized per-mode convergence rates, and consequently an accelerated exit from the test-error plateau (i.e., faster grokking). By comparison, Grokfast is a heuristic frequency filter without such guarantees.

5 Experimental Validation

5.1 Sparse Parity Problem

This is a well-known challenging problem in statistical learning theory (Barak et al., 2022). In addition, recent work has shown that this problem induces grokking in (stochastic) gradient descent (Merrill et al., 2023). An instance $\mathrm{Parity}(n, k)$ of this problem is as follows. n and k are positive integers with $k \leq n$. A random k-element subset S of $[n]$ is drawn once and for all, and then N i.i.d. samples $(x_1, y_1), \ldots, (x_N, y_N)$ are generated, where the x_i are i.i.d. uniform n-bit strings, and each y_i corresponds to the XOR of the bits of x_i restricted to the secret subset S, i.e $y_i := (-1)^{\sum_{j \in S} x_{ij}}$. The accuracy of a model $f : \{0,1\}^n \to \{-1, 1\}$ is counting the proportion of points in a large held-out test dataset (generated from the same distribution) have the true labels correctly predicted.

For the model, we consider a two-layer ReLU network $f(x) = \mathrm{sign}(v^\top \sigma(Wx))$ trained by optimizing hinge loss and weight decay with using different optimization strategies, on the parity problem with different values of n and k: $(n, k) \in \{(400, 2), (100, 3), (50, 4)\}$. The batch size is set to 32. For reproducibility, information on low-level details like learning rate, amount of weight decay, etc. is provided in Appendix B. We compare different optimization strategies: vanilla (stochastic) GD, our proposed method EGD (11) (applied only on the gradient matrix G for the hidden layer weights W), and an even simplified version of EGD where we replace each column of G.

The results are shown in Figure 3. As predicted by our theory, EGD groks remarkably early on in the optimization process (typically after only a few epochs). In contrast, vanilla (S)GD goes through an arbitrarily long plateau of stagnation of the test error before eventually grokking.

5.2 Modular Arithmetic

Another family of problems that exhibits grokking behavior is modular arithmetic. This class of problems has been extensively studied in the grokking literature (Power et al., 2022; Liu et al., 2023; Lee et al., 2024; Mohamadi et al., 2024; Nanda et al., 2023; Notsawo et al., 2023; Zhong et al., 2023; Gromov, 2023; Doshi et al., 2024; Prieto et al., 2025). To formally define this class of problems, an instance $\mathrm{Mod}(p, o)$ is defined by a prime modulus p and an operation $o \in \{+, \times\}$ over $\mathbb{Z}_p$. Training data consists of N i.i.d. samples (x_i, y_i), which are a fraction of all p^2 possible combinations. Here, $x_i = (a_i, b_i)$ is drawn uniformly from $\{0, 1, ..., p\} \times \{0, 1, ..., p\}$, and the label is calculated as

$$y_i = (a_i \; o \; b_i) \bmod p.$$

We train a two-layer ReLU network with cross-entropy loss and weight decay, using the same setup as in the sparse parity experiments. The complete set of hyperparameters is provided in Appendix B. Similarly to sparse parity, we compare vanilla (stochastic) GD, our proposed method EGD (11), and a simplified column-wise variant of EGD.

As shown in Figures 1 and 2, EGD groks considerably earlier than other methods, achieving high accuracy after only a few epochs in both modular addition and multiplication tasks.

6 Concluding Remarks

We studied grokking through the lens of gradient eigenspectral dynamics and proposed *Equalitarian Gradient Descent (EGD)*, a simple, hyperparameter-free modification of stochastic gradient descent that equalizes optimization speed across principal directions. By down-weighting high-frequency components while preserving progress on slow, symmetry-aligned modes, EGD provides a principled, spectrum-aware update that consistently shortens the test accuracy plateau without degrading final performance. Beyond empirical gains, our analysis clarifies how progress along low-frequency, task-aligned directions governs the timing of the generalization jump, and it yields compact diagnostics that relate early gradient spectra to later test improvement. The method is optimizer-agnostic, easy to integrate into existing training loops, and introduces no additional tuning knobs.

