Unveiling Memorization–Generalization Coexistence: A Case Study on Arithmetic Tasks with Label Noise

Linyu Liu[1] **Pinyan Lu**[1,2]
[1]Taylor Lab, Huawei Technologies Co., Ltd.
[2]Shanghai University of Finance and Economics
liulinyu001@gmail.com lu.pinyan@mail.shufe.edu.cn

Abstract

Highly over-parameterized models can simultaneously memorize noisy labels and generalize well, yet how these behaviors coexist remains poorly understood. In this work, we investigate the underlying mechanisms of this coexistence using modular arithmetic tasks under heavy label noise. Through extensive experiments on two-layer neural networks, we find that larger models tend to generalize better under appropriate optimization and model configurations, while noisy labels are memorized faster than clean data. Over-parameterized models internally form a generalization structure, but its expression in the output is suppressed by the need to fit noisy labels. Remarkably, even with 80% label noise, near-perfect test accuracy can be achieved by extracting this internal structure using frequency-based methods. We further propose a task-agnostic method to partition networks into generalization and memorization components. Although this subnetwork improves generalization, it is limited compared with frequency-based extraction, indicating that the generalization structure is distributed across neurons and motivating the development of new tools to retrieve generalizable knowledge from over-parameterized networks.

1 Introduction

Large Language Models succeed largely because increasing model capacity consistently improves generalization [1, 2, 3, 4]. In structured tasks, larger models exhibit emergent abilities to capture underlying rules from data [5]. However, real-world training data is often contaminated with noise, such as mislabeled samples or factual inconsistencies. While the scaling paradigm suggests that larger models perform better in clean environments [6], it remains unclear how they handle corrupted data. Specifically, we need to understand the internal competition between two distinct components: generalization, which is the discovery of structural rules that apply to unseen data, and memorization, which is the fitting of specific noisy labels that do not reflect the underlying rule.

Analyzing this interaction in realistic tasks is challenging because generalization is often difficult to diagnose mechanistically. Unseen examples may share local patterns, semantic features, or distributional correlations with the training data, so high test accuracy does not by itself reveal whether the model has learned a true rule or exploited shortcuts. To address this, we use modular arithmetic tasks (such as addition and multiplication modulo P) as a controlled experimental environment. These tasks are ideal for several reasons. First, they provide a clear distinction between rule-based generalization and point-wise memorization [7]. Second, they exhibit grokking [8, 9, 10], a phenomenon where generalization happens long after the model fully memorizes the training set. Third, these tasks allow for rigorous analysis because they have known analytical solutions under certain setups [11, 12].

Prior work has shown that networks can memorize corrupted labels while generalizing on modular arithmetic tasks [13]. It also showed that removing certain neurons can reduce memorization while

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

preserving generalization. Building on these findings, we systematically study how increasing model size affects generalization under label noise, and examine whether the benefits of increasing model size persist under extreme corruption. Test accuracy alone, however, may not reveal whether a model has learned the underlying rule. We therefore examine both the learned representations and the roles of individual neurons. This allows us to study how the model resolves the conflict between following the underlying rule and fitting corrupted labels. Our experiments cover different noise ratios, activation functions, optimizers, and regularization strengths. We focus on three key questions:

Q1. Under what conditions does increasing model size improve generalization when labels are noisy?

Q2. Can models still learn the underlying rule under extreme label noise, even when their test accuracy is low?

Q3. Can neuron pruning separate generalization from memorization without task-specific prior knowledge?

Through these experiments, we provide the following findings:

1. We show that scaling model size continues to improve generalization even with noisy data, provided that the optimization and regularization are properly configured.
2. We observe a counterintuitive learning pattern in which noisy labels are often memorized faster than the clean ones at a very early stage.
3. We find that models can *internally* recover the underlying algebraic rule under extreme label noise ($\sim 80\%$ noise), though its expression is suppressed by the need to interpolate noisy labels.
4. Although partitioning the network can improve generalization, generalization and memorization remain deeply entangled at the neuron level.

These results highlight both the robustness and the limitation of over-parameterized learning under corruption: models may learn generalizable rules even when forced to memorize noise, but neuron-level separation cannot fully extract this rule structure from memorization. The rest of the paper elaborates on these findings. Section 2 introduces the problem setup. Section 3 provides an overview of accuracy performance from both the end-of-training and dynamics perspectives (Q1). Section 4 analyzes how the learned rule is represented internally (Q2). Section 5 examines whether memorization and generalization can be separated at the neuron level (Q3). We defer related work to Appendix A.

2 Problem Setup

We study a modular arithmetic task $a \circ b = c$ following Power et al. [8], where $a, b, c \in \{0, \ldots, P-1\}$ and $\circ$ denotes a modular operation such as addition, subtraction, or multiplication. We focus primarily on modular addition and use other operations to test generality. The task is formulated as a P-class classification problem.

We train a two-layer neural network parameterized by θ:

$$\boldsymbol{f}_\theta(a, b) = \boldsymbol{W}\,\phi(\boldsymbol{U}\boldsymbol{e}_a + \boldsymbol{V}\boldsymbol{e}_b) + \boldsymbol{\mu}, \tag{1}$$

where $\theta = \{\boldsymbol{W}, \boldsymbol{U}, \boldsymbol{V}, \boldsymbol{\mu}\}$. Here, $\boldsymbol{e}_a, \boldsymbol{e}_b \in \mathbb{R}^P$ are one-hot encodings of a and b; $\boldsymbol{U}, \boldsymbol{V} \in \mathbb{R}^{M \times P}$ are first-layer weights; $\boldsymbol{W} \in \mathbb{R}^{P \times M}$ is the second-layer weight matrix; $\boldsymbol{\mu} \in \mathbb{R}^P$ is the bias; and M denotes the number of hidden neurons. The output $\boldsymbol{f}_\theta(a, b) \in \mathbb{R}^P$ is trained using cross-entropy loss to predict the one-hot label $\boldsymbol{e}_{a \circ b}$.

Unless stated otherwise, we fix $P = 113$ following Nanda et al. [9] and vary the model width as $M = 2^k + 1$ for $k \in \{4, \ldots, 12\}$, since ReLU models empirically generalize better at odd widths in these tasks [7].

Dataset. The total dataset is

$$\mathcal{D}_{\text{total}} = \{(a, b, c) : a, b \in \{0, \ldots, P-1\},\ c = a \circ b\}.$$

We randomly split $\mathcal{D}_{\text{total}}$ into training, validation, and test sets with ratios 50%, 20%, and 30%, respectively. The validation data is solely utilized in the post-hoc neuron selection procedure (Algorithm 1 in Section 5). In Appendix C.1, we also vary the training ratio while fixing the test ratio.

To introduce label noise, we randomly select a subset $\mathcal{D}_{\text{noise}} \subset \mathcal{D}_{\text{train}}$ with noise ratio α, i.e., $\frac{|\mathcal{D}_{\text{noise}}|}{|\mathcal{D}_{\text{train}}|} = \alpha$. For each $(a, b, c) \in \mathcal{D}_{\text{noise}}$, we define the observed label $\tilde{c}$ as

$$\tilde{c} \sim \text{Unif}\left(\{0, \cdots, P-1\} \setminus \{c\}\right), (a, b, c) \in \mathcal{D}_{\text{noise}}.$$

The resulting noisy training subset is given by

$$\tilde{\mathcal{D}}_{\text{noise}} \coloneqq \{(a, b, \tilde{c}) : (a, b, c) \in \mathcal{D}_{\text{noise}}\}.$$

Let $\mathcal{D}_{\text{clean}} = \mathcal{D}_{\text{train}} \setminus \mathcal{D}_{\text{noise}}$ denote the uncorrupted training subset. The noisy training set is $\tilde{\mathcal{D}}_{\text{train}} = \mathcal{D}_{\text{clean}} \cup \tilde{\mathcal{D}}_{\text{noise}}$. Validation and test labels remain uncorrupted.

Evaluation. To measure the performance of a model $\boldsymbol{f}$, we define its accuracy on a dataset $\mathcal{D}$ as

$$\text{Acc}\left(\boldsymbol{f}, \mathcal{D}\right) = \frac{1}{|\mathcal{D}|} \sum_{(a,b,c)\in\mathcal{D}} \mathbb{I}\left(\hat{c} = c\right),$$

where $\hat{c} = \arg\max_{c'} \left[\boldsymbol{f}\left(a, b\right)\right]_{c'}$, and $\mathbb{I}\left(\cdot\right)$ is the indicator function.

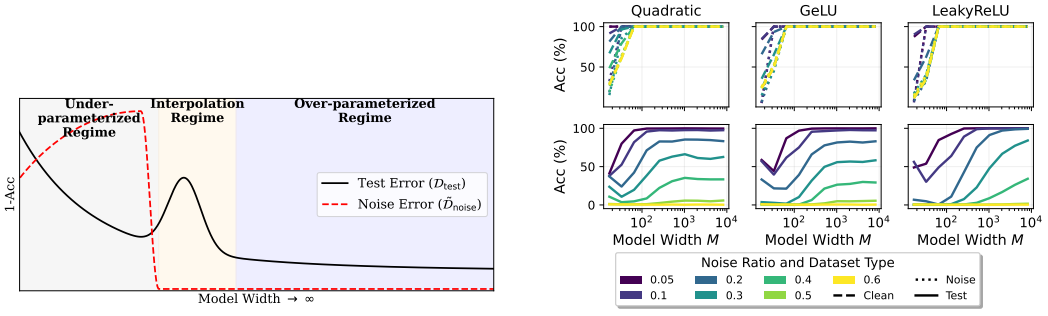
Optimizers. We train models with cross-entropy loss using Adam, AdamW [14], and Muon [15], with weight decay controlling regularization strength. We also experiment with SGD, but it is inefficient and requires careful hyperparameter tuning to achieve generalization (see Figure 14).

Unless specified otherwise, all results are reported after 200,000 epochs of full-batch training. We apply a linear learning-rate warmup over the first 10 steps. The default setting uses ReLU activations with AdamW (learning rate 10^{-3}, weight decay 0.1, random seed 42) on the modular addition task. We also repeat experiments with seeds 0 and 123 for the main findings (see Appendices C.2 and D.6).

3 When Bigger Is Better

In this section, we investigate how the presence of label noise interacts with model capacity in modular arithmetic tasks. We approach this question from two complementary perspectives. First, we examine end-of-training performance, asking under which conditions—such as optimization and regularization choices—increasing model size consistently improves generalization despite noise. Second, we analyze the training dynamics to study whether the model learns clean labels and noisy labels at different rates.

3.1 End-of-Training Performance



(a) Illustration of double descent. Beyond the interpolation threshold, the over-parameterized regime corresponds to the second descent, where increasing model size leads to improved generalization, even after fitting all training samples, including label noise.

(b) Performance across various activation functions. Generally, test accuracy is non-decreasing in the over-parameterized regime, but ceilings of test accuracy exist for the quadratic and GeLU activations.

Figure 1: Model-wise double descent and performance across various activation functions.

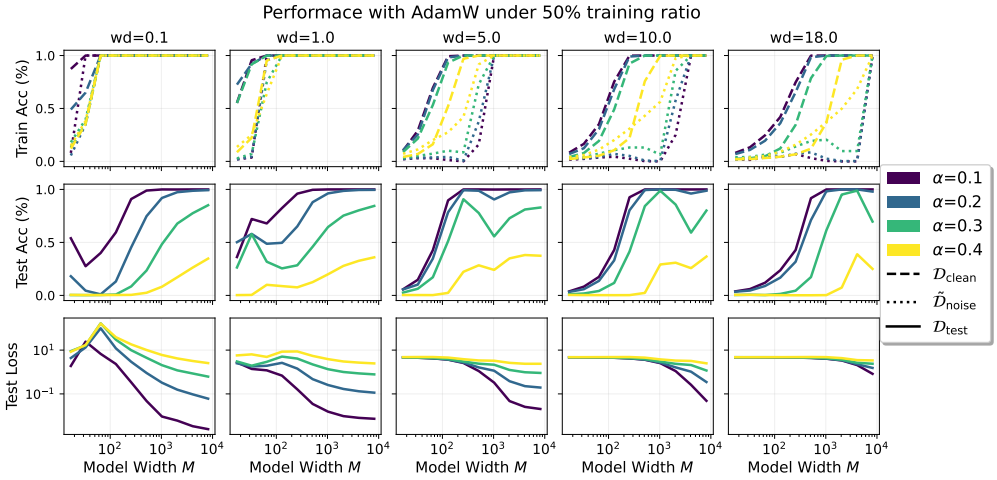


Figure 2: Performance of ReLU models trained with AdamW under varying weight decay. The first row shows training accuracy on clean ($\mathcal{D}_{\text{clean}}$) and noisy ($\tilde{\mathcal{D}}_{\text{noise}}$) data to characterize memorization. The second row shows test accuracy, and the third row shows test cross-entropy loss. Increasing weight decay shifts the interpolation threshold to larger sizes, delaying the second-descent regime.