Limitations and Future Work. Our proposed EGD method is lightweight and straightforward to deploy. The following directions aim to make it even more efficient at larger scales and longer runs. We will explore optional low-cost spectral surrogates, including random projections and sketching to compress gradient information, randomized SVD or Lanczos methods to approximate the top k directions, Nyström and block-diagonal layerwise approximations, and streaming or online PCA to maintain running spectral estimates with minimal overhead. We also plan to study plug-and-play combinations with adaptive optimizers and weight decay, behavior under non-stationary data and curriculum schedules, and broader benchmarks beyond algorithmic tasks.

References

Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.

Shun-ichi Amari, Tomoko Ozeki, Ryo Karakida, Yuki Yoshida, and Masato Okada. Dynamics of learning in mlp: Natural gradient and singularity revisited. *Neural Computation*, 30(1):1–33, 2018.

Boaz Barak, Benjamin Edelman, Surbhi Goel, Sham Kakade, Eran Malach, and Cyril Zhang. Hidden progress in deep learning: Sgd learns parities near the computational limit. *Advances in Neural Information Processing Systems*, 35: 21750–21764, 2022.

Xander Davies, Lauro Langosco, and David Krueger. Unifying grokking and double descent. *arXiv preprint arXiv:2303.06173*, 2023.

Darshil Doshi, Tianyu He, Aritra Das, and Andrey Gromov. Grokking modular polynomials. 2024. URL `https://arxiv.org/abs/2406.03495`.

Andrey Gromov. Grokking modular arithmetic. 2023. URL `https://arxiv.org/abs/2301.02679`.

Yufei Huang, Shengding Hu, Xu Han, Zhiyuan Liu, and Maosong Sun. Unified view of grokking, double descent and emergent abilities: A perspective from circuits competition. *arXiv preprint arXiv:2402.15175*, 2024.

Tanishq Kumar, Blake Bordelon, Samuel J. Gershman, and Cengiz Pehlevan. Grokking as the transition from lazy to rich training dynamics. In *International Conference on Learning Representations (ICLR)*, 2024.

Jaerin Lee, Bong Gyun Kang, Kihoon Kim, and Kyoung Mu Lee. Grokfast: Accelerated grokking by amplifying slow gradients. 2024. URL `https://arxiv.org/abs/2405.20233`. v2, 5 Jun 2024.

Ziming Liu, Ouail Kitouni, Niklas S Nolte, Eric Michaud, Max Tegmark, and Mike Williams. Towards understanding grokking: An effective theory of representation learning. *Advances in Neural Information Processing Systems*, 35, 2022.

Ziming Liu, Eric J. Michaud, and Max Tegmark. Omnigrok: Grokking beyond algorithmic data. In *International Conference on Learning Representations (ICLR)*, 2023. URL `https://arxiv.org/abs/2210.01117`. Spotlight.

Clare Lyle, Gharda Sokar, Razvan Pascanu, and Andras Gyorgy. What can grokking teach us about learning under nonstationarity? *arXiv preprint arXiv:2507.20057*, 2025.

William Merrill, Nikolaos Tsilivis, and Aman Shukla. A tale of two circuits: Grokking as competition of sparse and dense subnetworks. *arXiv preprint arXiv:2303.11873*, 2023.

Mohamad Amin Mohamadi, Zhiyuan Li, Lei Wu, and Danica J. Sutherland. Why do you grok? a theoretical analysis on grokking modular addition. In *International Conference on Machine Learning (ICML)*, 2024.

Shikhar Murty, Pratyusha Sharma, Jacob Andreas, and Christopher D Manning. Grokking of hierarchical structure in vanilla transformers. In *The 61st Annual Meeting Of The Association For Computational Linguistics*, 2023.

Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. 2023. URL `https://arxiv.org/abs/2301.05217`. v3, 19 Oct 2023.

Pascal Notsawo, Hattie Zhou, Mohammad Pezeshki, Irina Rish, Guillaume Dumas, et al. Predicting grokking long before it happens: A look into the loss landscape of models which grok. *arXiv preprint arXiv:2306.13253*, 2023.

Pascal Notsawo, Guillaume Dumas, and Guillaume Rabusseau. Grokking beyond the euclidean norm of model parameters. In *International Conference on Machine Learning (ICML)*, 2025.