We robustly observe the *double descent* phenomenon of the test *error* ($1 -$ test accuracy) across different activation functions, noise-to-training ratios, optimizers (including AdamW, Adam, and Muon), and arithmetic tasks (see the full results in Section C.1). As illustrated in Fig. 1a, *double descent* refers to the non-monotonic test-error curve in which generalization first worsens near the interpolation threshold and then improves again as model size further increases. In the over-parameterized regime, continually enlarging the model leads to non-decreasing test accuracy, while the threshold for entering this regime depends on the regularization strength. We summarize the observations in Result 3.1.

Result 3.1. *Double descent is consistently observed across varying setups:*

1. Within the over-parameterized regime, bigger models typically generalize better.

2. Strengthening regularization will shift models from the over-parameterized regime toward the under-parameterized regime. (Figure 2)

3. Model sizes of test-accuracy saturation are activation-dependent (Figure 1b)

Regularization does play a role in shaping the double-descent curve by controlling the effective model complexity (EMC) [16]. Intuitively, stronger regularization restricts the set of functions that can be realized during training, effectively reducing the model's capacity. When the EMC is limited, learning signals for generalization and memorization compete more strongly. As a result, increasing regularization can be beneficial when memorization of label noise is undesirable.

Models with quadratic or GeLU activation functions exhibit clear ceilings in test accuracy as model size increases within the explored model size range (width $< 10{,}000$, Figure 1b). We include quadratic activations here because their analytical solutions are available (see Section 4). One possible explanation for these ceilings is that the symmetry of these activations constrains how noise can be absorbed while preserving generalization. In contrast, ReLU activations store noise in a non-symmetric manner, while the representation of generalization remains symmetric (see Section 4), allowing model capacity to be used more flexibly and leading to improved generalization.

We also observe that models are more confident (i.e., exhibit lower test cross-entropy loss) in the over-parameterized regime than in the under-parameterized regime. Larger models are generally more confident (Figure 2, bottom row). Cross-entropy loss reflects the confidence calibration of model predictions. In the under-parameterized regime, models often predict the true label even for corrupted samples. To align the predicted probability with the portion of correctness, the model assigns lower probabilities, resulting in higher uncertainty and higher test cross-entropy. In contrast, in the over-parameterized regime, larger models can separate data with larger margins while still

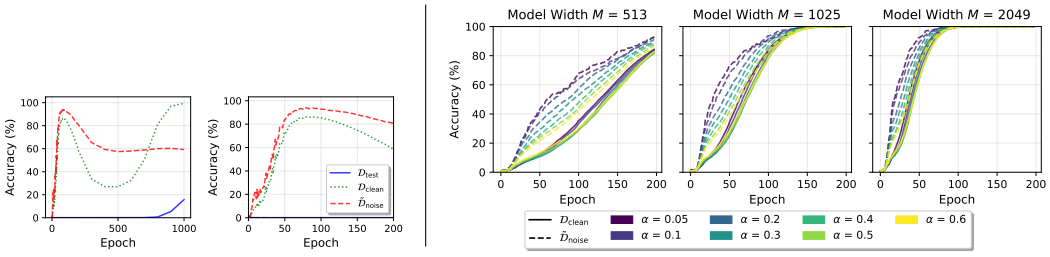
satisfying the same ℓ_2 regularization constraint. Larger margins lead to more confident predictions and, consequently, lower test cross-entropy loss.

Remarks on model misspecification. We note that some model architectures are unable to interpolate all noisy labels, in which case the over-parameterized regime does not strictly exist. For example, models with tied first-layer weights ($\boldsymbol{U} = \boldsymbol{V}$) cannot fit conflicting observed labels for (a, b) and (b, a) in modular addition. In such cases, test accuracy does not vary monotonically with model size, and selecting an appropriate regularization strength may be more effective than increasing model capacity. We provide a detailed discussion in Section C.3.

3.2 Training Dynamics

We observe a counter-intuitive phenomenon in the training dynamics: noisy labels are consistently memorized faster than clean labels at the initial training epochs, as summarized in Result 3.2. Although the two subsets are fitted at similar rates, this inversion of the typical learning order is robust across various optimizers, including AdamW, Adam, and Muon (see Section C.5 for details). Even for SGD, which generally requires significantly longer timescales to achieve generalization, this trend remains consistent (Figure 14).

Result 3.2. *In both under- and over-parameterized regimes, the model prioritizes the memorization of noisy labels over the fitting of clean labels.*



(a) Under-parameterized regime.

(b) Over-parameterized regime.

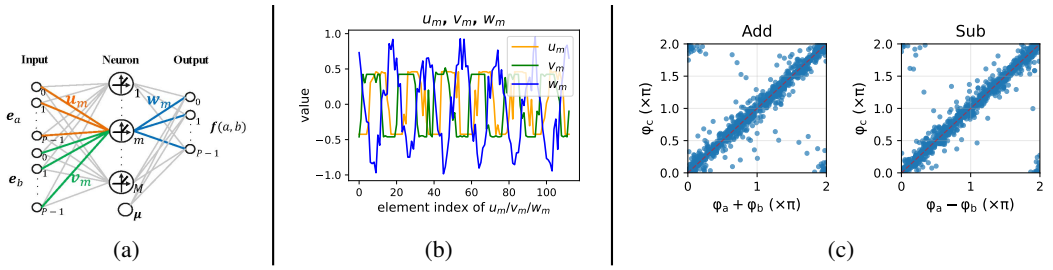
Figure 3: The faster memorization of noisy labels is observed in both the under- and over-parameterized regimes. **(a)** Under-parameterized learning dynamics with a zoom-in on the first 200 epochs ($\alpha = 0.3$, $M = 2049$, Adam with weight decay 0.001). **(b)** Over-parameterized learning dynamics under AdamW. In both cases, noisy-label accuracy rises earlier than clean-label accuracy.

Selected learning curves are shown in Figure 3. The faster memorization of noisy labels appears in both regimes, but the over-parameterized case in Figure 3b further reveals two patterns: (i) increased model capacity accelerates the memorization process, aligning with empirical findings in large language models [17]; and (ii) the gap between the memorization speeds of noise and clean data becomes more pronounced at lower noise ratios α.

While previous studies (e.g., Li et al. [18]) show that models tend to fit typical samples first by leveraging distinct learning phases or initialization distances, making early stopping effective, our findings reveal a complementary *early-stage* dynamic: under full-batch training, noisy labels can receive a larger update per sample, causing their accuracy to rise earlier before clean patterns take over. Importantly, this phenomenon is not restricted to two-layer networks on modular arithmetic tasks. In Section C.4, we provide a gradient-based discussion and show that a similar early noisy-over-clean learning order also appears on MNIST with a three-layer neural network under full-batch training. Thus, while modular arithmetic provides a controlled setting where the behavior can be cleanly measured and analyzed, the phenomenon may reflect a broader property of early full-batch optimization under explicit label corruption. Additional results are provided in Section C.5.

4 Representation of Generalization

We address Q2 by asking whether the apparent failure of ReLU networks under extreme label noise reflects the absence of a learned internal rule, or whether such a rule is learned but obscured by noisy outputs. We find evidence for the latter by separating the internal representation of the generalizable



(a) (b) (c)

Figure 4: **(a)** A neuron is represented by $(\boldsymbol{u}_m, \boldsymbol{v}_m, \boldsymbol{w}_m)$. **(b)** A representative neuron from a trained ReLU model exhibits periodicity. **(c)** Relationship of phases under $M = 1025$ and $\alpha = 0.3$. Each point represents a neuron, where φ_a, φ_b, and φ_c are the phases of $\boldsymbol{u}^G$, $\boldsymbol{v}^G$, and $\boldsymbol{w}^G$, respectively.

rule from that of memorization. Specifically, our main finding is that, although ReLU networks do not have the same functional form as networks with quadratic activations, their generalization component closely matches the analytical solution known for quadratic activations under *clean* data [11] (Equation (2)). This observation suggests a way to separate the learned rule from memorized noise: retain the dominant frequency component aligned with this analytical structure and remove the residual components.

Analytical solution for NN with quadratic activation. The two-layer NN (1) can be equivalently written as the summation of the neurons:

$$\boldsymbol{f}_\theta(a,b) = \sum_{m\in\mathcal{M}} \boldsymbol{w}_m \phi\left(\boldsymbol{u}_m^\top \boldsymbol{e}_a + \boldsymbol{v}_m^\top \boldsymbol{e}_b\right) + \boldsymbol{\mu},$$

where $\mathcal{M} = \{1, \cdots, M\}$. The triple, $(\boldsymbol{u}_m, \boldsymbol{v}_m, \boldsymbol{w}_m)$, represents the m^{th} neuron. Here, $\boldsymbol{w}_m$ is the m^{th} column vector of $\boldsymbol{W}$, and $\boldsymbol{u}_m^\top$ and $\boldsymbol{v}_m^\top$ are the m^{th} row vectors of $\boldsymbol{U}$ and $\boldsymbol{V}$, respectively. Figure 4 (a) visualizes this decomposition. The margin-maximizing solution for modular addition with quadratic activation $\phi(z) = z^2$ takes the form [19, 11]

$$u_{mi} = \lambda\cos(\tfrac{2\pi}{P}\omega_m i + \varphi_m^{(a)}), v_{mj} = \lambda\cos(\tfrac{2\pi}{P}\omega_m j + \varphi_m^{(b)}), w_{mk} = \lambda\cos(\tfrac{2\pi}{P}\omega_m k + \varphi_m^{(c)}), \quad (2)$$

with $\varphi_m^{(a)} + \varphi_m^{(b)} = \varphi_m^{(c)}$ and $\omega_m \in \{1, \ldots, \frac{P-1}{2}\}$ (Prop. B.1). This solution will achieve 100% accuracy on all the clean samples. Empirically, ReLU models trained on clean data also develop a periodic pattern in $(\boldsymbol{u}_m, \boldsymbol{v}_m, \boldsymbol{w}_m)$ (Figure 4 (b)), but there is no explicit solution for them.

Frequency filtration separates rule learning from noise memorization. Drawing inspiration from the analytical solution of the NN with quadratic activation, we hypothesize that the generalization signal in a ReLU neuron is carried by its dominant frequency. We perform a Fourier-based decomposition on each hidden neuron $m \in \mathcal{M}$, characterized by the parameters $(\boldsymbol{u}_m, \boldsymbol{v}_m, \boldsymbol{w}_m)$. Specifically, we apply a filter to isolate the frequency ω with the maximum magnitude in the weight vector $\boldsymbol{w}_m$, and retrieve the same frequency component from $\boldsymbol{u}_m$ and $\boldsymbol{v}_m$. We denote the filtered weights by $(\boldsymbol{u}_m^G, \boldsymbol{v}_m^G, \boldsymbol{w}_m^G)$ and define the residual weights $(\boldsymbol{u}_m^R, \boldsymbol{v}_m^R, \boldsymbol{w}_m^R)$ by subtracting the filtered weights from the original weights. The formal mathematical definitions of this filtration process are detailed in Appendix D.1. We subsequently construct two sub-networks: one composed of the dominant frequency components, denoted by $\{\boldsymbol{U}^G, \boldsymbol{V}^G, \boldsymbol{W}^G, \boldsymbol{\mu}\}$, and another consisting of the residuals, $\{\boldsymbol{U}^R, \boldsymbol{V}^R, \boldsymbol{W}^R, \boldsymbol{\mu}\}$. By evaluating these partitioned models separately, we quantify their respective contributions to memorization and generalization.

Result 4.1. *The generalization representation of ReLU models closely matches the analytical solution* (2) *for modular addition and subtraction tasks.*

Figure 5 (left) shows that the dominant-frequency sub-network achieves high test accuracy even under severe label noise, whereas the residual component is much more aligned with noisy-label memorization. This provides direct evidence that generalization and memorization coexist in the trained network but are encoded differently in frequency space.

The rule-separation effect of FF is surprisingly strong. Even when the model is trained with 80% label noise, the dominant-frequency sub-network can recover near-perfect test accuracy, indicating that the model has internally learned the correct rule although its raw output is heavily corrupted by noise

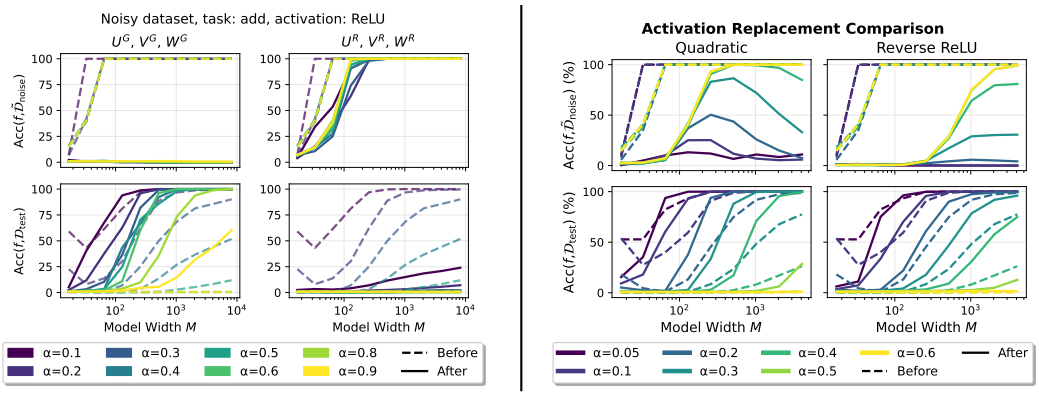


Figure 5: *Left:* FF for ReLU networks. The dominant-frequency component $\{U^G, V^G, W^G\}$ recovers high test accuracy (after FF), while the residual component $\{U^R, V^R, W^R\}$ mainly retains noise memorization. *Right:* Replacing the trained ReLU activation with quadratic or reverse ReLU preserves or even improves test accuracy and reduces noisy-label accuracy, suggesting that the learned weights already encode a rule representation compatible with other activation functions.

memorization. Conversely, the residual sub-network largely preserves the ability to fit noisy labels while contributing little to test generalization. This sharp contrast suggests that label noise does not simply prevent the model from learning the rule. Instead, the rule is learned but masked by additional residual frequency components used for memorization. We further observe that FF also improves test accuracy for models trained on noise-free samples, for networks with quadratic activations, and for models on modular subtraction tasks, suggesting that dominant-frequency filtration is not merely removing noisy artifacts but extracting a general rule representation. Preliminary evidence shows that this frequency-based filtering idea can also be extended to a Transformer (see Appendix D.2).

To test whether this rule recovery is specific to the selected dominant frequency and learned weights, we perform control experiments with conjugate-pair filtering, random-frequency filtering, and untrained networks. These controls confirm that high test accuracy requires both the trained weights and the proper choice of dominant frequencies, rather than the filtering prior alone (see Appendix D.5).

Additional evidence for the analytical structure. We provide two complementary checks that the ReLU representation is coarsely aligned with the analytical solution. First, activation replacement shows that the learned rule does not rely on the exact shape of ReLU: after training, replacing ReLU with quadratic or reverse ReLU (defined as $\phi(x) = \max\{-x, 0\}$) preserves and often enhances generalization while reducing noisy-label accuracy (Figure 5 (right)). This is consistent with the hypothesis that the weights have already formed a symmetric rule representation.

Second, the phases of the filtered dominant components match the phase relationship in the analytical solution. For addition, the phases satisfy $\varphi_m^{(a)} + \varphi_m^{(b)} \approx \varphi_m^{(c)}$; for subtraction, they satisfy $\varphi_m^{(a)} - \varphi_m^{(b)} \approx \varphi_m^{(c)}$ (Figure 4 (c)). These results, together with FF and activation replacement, support the same conclusion: ReLU networks generalize by forming a frequency-structured representation close to the quadratic analytical solution, enabling the disentanglement of the generalization and memorization in the internal representation. Additional experiment results are deferred to Appendix D.3-D.4.

This evidence suggests that FF can disentangle memorization from generalization. However, FF is inherently limited in two aspects: (i) it does not yield a strict functional decomposition, since $\boldsymbol{f}_\theta \neq \boldsymbol{f}_{\{\boldsymbol{W}^G, \boldsymbol{U}^G, \boldsymbol{V}^G, \boldsymbol{\mu}\}} + \boldsymbol{f}_{\{\boldsymbol{W}^R, \boldsymbol{U}^R, \boldsymbol{V}^R, \boldsymbol{\mu}\}}$; and (ii) it relies on prior knowledge of task-specific structure. This motivates the question of whether memorization and generalization can instead be separated in a task-agnostic manner (Q3).

5 Network Partitioning

We now address Q3 by asking whether generalization and memorization can be separated at the neuron level. Specifically, we investigate whether a trained network can be partitioned into two neuron subsets: one subset $\mathcal{M}^G \subseteq \mathcal{M}$ responsible for generalization, and another subset $\mathcal{M}^R \subseteq \mathcal{M}$

responsible for memorizing label noise. Selecting $\mathcal{M}^G$ for generalization is equivalent to pruning the remaining neurons from the full network. For a selected neuron subset, the corresponding sub-networks are defined as

$$\boldsymbol{f}_{\mathcal{M}^G}(a,b) = \sum_{m\in\mathcal{M}^G} \boldsymbol{w}_m \phi\Big(\boldsymbol{u}_m^\top \boldsymbol{e}_a + \boldsymbol{v}_m^\top \boldsymbol{e}_b\Big) + \boldsymbol{\mu},\ \boldsymbol{f}_{\mathcal{M}^R}(a,b) = \sum_{m\in\mathcal{M}^R} \boldsymbol{w}_m \phi\Big(\boldsymbol{u}_m^\top \boldsymbol{e}_a + \boldsymbol{v}_m^\top \boldsymbol{e}_b\Big) + \boldsymbol{\mu}. \quad (3)$$

Removing the output bias ($\boldsymbol{\mu}$) has little effect on the reported accuracies in the tested setting (Appendix E.2). We rank neurons using an importance score and select high-scoring neurons for generalization and low-scoring neurons for noise memorization. The full selection procedure is given in Algorithm 1.

Algorithm 1 Neuron Selection for Network Decomposition

Input: Network $\boldsymbol{f}_\theta$ with M neurons $\mathcal{M}$; Metric $S(m)$; $\mathcal{D}_{\text{val}}$; $\tilde{\mathcal{D}}_{\text{noise}}$; Threshold τ.
Initialize: $\mathcal{M}^G \leftarrow \emptyset, \mathcal{M}^R \leftarrow \emptyset$.
{// Step 1: Ranking neurons}
Calculate score $s_m = S(m)$ for each $m \in \mathcal{M}$.
Sort neurons descending: $\{m_{\pi(1)}, \ldots, m_{\pi(M)}\}$ where $s_{m_{\pi(i)}} \geq s_{m_{\pi(j)}}$ for $i < j$.
{// Step 2: Selection for generalization}
for $k \in \{1, \ldots, M\}$ **do**
  $\mathcal{M}_k = \{m_{\pi(1)}, \ldots, m_{\pi(k)}\}$; $A_k = \text{Acc}(\boldsymbol{f}_{\mathcal{M}_k}, \mathcal{D}_{\text{val}})$.
end for
$k^G = \arg\max_k A_k$; $\gamma^G = k^G/M$ and $\mathcal{M}^G = \{m_{\pi(1)}, \ldots, m_{\pi(k^G)}\}$.
{// Step 3: Minimal selection for noise memorization}
for $k \in \{1, \ldots, M\}$ **do**
  $\mathcal{M}'_k = \{m_{\pi(M-k+1)}, \ldots, m_{\pi(M)}\}$. {Bottom-up}
  if $\text{Acc}(\boldsymbol{f}_{\mathcal{M}'_k}, \tilde{\mathcal{D}}_{\text{noise}}) \geq \tau$ **then**
    $k^R = k$, **break**.
  end if
end for
$\gamma^R = k^R/M$ and $\mathcal{M}^R = \{m_{\pi(M-k^R+1)}, \ldots, m_{\pi(M)}\}$.
Output: Ratios γ^G, γ^R; Selected neurons $\mathcal{M}^G$, $\mathcal{M}^R$; Sub-networks $\boldsymbol{f}^G = \boldsymbol{f}_{\mathcal{M}^G}$, $\boldsymbol{f}^R = \boldsymbol{f}_{\mathcal{M}^R}$.

A natural importance score is the inverse participation ratio (IPR) [20, 21, 19, 13], proposed and applied in Doshi et al. [13]. It measures the periodicity of each neuron and is defined as

$$\text{IPR}_m \coloneqq \left(\frac{\|\tilde{\boldsymbol{w}}_m\|_4}{\|\tilde{\boldsymbol{w}}_m\|_2}\right)^4, \quad (4)$$

where $\|\cdot\|_p$ is the ℓ_p-norm and $\tilde{\boldsymbol{w}}_m$ is the Fourier transform of $\boldsymbol{w}_m$. Equation (4) implies $\text{IPR}_m \in [1/P, 1]$, where a larger value indicates stronger periodicity. This metric is well-suited for modular addition [13], whose generalizable solutions rely on periodic structures.

To avoid relying on a task-specific Fourier prior, we also propose a task-agnostic score, neuron strength (Str.), defined as

$$\text{Str}_m \coloneqq \|\boldsymbol{w}_m\|_\infty \phi\left(\|\boldsymbol{u}_m\|_\infty + \|\boldsymbol{v}_m\|_\infty\right), \quad (5)$$

where $\|\cdot\|_\infty$ denotes the ℓ_∞-norm. Intuitively, Str. measures the largest possible contribution of a neuron to the output margin. Unlike IPR, it does not assume periodicity and can therefore be applied to tasks such as modular multiplication, where periodicity in the original input domain is no longer a direct proxy for generalization.

Result 5.1. *Neurons that contribute most to generalization typically exhibit higher neuron strength. Specifically, neurons with high IPR also attain high Str. in modular addition and subtraction.*

Result 5.2. *The two subnetworks do not overlap when model width is sufficiently large, i.e., $\gamma^G + \gamma^R \leq 1$, where $\gamma^G = \frac{|\mathcal{M}^G|}{|\mathcal{M}|}$ and $\gamma^R = \frac{|\mathcal{M}^R|}{|\mathcal{M}|}$. (Figure 6 right)*

Figure 6 (left) shows that IPR and Str. are highly correlated. We also observe that a larger noise ratio reduces the number of high-IPR or high-Str. neurons. The optimal neuron ratio for generalization

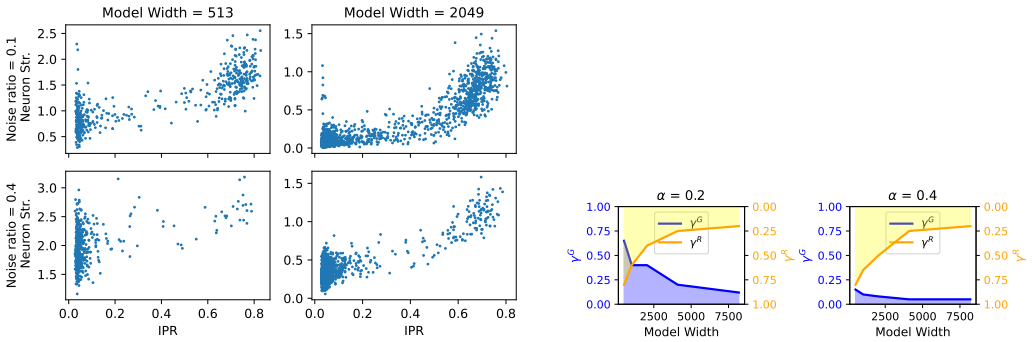


Figure 6: *Left:* IPR versus Str. for individual neurons on the modular addition task. Neurons with stronger periodicity also tend to have larger Str., supporting Str. as a task-agnostic proxy for neuron importance. *Right:* Neuron selection ratios using Str. on the modular addition task. A higher noise ratio leads to a smaller sub-network for generalization (γ^G). Two sub-networks have no overlapping neurons under sufficiently large model width.

(γ^G) is also lower at a higher noise ratio (Figure 6 right). These may explain the deterioration of generalization under heavy label noise. Additional results in Appendix E show that Str. yields performance comparable to IPR on addition and subtraction, while being more effective on modular multiplication, where IPR is less suitable.