Yann Ollivier, Ludovic Arnold, Anne Auger, and Nikolaus Hansen. Information-geometric optimization algorithms: A unifying picture via invariance principles. *Journal of Machine Learning Research*, 18(18):1–65, 2017.

Razvan Pascanu and Yoshua Bengio. Revisiting natural gradient for deep networks. *arXiv preprint arXiv:1301.3584*, 2013.

Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets. 2022. URL `https://arxiv.org/abs/2201.02177`.

Lucas Prieto, Melih Barsbey, Pedro A. M. Mediano, and Tolga Birdal. Grokking at the edge of numerical stability. 2025. URL `https://arxiv.org/abs/2501.04697`.

David Saad and Sara A Solla. On-line learning in soft committee machines. *Physical Review E*, 52(4):4225, 1995.

Vimal Thilak, Etai Littwin, Shuangfei Zhai, Omid Saremi, Roni Paiss, and Joshua Susskind. The slingshot mechanism: An empirical study of adaptive optimizers and the grokking phenomenon. *arXiv preprint arXiv:2206.04817*, 2022.

Vikrant Varma, Rohin Shah, Zachary Kenton, János Kramár, and Ramana Kumar. Explaining grokking through circuit efficiency. *arXiv preprint arXiv:2309.02390*, 2023.

Thomas Walker, Ahmed Imtiaz Humayun, Randall Balestriero, and Richard Baraniuk. Grokalign: Geometric characterisation and acceleration of grokking. *arXiv preprint arxiv:2506.12284*, 2025.

Sumio Watanabe. *Algebraic geometry and statistical learning theory*, volume 25. Cambridge university press, 2009.

Haikun Wei, Jun Zhang, Florent Cousseau, Tomoko Ozeki, and Shun-ichi Amari. Dynamics of learning near singularities in layered networks. *Neural Comput.*, 20(3), 2008.

Yuki Yoshida and Masato Okada. Data-dependence of plateau phenomenon in learning with neural network—statistical mechanical analysis. *Advances in Neural Information Processing Systems*, 32, 2019.

Ziqian Zhong, Ziming Liu, Max Tegmark, and Jacob Andreas. The clock and the pizza: Two stories in mechanistic explanation of neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. URL `https://proceedings.neurips.cc/`.

Xuekai Zhu, Yao Fu, Bowen Zhou, and Zhouhan Lin. Critical data size of language models from a grokking perspective. *arXiv preprint arXiv:2401.10463*, 2024.

A Proofs

A.1 Analytic Theory for the Toy Problem

Proof of Theorem 1. First observe that the vanilla gradient descent dynamics can be expanded as follows:

$$\begin{aligned} w(k) &= w(k-1) - \eta X^\top (Xw(k-1) - Y)/n = Aw(k-1) + b \\ &= A^2 w(k-2) + Ab + b = \ldots = A^k w(0) + (A^{k-1} + \ldots + A + I)b \\ &= A^k w(0) + (I - A^k)(I - A)^{-1} b = A^k w(0) + (I - A^k)\eta^{-1}\hat{\Sigma}^{-1} b \\ &= A^k w(0) + (I - A^k)\hat{w}_{ols}. \end{aligned}$$

Now, for $z \sim \mathcal{N}(0, \Sigma)$ and $z_0 := \Sigma^{-1/2} z \sim \mathcal{N}(0, I)$, we can write

$$\begin{aligned} E_{test}(\hat{w}(k)) &= \mathbb{P}((z^\top e_1)(z^\top \hat{w}(k)) \le 0 \mid (z^\top e_1)(z^\top v) \le 0) \\ &= \mathbb{P}((z_0^\top \bar{e}_1)(z_0^\top \hat{w}(k)) \le 0 \mid (z_0^\top \bar{e}_1)(z_0^\top \bar{v}) \le 0) \\ &= \frac{\arccos(r_{e_1,v}) + \arccos(r_{\hat{w}(k),e_1}) - \arccos(r_{\hat{w}(k),v})}{2\arccos(r_{e_1,v})} \\ &= \frac{1}{2}\left(1 - \frac{\arccos(r_{\hat{w}(k),v}) - \arccos(r_{\hat{w}(k),e_1})}{\arccos(r_{e_1,v})}\right), \end{aligned}$$

where $\bar{w} := \Sigma^{1/2} w$ and $r_{w,v} := \bar{w}^\top \bar{v}/(\|\bar{w}\|\|\bar{v}\|)$ is the cosine of the angle between w and v, relative to the inner-product structure induced by Σ. The third line in the above display is a direct application of standard orthant probability formulae for bi-variate Gaussian random variables.