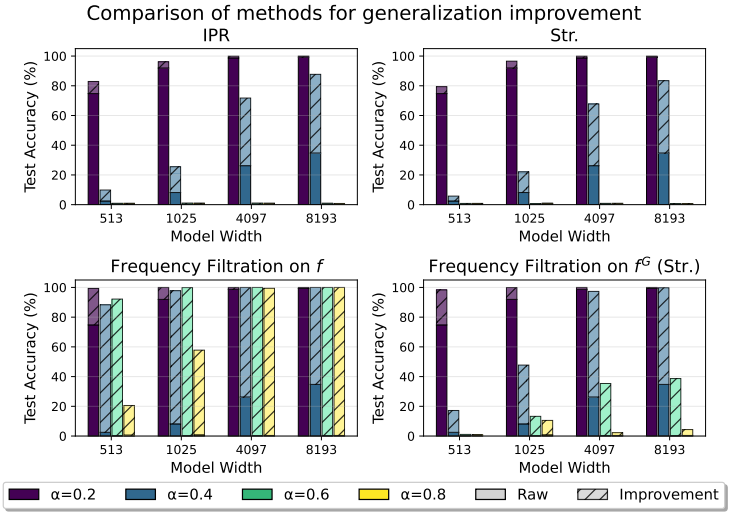


Figure 7: Generalization improvement on modular addition. IPR and Str. achieve similar performance. Applying FF to the sub-network f^G selected by Str. further improves generalization, but the gain remains substantially smaller than applying FF directly to the full model f.

Result 5.3. *Generalization improvement from neuron selection/pruning is limited compared with FF.*

Figure 7 shows that neuron selection can improve generalization, especially under light label noise, and that the task-agnostic Str. score performs comparably to the task-specific IPR score. However, this improvement is much weaker than the gain achieved by applying FF directly to the full model. Even applying FF after selecting the generalization sub-network f^G does not recover the performance of FF on the original network. This contrast indicates that generalizable structure is not localized to a small subset of neurons; instead, it is diffusely encoded across the network and intertwined with components used for noise memorization.

This comparison clarifies the limitation of task-agnostic post-hoc neuron selection. Unlike FF, which uses the domain-specific Fourier structure of the task to extract the latent rule, neuron selection can only identify neurons that are more important on average. As a result, pruning may partially improve

generalization but cannot fully unleash the rule already learned by the model. This suggests two broader implications: future learning paradigms may need to explicitly isolate memorization from logical inference during training, and post-hoc knowledge extraction may require domain-specific structural priors rather than generic neuron-importance scores.

Conclusion. Our results demonstrate that over-parameterized models can internally learn underlying rules even under explicit label noise, though this generalization remains heavily entangled with memorization across distributed neurons. While task-agnostic neuron selection provides only limited improvement, frequency filtration can successfully recover latent rules when a suitable structural prior is available. Crucially, we observe that these core entanglement mechanisms extend beyond algebraic tasks to architectures like Transformers and datasets such as MNIST. We acknowledge that our setup inherently abstracts away the complex variability of natural vision or language tasks, but this controlled setting is necessary to provide a rigorous, verifiable boundary between structural generalization and unstructured memorization. Ultimately, this foundational study highlights a fundamental limitation of post-hoc pruning-based extraction, suggesting that fully disentangling these processes in practical applications will require novel training paradigms or architectural designs with explicit function separation.

Acknowledgments and Disclosure of Funding

We thank our colleagues Xiaodong Ai, Zhengyang Ai, Jingxian Tang, Hangkai Hu, and Siwei Liu for helping organize research discussions and for their helpful comments on the experiments and results. This research received no specific grant from any funding agency. The authors declare no competing interests.

References

[1] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[2] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[3] Tom Henighan, Jared Kaplan, Mor Katz, Mark Chen, Christopher Hesse, Jacob Jackson, Heewoo Jun, Tom B Brown, Prafulla Dhariwal, Scott Gray, et al. Scaling laws for autoregressive generative modeling. *arXiv preprint arXiv:2010.14701*, 2020.

[4] 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.

[5] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022.

[6] Yizhou Liu, Ziming Liu, and Jeff Gore. Superposition yields robust neural scaling. *arXiv preprint arXiv:2505.10465*, 2025.

[7] Adam Pearce, Asma Ghandeharioun, Nada Hussein, Nithum Thain, Martin Wattenberg, and Lucas Dixon. Do machine learning models memorize or generalize?, August 2023. URL `https://pair.withgoogle.com/explorables/grokking/`. Accessed: 2024-01-20.

[8] 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.

[9] Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. In *The Eleventh International Conference on Learning Representations*, 2023.

[10] Ziqian Zhong, Ziming Liu, Max Tegmark, and Jacob Andreas. The clock and the pizza: Two stories in mechanistic explanation of neural networks. *Advances in neural information processing systems*, 36:27223–27250, 2023.

[11] Depen Morwani, Benjamin L Edelman, Costin-Andrei Oncescu, Rosie Zhao, and Sham M Kakade. Feature emergence via margin maximization: case studies in algebraic tasks. In *The Twelfth International Conference on Learning Representations*, 2024.

[12] Yuandong Tian. Provable scaling laws of feature emergence from learning dynamics of grokking. *arXiv preprint arXiv:2509.21519*, 2025.

[13] Darshil Doshi, Aritra Das, Tianyu He, and Andrey Gromov. To grok or not to grok: Disentangling generalization and memorization on corrupted algorithmic datasets. In *The Twelfth International Conference on Learning Representations*, 2024.

[14] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[15] Keller Jordan, Yuchen Jin, Vlado Boza, You Jiacheng, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. *URL https://kellerjordan. github. io/posts/muon*, 6.

[16] Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. Deep double descent: Where bigger models and more data hurt. *Journal of Statistical Mechanics: Theory and Experiment*, 2021(12):124003, 2021.

[17] Kushal Tirumala, Aram Markosyan, Luke Zettlemoyer, and Armen Aghajanyan. Memorization without overfitting: Analyzing the training dynamics of large language models. *Advances in Neural Information Processing Systems*, 35:38274–38290, 2022.

[18] Mingchen Li, Mahdi Soltanolkotabi, and Samet Oymak. Gradient descent with early stopping is provably robust to label noise for overparameterized neural networks. In *International conference on artificial intelligence and statistics*, pages 4313–4324. PMLR, 2020.

[19] Andrey Gromov. Grokking modular arithmetic. *arXiv preprint arXiv:2301.02679*, 2023.

[20] Romualdo Pastor-Satorras and Claudio Castellano. Distinct types of eigenvector localization in networks. *Scientific reports*, 6(1):18847, 2016.

[21] Steven M Girvin and Kun Yang. *Modern condensed matter physics*. Cambridge University Press, 2019.

[22] Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019. doi: 10.1073/pnas.1903070116. URL `https://www.pnas.org/doi/abs/10.1073/pnas.1903070116`.

[23] Zitong Yang, Yaodong Yu, Chong You, Jacob Steinhardt, and Yi Ma. Rethinking bias-variance trade-off for generalization of neural networks. In *International Conference on Machine Learning*, pages 10767–10777. PMLR, 2020.

[24] Mikhail Belkin, Daniel Hsu, and Ji Xu. Two models of double descent for weak features. *SIAM Journal on Mathematics of Data Science*, 2(4):1167–1180, 2020.

[25] Madhu S Advani, Andrew M Saxe, and Haim Sompolinsky. High-dimensional dynamics of generalization error in neural networks. *Neural Networks*, 132:428–446, 2020.

[26] Ben Adlam and Jeffrey Pennington. The neural tangent kernel in high dimensions: Triple descent and a multi-scale theory of generalization. In *International Conference on Machine Learning*, pages 74–84. PMLR, 2020.

[27] Peter L Bartlett, Philip M Long, Gábor Lugosi, and Alexander Tsigler. Benign overfitting in linear regression. *Proceedings of the National Academy of Sciences*, 117(48):30063–30070, 2020.

[28] Song Mei and Andrea Montanari. The generalization error of random features regression: Precise asymptotics and the double descent curve. *Communications on Pure and Applied Mathematics*, 75(4):667–766, 2022.

[29] Trevor Hastie, Andrea Montanari, Saharon Rosset, and Ryan J Tibshirani. Surprises in high-dimensional ridgeless least squares interpolation. *Annals of statistics*, 50(2):949–986, 2022.

[30] Vidya Muthukumar, Kailas Vodrahalli, Vignesh Subramanian, and Anant Sahai. Harmless interpolation of noisy data in regression. *IEEE Journal on Selected Areas in Information Theory*, 1(1):67–83, 2020.

[31] Preetum Nakkiran, Prayaag Venkat, Sham Kakade, and Tengyu Ma. Optimal regularization can mitigate double descent. *arXiv preprint arXiv:2003.01897*, 2020.

[32] Ali Hussaini Umar, Franky Kevin Nando Tezoh, Jean Barbier, Santiago Acevedo, and Alessandro Laio. The effect of label noise on the information content of neural representations. *arXiv preprint arXiv:2510.06401*, 2025.

[33] Matteo Gamba, Erik Englesson, Mårten Björkman, and Hossein Azizpour. Deep double descent via smooth interpolation. *arXiv preprint arXiv:2209.10080*, 2022.

[34] Gowthami Somepalli, Liam Fowl, Arpit Bansal, Ping Yeh-Chiang, Yehuda Dar, Richard Baraniuk, Micah Goldblum, and Tom Goldstein. Can neural nets learn the same model twice? investigating reproducibility and double descent from the decision boundary perspective. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, pages 13699–13708, 2022.

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

[36] 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.

[37] 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*, 35:34651–34663, 2022.

[38] Neil Rohit Mallinar, Daniel Beaglehole, Libin Zhu, Adityanarayanan Radhakrishnan, Parthe Pandit, and Mikhail Belkin. Emergence in non-neural models: grokking modular arithmetic via average gradient outer product. In *Forty-second International Conference on Machine Learning*, 2025.

[39] Gavin McCracken, Gabriela Moisescu-Pareja, Vincent Letourneau, Doina Precup, and Jonathan Love. Uncovering a universal abstract algorithm for modular addition in neural networks. *arXiv preprint arXiv:2505.18266*, 2025.

[40] Sheng Liu, Zhihui Zhu, Qing Qu, and Chong You. Robust training under label noise by over-parameterization. In *International Conference on Machine Learning*, pages 14153–14172. PMLR, 2022.

[41] Hwanjun Song, Minseok Kim, Dongmin Park, Yooju Shin, and Jae-Gil Lee. Learning from noisy labels with deep neural networks: A survey. *IEEE transactions on neural networks and learning systems*, 34(11):8135–8153, 2023.

[42] Zhiwei Xu, Yutong Wang, Spencer Frei, Gal Vardi, and Wei Hu. Benign overfitting and grokking in relu networks for xor cluster data. *arXiv preprint arXiv:2310.02541*, 2023.

A Related Work

Model-wise Double Descent and Over-parameterization. The discovery of the double descent phenomenon (illustrated in Figure 1a) marked a significant shift in modern machine learning, suggesting that the classical bias–variance trade-off is incomplete [22, 23]. In the over-parameterized regime, test error can decrease again after the interpolation threshold, leading to superior generalization. Theoretically, this second descent has been rigorously characterized in linear regression under various assumptions [24, 25, 26, 27, 28, 29, 30], although some studies indicate that appropriate regularization can mitigate or eliminate this effect [31]. In non-linear settings, double descent has been robustly observed in computer vision tasks, particularly in the presence of label noise [16, 32].

Recent efforts have sought mechanistic explanations for the generalization power of large-scale models. One perspective proposes that over-parameterized models generalize through smoother interpolation of noisy data points [33, 34]. Furthermore, the emergence of neural scaling laws has been increasingly attributed to the superposition of discrete features or representations [6]. Davies et al. [35], Huang et al. [36] further unify double descent with grokking by framing both phenomena as a competition between memorization and generalization circuits. Our work extends this line of research by investigating how model size specifically governs the resolution of this competition under label noise.

Grokking and Emergent Generalization. In over-parameterized models, generalization often emerges long after the training loss has vanished, a phenomenon known as grokking [8]. This decoupling between fitting and rule discovery is prominently observed in modular arithmetic tasks across various architectures [37, 9, 38]. While specific algorithms, such as Pizza and Clock, have been identified for Transformers solving these tasks [9, 10], and a universal abstract algorithm for deep neural networks has been uncovered [39], the exact analytical solution for two-layer ReLU networks remains unsolved. Existing theoretical works have explored provable solutions using quadratic activations or specific regularization schemes [19, 12], yet the role and impact of label noise in these settings remain largely unexplored.