Note that $\bar{e}_1 = e_1$, and so $r_{e_1,v} = \cos(\theta)/\|v\| = \rho/\gamma =: r$. The result then follows upon invoking Lemma 2 to estimate $r_{\hat{w}(k),v}$ and $r_{\hat{w}(k),e_1}$. □

Proof of Corollary 1 (Grokking Profile of Vanilla GD). We know that

$$\mu_k = \alpha^k u_1 + (1 - \alpha^k) m_1/m_2 = \alpha^k (u_1 - m_1/m_2) + m_1/m_2 \asymp m_1/m_2,$$

provided $u_1 = O(1)$. We get

$$r_k = \mu_k/L_k \asymp \frac{1}{\sqrt{1 + (\beta^k u_2 m_2/m_1)^2 \varepsilon}}.$$

Thus, the condition $r_k \le r$ reduces to $(\beta^{k_*} u_2 m_2/m_1)^2 \varepsilon \asymp 1/r^2 - 1 = \varepsilon \cdot (1 - \rho^2)/\rho^2$, i.e

$$k_* \asymp \frac{\log\left(|u_2||\tan\theta| m_2/m_1\right)}{\log 1/\beta} \asymp \frac{1}{\eta\varepsilon}\log\tau,$$

as claimed. Note that we have used the fact that $\log 1/\beta = -\log(1 - \eta\varepsilon) \asymp \eta\varepsilon$, for small ε.

The case of small $|u_2|$ follows a similar argument to analyze the growth rate of $\arccos(r_k)$ and ultimately get $k_* \asymp (1/\eta)\log\frac{1}{\tau\varepsilon}$. □

A.2 Important Lemmas

The following lemma are easily proved proved via the law of large numbers.

Lemma 1. *For large n, we have the deterministic approximations*

$$\hat{\Sigma} \simeq \begin{pmatrix} m_2 & 0 \\ 0 & \varepsilon \end{pmatrix}, \quad \frac{1}{n} X^\top Y \simeq \begin{pmatrix} m_1 \\ 0 \end{pmatrix}, \quad \hat{w}_{ols} \simeq \begin{pmatrix} m_1/m_2 \\ 0 \end{pmatrix}, \tag{15}$$

where the notation "$\simeq$" ignores fluctuations of order $O_{\mathbb{P}}(n^{-1/2})$ in spectral or L_2 norm.

The following lemma is a direct consequence of the previous via equation (4).

Lemma 2. *For any deterministic vector $v \in \mathbb{R}^2$, it holds that*

$$A^k \simeq \begin{pmatrix} \alpha^k & 0 \\ 0 & \beta^k \end{pmatrix}, \quad w(k) \simeq \begin{pmatrix} \mu_k \\ \nu_k \end{pmatrix}, \quad w(k)^\top \Sigma w(k) \simeq L_k^2, \quad w(k)^\top \Sigma v \simeq \mu_k v_1 + \varepsilon \nu_k v_2, \tag{16}$$

where the notation "$\simeq$" ignores fluctuations of order $O_{\mathbb{P}}(n^{-1/2})$.

B Hyperparameters for Experiments in Section 5

Hyperparameters used in different tasks are listed in Table 1. In this table, n is the number of input bits for the subset parity task, DR is data ratio which shows what fraction of all possible combinations have been used as the training set, LR is the learning rate, WD is weight decay, and BS is batch size. In all of the cases ReLU have been used as the activation function.

Table 1: Hyperparameters for Different Tasks

Task	**Setting**	n	DR	Width	LR	WD	BS
Subset Parity	$k = 2$	400	NA	50	0.01	10^{-3}	32
	$k = 3$	100		100	0.042	10^{-2}	
	$k = 4$	50		100	0.023	10^{-2}	
Modular Add./Mult.	$p = 79$	NA	0.5	512	0.7	10^{-4}	512
	$p = 97$						
	$p = 127$						