Learning under Label Noise. Most existing work on learning with label noise focuses on robust training strategies to prevent overfitting [18, 40]. We refer readers to Song et al. [41] for a comprehensive survey. Xu et al. [42] theoretically analyze the performance of two-layer ReLU networks on noisy XOR data under the assumption of fixed second-layer weights, which deviates from standard end-to-end training dynamics. Closest to our work, Doshi et al. [13] investigate memorization–generalization coexistence on corrupted algorithmic tasks, demonstrating that pruning neurons with low Inverse Participation Ratio (IPR) can reduce memorization. Building upon their findings, our work advances this direction in three key aspects: (i) we systematically study how test accuracy varies with model width across noise levels and regularization strengths, and how width affects recovery of the latent rule under heavy noise; (ii) we introduce *frequency filtration (FF)* to precisely extract the internal generalizable structure at the representation level, achieving near-perfect rule recovery even when raw outputs are corrupted; and (iii) we propose a task-agnostic *neuron strength (Str.)* metric to evaluate neuron-level partitioning beyond Fourier priors, while revealing the fundamental limitations of neuron pruning compared with frequency-level extraction.

B Analytical Solution Under Quadratic Activation

The fact that the solution in Equation (2) is exact for quadratic activations has been well discussed in Pearce et al. [7], Morwani et al. [11]. We present their theory here, adapted to our notation, in Proposition B.1.

Proposition B.1. *(Theorem 5, Morwani et al. [11]) For the modular addition task, any two-layer network* $f_\theta = \boldsymbol{W}\left(\boldsymbol{U}\boldsymbol{e}_a + \boldsymbol{V}\boldsymbol{e}_b\right)^2$ *with* $M \geq 4(P-1)$ *that maximizes the* $L_{2,3}$*-normalized margin on* $\mathcal{D}_{total}$ *satisfies the following conditions:*

1. for each neuron $\{\boldsymbol{u}_m, \boldsymbol{v}_m, \boldsymbol{w}_m\}$*, there exists a scaling constant* $\lambda \in \mathbb{R}$ *and a frequency* $\omega_m \in \{1, \cdots, \frac{P-1}{2}\}$*, such that*

$$u_{mi} = \lambda \cos\left(\frac{2\pi}{P}\omega_m i + \varphi_m^{(a)}\right), \tag{6a}$$

$$v_{mj} = \lambda \cos\left(\frac{2\pi}{P}\omega_m j + \varphi_m^{(b)}\right), \tag{6b}$$

$$w_{mk} = \lambda \cos\left(\frac{2\pi}{P}\omega_m k + \varphi_m^{(c)}\right), \tag{6c}$$

for some phase offsets $\varphi_m^{(a)}, \varphi_m^{(b)}, \varphi_m^{(c)} \in \mathbb{R}$ *satisfying* $\varphi_m^{(a)} + \varphi_m^{(b)} = \varphi_m^{(c)}$.

2. For every frequency $\omega \in \{1, \cdots, \frac{P-1}{2}\}$*, at least one neuron in the network uses this frequency.*

C Supplementary Experiments for Section 3

C.1 Double-descent curves on varying setups

Model-wise double descent is clearly observed across different activation functions, training ratios, optimizers, and arithmetic tasks. Table 1 lists the corresponding figures.

Table 1: Figure indices showing double descent curves under varying setups.

Activation functions	Training ratios	Optimizers	Arithmetic tasks
1b, 8	9 (40%), 2 (50%), 10 (60%)	2 (AdamW), 11 (Adam), 12 (Muon)	13

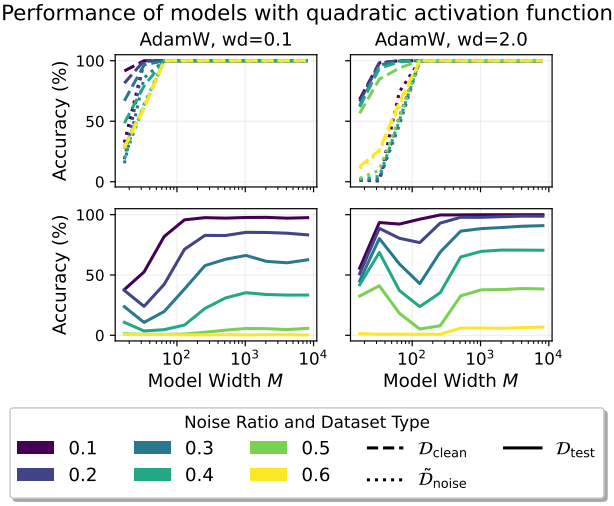


Figure 8: Ceilings of test accuracy for quadratic activation functions exist across varying weight decays.

SGD is not a good choice. Figure 14 shows the training dynamics of SGD under different choices of learning rate and weight decay. Among all configurations, only the setting with lr $= 0.1$ and wd $= 0.001$ exhibits noticeable generalization behavior. This suggests that SGD requires highly restrictive hyperparameter tuning to achieve reasonable generalization. Moreover, even at epoch 200,000, the test accuracy remains relatively low and is still increasing, indicating that SGD converges extremely slowly. Overall, SGD is inefficient for this task, as it consumes an excessive number of training epochs to reach a modest level of generalization.

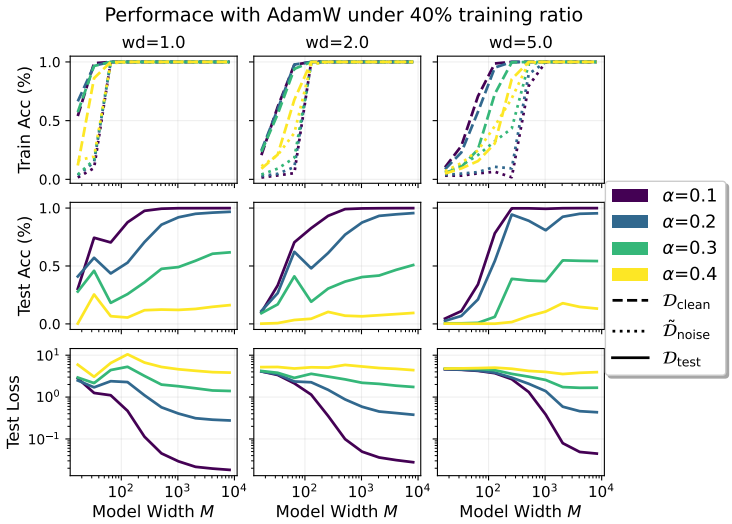


Figure 9: Performance of AdamW with a training ratio of 40%.

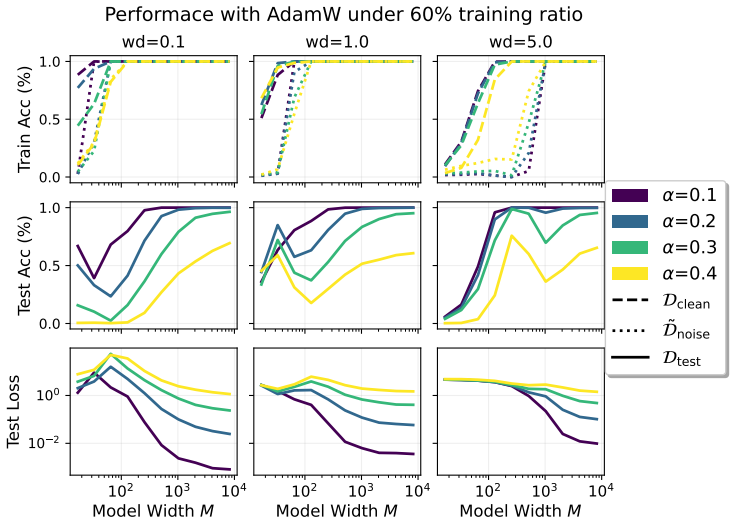


Figure 10: Performance of AdamW with a training ratio of 60%.

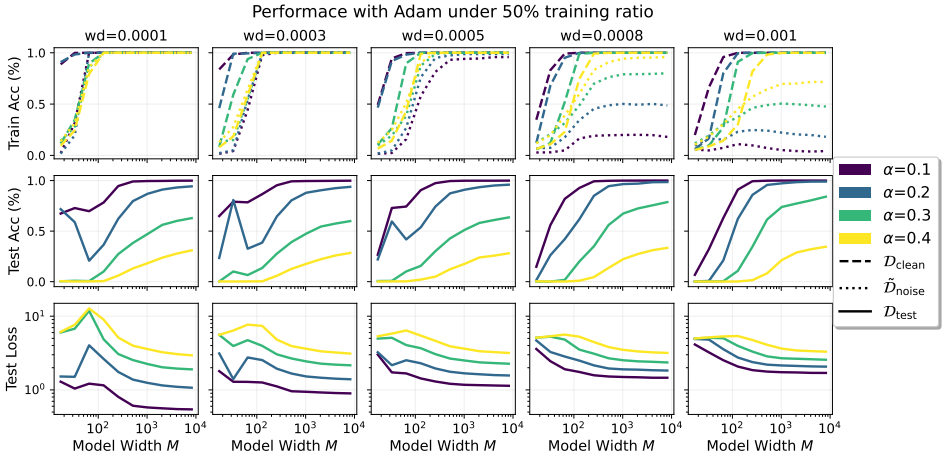


Figure 11: Performance of Adam under varying weight decays.

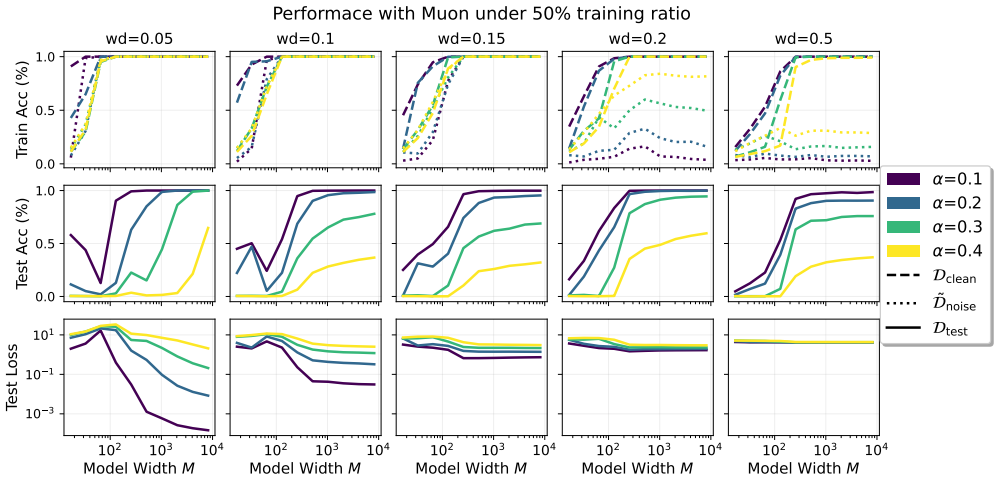


Figure 12: Performance of Muon under varying weight decays.

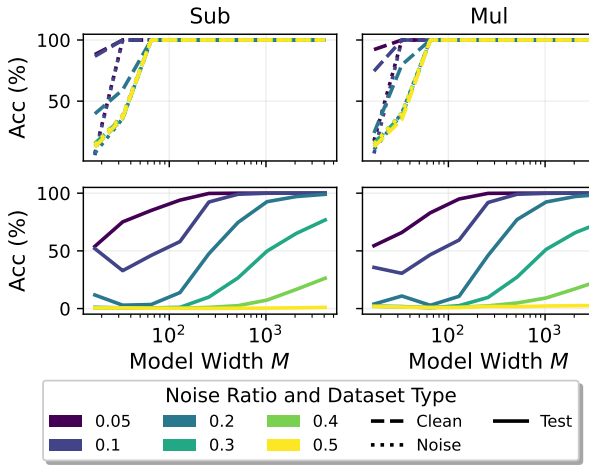


Figure 13: Performance on modular subtraction (*left*) and multiplication (*right*) tasks. Models are trained using AdamW under 50% training ratio.

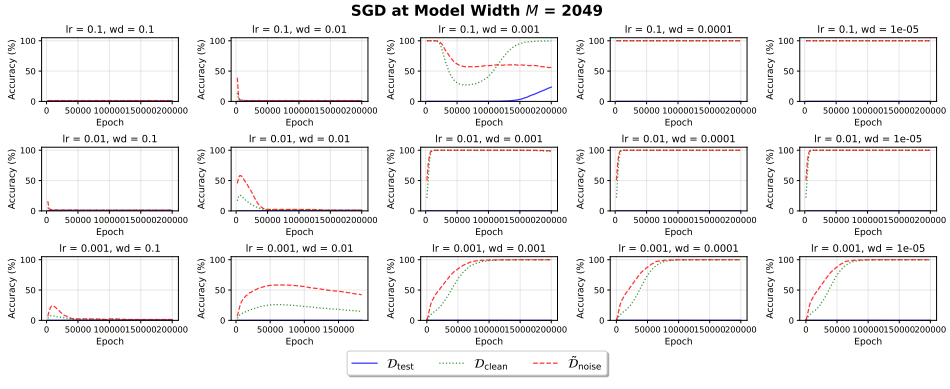


Figure 14: Training dynamics of SGD under different learning rates (lr) and weight decay (wd) values at $M = 2049$ and $\alpha = 0.3$. Results are reported every 2,000 epochs, over a total of 200,000 epochs.

C.2 Sensitivity to random seeds

Width scaling. Table 2 reports test accuracy before FF or neuron pruning across seeds 0, 123, and 42. The standard deviation is below one percentage point in every setting. By comparison, increasing width from 2049 to 16385 improves mean test accuracy by 4.92 and 6.38 percentage points at $\alpha = 0.2$ and $\alpha = 0.4$, respectively. At $\alpha = 0.1$, accuracy stays near 100%. These results support the overall width trend across the tested seeds.

Table 2: Test accuracy across three random seeds at 50% training ratio, weight decay 1.0. Values are mean (standard deviation), in percent, over seeds 0, 123, and 42. All other settings follow the default configuration in Section 2.

Model width M	$\alpha = 0.1$	$\alpha = 0.2$	$\alpha = 0.4$
2049	99.94 (0.09)	92.39 (0.54)	4.36 (0.28)
4097	99.96 (0.04)	95.32 (0.80)	6.49 (0.12)
8193	99.99 (0.02)	96.43 (0.60)	8.58 (0.64)
16385	99.98 (0.02)	97.31 (0.54)	10.74 (0.47)

Non-monotonicity with less training data. The scaling trend is not always monotonic. Table 3 revisits the setting in Figure 9, with a training ratio of 40% and weight decay 5.0. At $\alpha = 0.4$, mean test accuracy peaks at $M = 4097$ and decreases at larger widths. The decline is larger than the reported standard deviations, so it is unlikely to be explained by seed variation alone. In contrast, accuracy increases with width at $\alpha = 0.2$ and remains near 100% at $\alpha = 0.1$. This exception shows that the benefit of width depends on the amount of training data, the noise level, and regularization.

Table 3: Test accuracy with a training ratio of 40% and weight decay 5.0. Values are mean (standard deviation), in percent, over seeds 0, 123, and 42. All other settings follow the default configuration in Section 2.

Model width M	$\alpha = 0.1$	$\alpha = 0.2$	$\alpha = 0.4$
2049	99.93 (0.02)	91.86 (0.58)	12.19 (0.17)
4097	99.98 (0.03)	95.63 (0.65)	15.05 (0.92)
8193	99.95 (0.03)	96.05 (0.77)	13.41 (0.03)
16385	99.94 (0.07)	96.17 (0.47)	11.21 (0.31)

Early fitting of noisy labels. We also repeat the early-training experiments with seeds 0 and 123. Figure 15 shows that noisy-label accuracy rises earlier than clean-label accuracy in both runs. The gap is small and disappears as both accuracies approach 100% in wider models. These results reproduce the early fitting order observed with seed 42.

C.3 Discussion on Model Misspecification

We consider a scenario where the model architecture is not expressive enough to memorize all noisy labels, which leads to atypical generalization behavior.

Modular addition and multiplication are commutative operations, i.e., $a \circ b = b \circ a$. Accordingly, an alternative architecture enforces a tied first layer by setting $U = V$ [7]. Under random label noise, however, the noisy samples generally violate this symmetry. As a result, the tied model is unable to interpolate all noisy labels, and the coexistence phase does not emerge in this setting, leading to irregular test accuracy curves (see the left panel in Figure 16).

To verify that the asymmetric structure of random noise is responsible for this behavior, we construct a symmetric noise pattern and re-evaluate the tied model. Specifically, for a noise ratio α in modular addition, we randomly sample $\frac{\alpha}{2}|\mathcal{D}_{\text{train}}|$ examples to form $\mathcal{D}_{\text{asym}}$, where each $(a, b, c) \in \mathcal{D}_{\text{asym}}$ satisfies $a < b$. The noisy dataset is then constructed as

$$\tilde{\mathcal{D}}_{\text{noise}} = \{(a, b, \tilde{c}) \mid (a, b, c) \in \mathcal{D}_{\text{asym}} \text{ or } (b, a, c) \in \mathcal{D}_{\text{asym}},\ \tilde{c} \neq c,\ \tilde{c} \in \{0, \ldots, P-1\}\}.$$

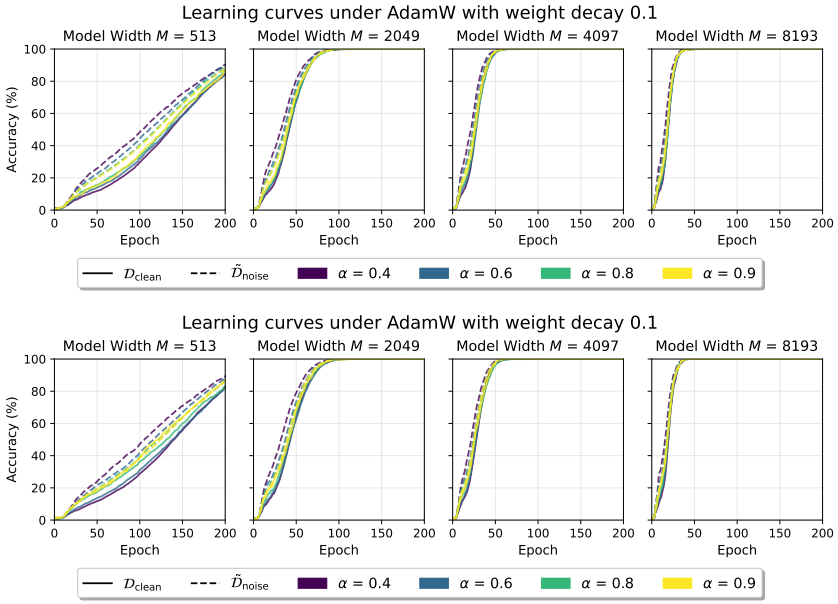


Figure 15: Early training dynamics with seed 0 (top) and seed 123 (bottom). Models use ReLU activations and full-batch AdamW with weight decay 0.1. All other settings follow Section 2.

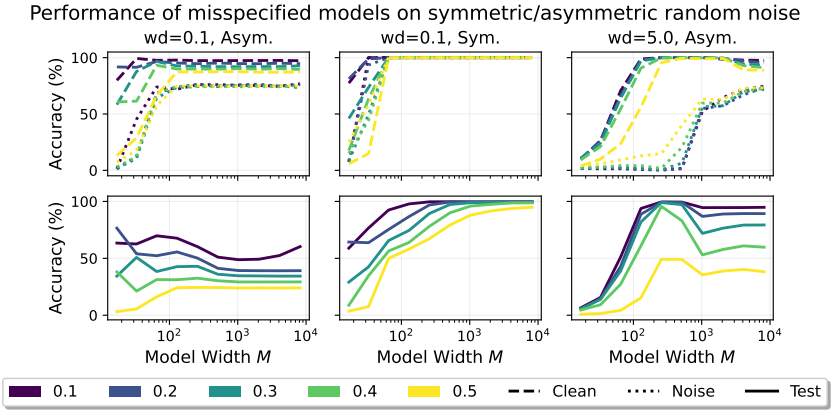


Figure 16: Performance of first-layer tied models under model misspecification. *(Left)* With asymmetric random noise, the tied model fails to fully interpolate noisy labels and exhibits irregular generalization behavior. *(Middle)* Under symmetric noise, the tied model successfully interpolates all noisy labels, and generalization improves with increasing model size. *(Right)* Stronger regularization mitigates the adverse effect of misspecification by encouraging the model to ignore noisy labels.

Under this symmetric noise, the tied model exhibits predictable behavior: noisy labels can be fully memorized once the model width exceeds a threshold (i.e., the coexistence phase appears), and generalization improves with increasing model size (see the middle panel in Figure 16).

When the noise structure is unknown, model misspecification becomes a potential risk. One practical mitigation strategy is to strengthen regularization so as to extend the inversion phase. The tied model is able to interpolate all clean samples, and increasing weight decay encourages the model to suppress/neglect noisy labels, thereby recovering good generalization performance (see the right panel in Figure 16).

C.4 Discussion on Result 3.2

We provide a gradient-based explanation for the counter-intuitive observation in Result 3.2. Let $\mathcal{D}_{\mathrm{c}} = \mathcal{D}_{\text{clean}}$ and $\mathcal{D}_{\mathrm{n}} = \tilde{\mathcal{D}}_{\text{noise}}$ denote the clean and noisy subsets of the training data, respectively. For a model parameterized by θ, we decompose the empirical loss as

$$L(\theta) = L_{\mathrm{c}}(\theta) + L_{\mathrm{n}}(\theta), \qquad L_{\mathrm{c}}(\theta) = \sum_{z \in \mathcal{D}_{\mathrm{c}}} \ell(z;\theta), \qquad L_{\mathrm{n}}(\theta) = \sum_{z \in \mathcal{D}_{\mathrm{n}}} \ell(z;\theta),$$

where ℓ is the cross-entropy loss on a single example. Under full-batch gradient descent, the update direction is

$$-\nabla L(\theta) = -\nabla L_{\mathrm{c}}(\theta) - \nabla L_{\mathrm{n}}(\theta).$$

For a sufficiently small step size η, the first-order change of the clean and noisy losses after one update is approximately

$$\Delta L_{\mathrm{c}} \approx -\eta \left\langle \nabla L_{\mathrm{c}}, \nabla L_{\mathrm{c}} + \nabla L_{\mathrm{n}} \right\rangle, \qquad \Delta L_{\mathrm{n}} \approx -\eta \left\langle \nabla L_{\mathrm{n}}, \nabla L_{\mathrm{c}} + \nabla L_{\mathrm{n}} \right\rangle.$$

Thus, the instantaneous learning speed of each subset is controlled by how well its own gradient aligns with the full-batch training direction. To compare the two subsets fairly, we normalize by their sizes and define the per-sample loss decrease rates

$$s_{\mathrm{c}} = \frac{\langle \nabla L_{\mathrm{c}}, \nabla L_{\mathrm{c}} + \nabla L_{\mathrm{n}} \rangle}{|\mathcal{D}_{\mathrm{c}}|}, \qquad s_{\mathrm{n}} = \frac{\langle \nabla L_{\mathrm{n}}, \nabla L_{\mathrm{c}} + \nabla L_{\mathrm{n}} \rangle}{|\mathcal{D}_{\mathrm{n}}|}.$$

Empirically, at the beginning of training, we observe that $\langle \nabla L_{\mathrm{c}}, \nabla L_{\mathrm{n}} \rangle < 0$, meaning that the clean and noisy subsets induce partially conflicting gradient directions. Nevertheless, both normalized progress measures remain positive, i.e., $s_{\mathrm{c}} > 0$ and $s_{\mathrm{n}} > 0$. This indicates that the full-batch update still decreases both the clean and noisy losses at the very beginning of training, although it does so at different per-sample rates. However, since the noisy subset is much smaller when the noise ratio α is small, the update direction can produce a larger *per-sample* decrease on the noisy subset. In other words, although the noisy labels are unstructured, each noisy example receives a stronger individual push from the full-batch gradient. This explains why the noisy-label accuracy can increase earlier than the clean-label accuracy, especially at lower noise ratios.

This gradient-based explanation is also consistent with additional experiments beyond modular arithmetic. To test whether the early noisy-over-clean learning order is purely induced by the algebraic structure of modular arithmetic, we further train a three-layer neural network on MNIST with randomly corrupted labels, using hidden-layer widths of 1,024 and 256 and AdamW with a learning rate of 10^{-3}. As shown in Table 4, under full-batch training, noisy-label accuracy is higher than clean-label accuracy at epoch 1 for both $\alpha = 0.05$ and $\alpha = 0.2$. In the next epoch, clean accuracy quickly catches up and becomes much larger, while noisy accuracy drops. Over the remaining observed epochs, noisy-label accuracy does not increase, while clean-label accuracy does not decrease. We do not observe the early noisy-over-clean ordering in the minibatch setting we tested. These results suggest that the early advantage of noisy labels is not solely a consequence of the Fourier structure of modular arithmetic, but can also arise in a standard image classification task under a similar optimization protocol.

The MNIST experiment also helps clarify the scope of the phenomenon. The effect is most visible at the very beginning of full-batch training and should not be interpreted as saying that noisy labels are ultimately easier to learn than clean labels. Instead, the first few updates may create a larger per-sample progress on the smaller noisy subset, while the clean structure dominates soon afterward. This is consistent with our gradient analysis: the initial quantities s_{c} and s_{n} can both be positive, with $s_{\mathrm{n}} > s_{\mathrm{c}}$, even though the clean and noisy gradients partially conflict.

Table 4: Early-stage clean and noisy accuracy on MNIST with a three-layer neural network under full-batch training.

	$\alpha = 0.05$		$\alpha = 0.2$	
Period	Noisy acc.	Clean acc.	Noisy acc.	Clean acc.
Epoch 1	10.87%	2.86%	10.75%	2.89%
Epoch 2	6.47%	46.29%	5.71%	48.09%

This explanation also clarifies why the phenomenon depends on the training protocol. Under full-batch training, the gradient contribution of the noisy subset is accumulated coherently in every update, so the direction that benefits the current noisy labels is repeatedly applied. In contrast, under small-batch training, different mini-batches contain different noisy examples, and the gradient direction that helps memorize the noisy labels in one batch is generally not transferable to another batch. As a result, the per-sample push toward any specific corrupted label becomes less stable, and the early noisy-over-clean learning order is weakened or may disappear.

C.5 Accuracy learning curves

The accuracy learning curves capture when memorization and generalization emerge during training. We observe the grokking phenomenon even under noisy labels: the model quickly reaches 100% training accuracy, while test accuracy improves significantly only later. To illustrate this, we separately present the training dynamics on short and long time scales, highlighting memorization and generalization, respectively.

C.5.1 Memorization on a short time scale

Without noise in the training dataset, previous studies have found that larger models memorize the training data faster [17, 36]. This trend persists under noisy training data, although memorization speeds differ for clean and noisy subsets. We find that the model memorizes noisy labels faster than clean labels, and that the gap is larger at lower noise ratios. These phenomena are consistently observed across several optimizers, including Adam (Figure 17), AdamW (Figure 18), and Muon (Figure 19).

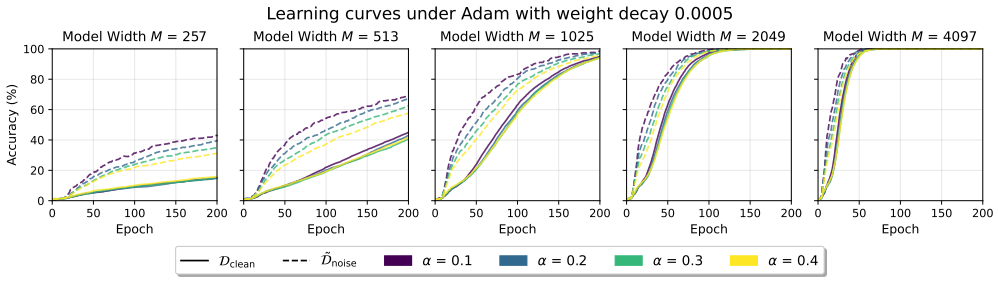


Figure 17: Memorization curves for the Adam optimizer. Larger models reach high training accuracy more quickly. Noisy labels are memorized faster than clean labels, and lower noise ratios further accelerate this effect.

C.5.2 Generalization on a long time scale

In terms of test accuracy, we observe a general tendency that models under larger weight decay grok faster, although the trend is not equally pronounced across all hyperparameter settings. This trend is clear for AdamW (Figure 20) and Muon (Figure 22), but not evident for Adam (Figure 21). Across all of them, models with lower noise ratios grok faster.

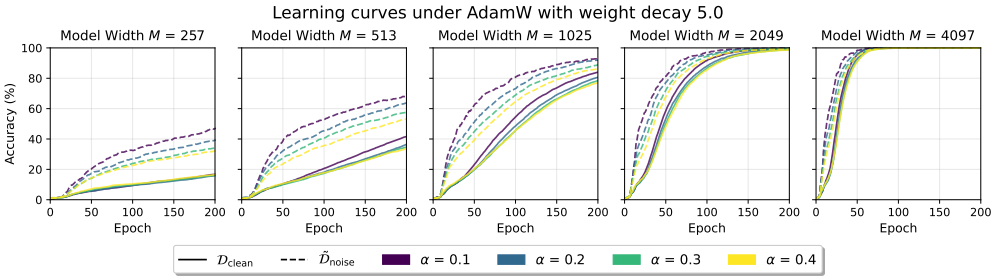


Figure 18: Memorization curves for the AdamW optimizer.

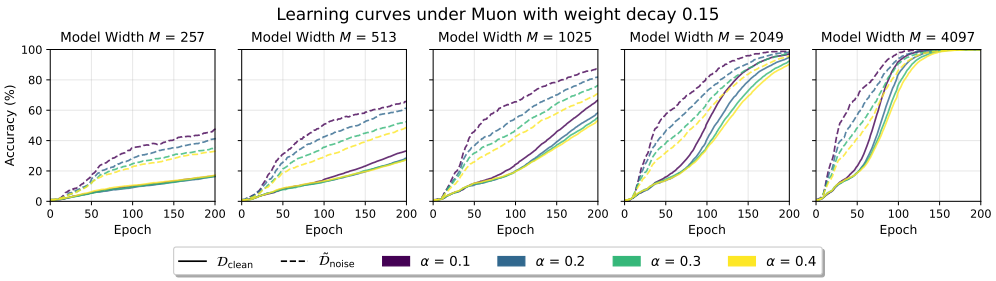


Figure 19: Memorization curves for the Muon optimizer.

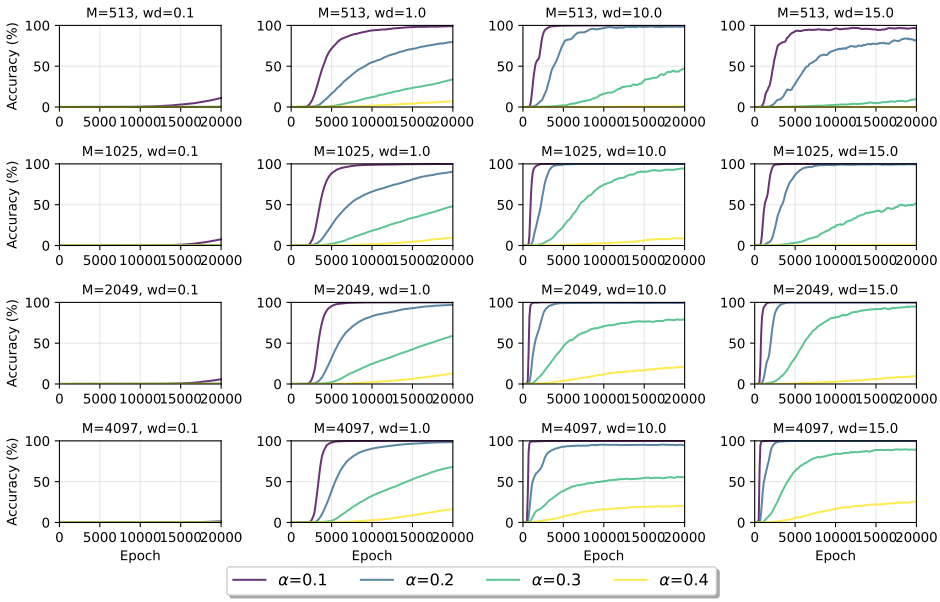


Figure 20: Training dynamics of test accuracy across various weight decays for AdamW. Small weight decay (wd=0.1) leads to delayed and slow grokking.

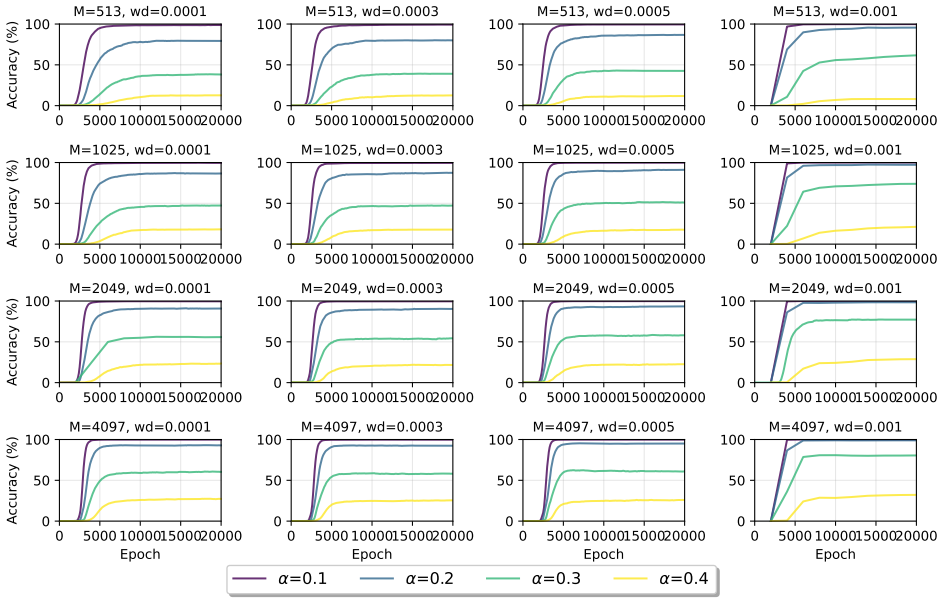


Figure 21: Training dynamics of test accuracy across various weight decays for Adam.

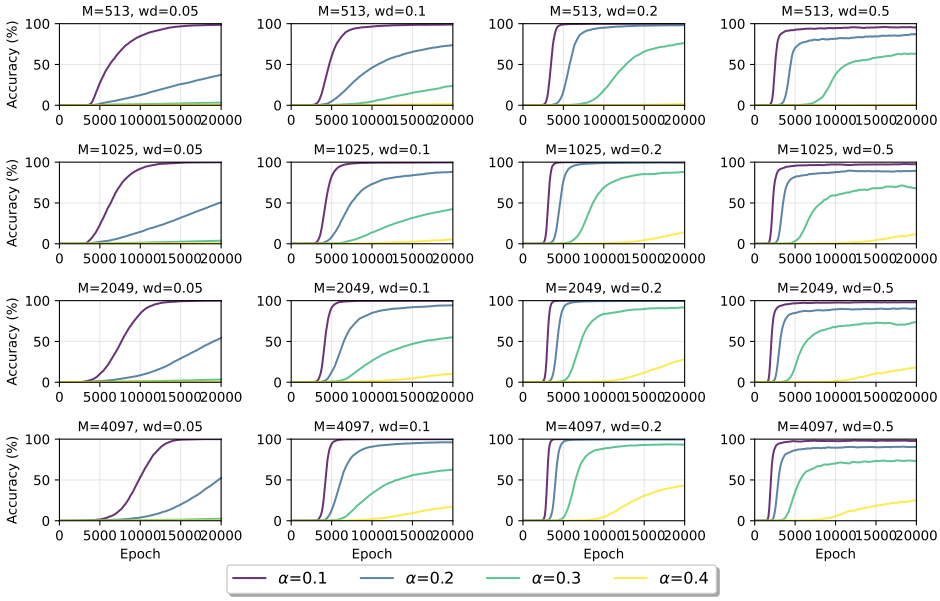


Figure 22: Training dynamics of test accuracy across various weight decays for Muon.

D Supplementary Materials for Section 4

D.1 Details of frequency filtration

In this section, we provide more details about decomposing $(\boldsymbol{u}_m, \boldsymbol{v}_m, \boldsymbol{w}_m)$, the m^{th} neuron, into $\left(\boldsymbol{u}_m^G, \boldsymbol{v}_m^G, \boldsymbol{w}_m^G\right)$ and $\left(\boldsymbol{u}_m^R, \boldsymbol{v}_m^R, \boldsymbol{w}_m^R\right)$.

Let $\omega = e^{-2\pi i/P}$. Define the Fourier transform matrix $F \in \mathbb{C}^{P\times P}$ by

$$F_{k,n} = \frac{1}{\sqrt{P}}\omega^{kn} = \frac{1}{\sqrt{P}}e^{-2\pi ikn/P}.$$

For any vector $\boldsymbol{x} \in \mathbb{R}^{P\times 1}$, denote its Fourier transform as $\tilde{\boldsymbol{x}} = \boldsymbol{F}\boldsymbol{x}$. Since $\boldsymbol{F}$ is unitary, its conjugate transpose $\boldsymbol{F}^H$ satisfies $\boldsymbol{F}^H\boldsymbol{F}\boldsymbol{x} = \boldsymbol{x}$. For real-valued weights, Fourier coefficients occur in conjugate pairs. When multiple indices attain the maximum magnitude, we select the smallest index.

We define the main frequency retrieval operator (for the generalization component) as $\boldsymbol{F}^H\boldsymbol{F}_G(\boldsymbol{x})$, filtering the frequency with maximum magnitude, namely,

$$F_G(\boldsymbol{x})_{k,n} = \begin{cases} F_{k,n}, & k = \min\arg\max_{k'}|\tilde{x}_{k'}| \\ 0, & k \neq \min\arg\max_{k'}|\tilde{x}_{k'}| \end{cases},$$

and the remaining frequency retrieval operator (for the residual weights) as $\boldsymbol{F}^H\boldsymbol{F}_R(\boldsymbol{x})$, where $\boldsymbol{F}_R(\boldsymbol{x}) = \boldsymbol{F} - \boldsymbol{F}_G(\boldsymbol{x})$.

For the m^{th} neuron, let $\boldsymbol{P}_{G,m} = \boldsymbol{F}^H\boldsymbol{F}_G(\boldsymbol{w}_m)$ and $\boldsymbol{P}_{R,m} = \boldsymbol{F}^H\boldsymbol{F}_R(\boldsymbol{w}_m)$. We then decompose the weights of neuron m as follows.

$$\begin{aligned} \boldsymbol{u}_m^G &= \text{Real}(\boldsymbol{P}_{G,m}\boldsymbol{u}_m),\ \boldsymbol{u}_m^R = \text{Real}(\boldsymbol{P}_{R,m}\boldsymbol{u}_m), \\ \boldsymbol{v}_m^G &= \text{Real}(\boldsymbol{P}_{G,m}\boldsymbol{v}_m),\ \boldsymbol{v}_m^R = \text{Real}(\boldsymbol{P}_{R,m}\boldsymbol{v}_m), \\ \boldsymbol{w}_m^G &= \text{Real}(\boldsymbol{P}_{G,m}\boldsymbol{w}_m),\ \boldsymbol{w}_m^R = \text{Real}(\boldsymbol{P}_{R,m}\boldsymbol{w}_m). \end{aligned}$$

It is easy to check $\boldsymbol{P}_{G,m} + \boldsymbol{P}_{R,m} = \boldsymbol{I}$, so

$$(\boldsymbol{u}_m, \boldsymbol{v}_m, \boldsymbol{w}_m) = \left(\boldsymbol{u}_m^G, \boldsymbol{v}_m^G, \boldsymbol{w}_m^G\right) + \left(\boldsymbol{u}_m^R, \boldsymbol{v}_m^R, \boldsymbol{w}_m^R\right).$$

We provide the visualization of a neuron under such frequency filtration in Figure 23.

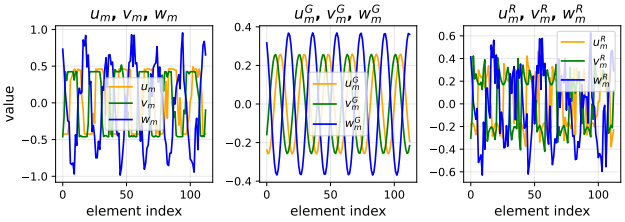


Figure 23: Visualization of applying frequency filtration to a neuron. The frequency selected from $\boldsymbol{w}_m$ is used to filter all three weight vectors.

D.2 Frequency filtration benefits generalization under varying setups

Frequency filtration also disentangles generalization and memorization for the modular *subtraction* task (Figure 24) and for models with *quadratic* activations (Figure 25). We observe substantial improvement in generalization after frequency filtration: ReLU models recover test accuracy under severe label noise, while quadratic models exceed previously observed accuracy limits. Test accuracy also increases for models trained on noise-free data (Figure 26).

Frequency filtration on Transformers. We also test FF on a 1-layer decoder-only Transformer trained on noisy modular addition. Prior work has studied its mechanism on clean data [9]. Our model uses hidden dimension $d = 2048$, four attention heads, ReLU activations, and no LayerNorm. We set the noise ratio to 20% and train for 200,000 epochs, using AdamW with betas $(0.9, 0.98)$,

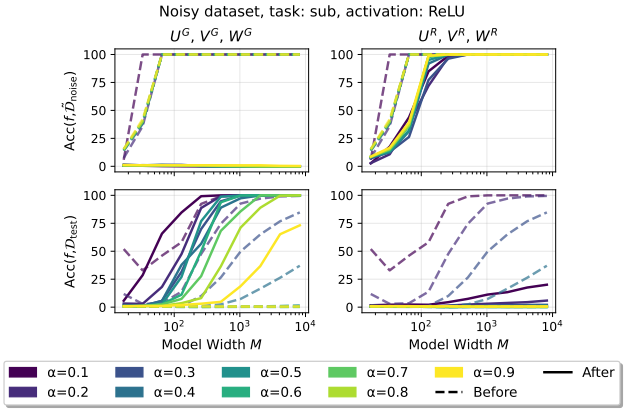


Figure 24: Performance before (dashed) and after (solid) the frequency filtration on the modular *subtraction* task.

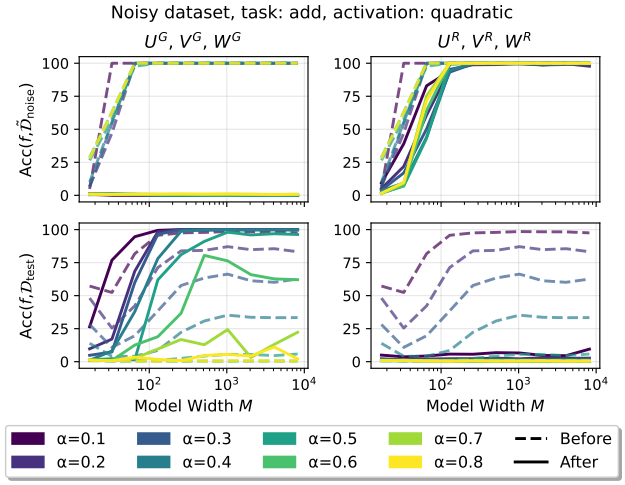


Figure 25: Performance before (dashed) and after (solid) the frequency filtration on models with *quadratic* activation. We remark that frequency filtration also breaks the ceilings of test accuracy.

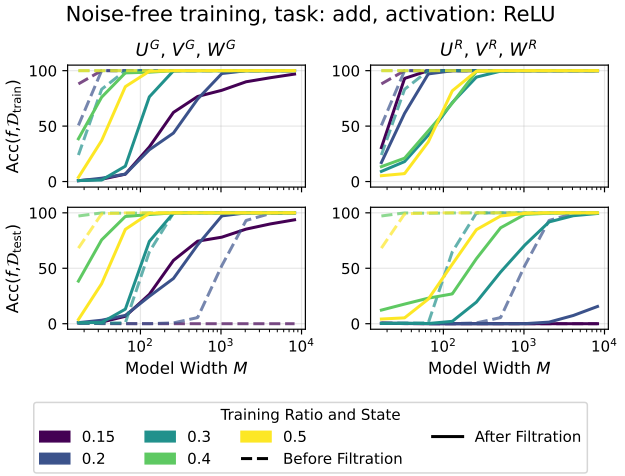


Figure 26: Performance before (dashed) and after (solid) frequency filtration for models trained on the clean modular addition data. The models use ReLU activation and are trained with AdamW. Overall, test accuracy after filtration (U^G, V^G, W^G) is generally higher, particularly for large models.

Table 5: Frequency filtration on a Transformer trained with 20% label noise under different weight decay values. FF retains the top 20 frequencies in the unembedding head. All other layers remain unchanged. Values are accuracies in percent.

Weight decay	Method	Clean Acc.	Noisy Acc.	Test Acc.
1.0	Before FF	100.00	100.00	74.14
	After FF	97.45	37.23	80.40
2.0	Before FF	100.00	100.00	67.51
	After FF	97.73	42.71	81.05
5.0	Before FF	100.00	100.00	78.73
	After FF	97.98	39.03	86.93

learning rate 0.001, full batch, and linear warmup over the first 10 steps. After training, we retain only the top 20 frequencies in the unembedding head. All other layers remain unchanged. Table 5 shows that this intervention improves test accuracy and reduces accuracy on corrupted labels. This result provides preliminary evidence that frequency-based recovery also works beyond two-layer networks.

D.3 Frequency and phase alignment

Frequency alignment. We measure how often the dominant frequencies of $\boldsymbol{u}_m$, $\boldsymbol{v}_m$, and $\boldsymbol{w}_m$ agree before FF. Table 6 reports both the fraction and the number of aligned neurons with weight decay 0.1 and seed 42. At each noise ratio, the fraction decreases with width, but the absolute number increases. Thus, wider models contain more frequency-aligned neurons even though these neurons form a smaller fraction of the network.

Table 6: Agreement among the dominant frequencies of $\boldsymbol{u}_m$, $\boldsymbol{v}_m$, and $\boldsymbol{w}_m$ before FF. Each entry reports the percentage of aligned neurons, followed by their count over the model width.

M	$\alpha = 0.1$	$\alpha = 0.2$	$\alpha = 0.4$	$\alpha = 0.8$
513	76.61% (393/513)	67.84% (348/513)	42.88% (220/513)	0.97% (5/513)
2049	63.40% (1299/2049)	54.27% (1112/2049)	36.12% (740/2049)	0.59% (12/2049)
4097	57.92% (2373/4097)	47.82% (1959/4097)	35.10% (1438/4097)	0.56% (23/4097)

Phase alignment. Suppose the dominant frequency of neuron m has the form

$$u_{mi}^G = \lambda_u \cos\left(\frac{2\pi}{P}\omega_m i + \varphi_{m,G}^{(a)}\right), \tag{7a}$$

$$v_{mj}^G = \lambda_v \cos\left(\frac{2\pi}{P}\omega_m j + \varphi_{m,G}^{(b)}\right), \tag{7b}$$

$$w_{mk}^G = \lambda_w \cos\left(\frac{2\pi}{P}\omega_m k + \varphi_{m,G}^{(c)}\right). \tag{7c}$$

For models trained with either ReLU or quadratic activations, there exists $q \in \mathbb{Z}$ such that the following equations are approximately satisfied:

$$\varphi_{m,G}^{(a)} + \varphi_{m,G}^{(b)} = \varphi_{m,G}^{(c)} + 2q\pi \qquad \text{(modular addition)}, \tag{8}$$

$$\varphi_{m,G}^{(a)} - \varphi_{m,G}^{(b)} = \varphi_{m,G}^{(c)} + 2q\pi \qquad \text{(modular subtraction)}. \tag{9}$$

We quantify the empirical deviation of these relationships using the mean squared error (MSE). Specifically, for modular addition:

$$\text{Phase MSE} = \frac{1}{M}\sum_{m=1}^{M} \min_{q\in\mathbb{Z}} \left(\varphi_{m,G}^{(a)} + \varphi_{m,G}^{(b)} - \varphi_{m,G}^{(c)} + 2q\pi\right)^2,$$

and for modular subtraction, replace $\varphi^{(a)}_{m,G} + \varphi^{(b)}_{m,G}$ with $\varphi^{(a)}_{m,G}\varphi^{(b)}_{m,G}$.

We visualize these relationships in Figures 27 (addition) and 28 (subtraction), and the Phase MSE trends with respect to model size and noise ratio are shown in Figure 29. We observe that higher noise ratios generally lead to larger Phase MSE, i.e., larger deviation from the analytical relationship.

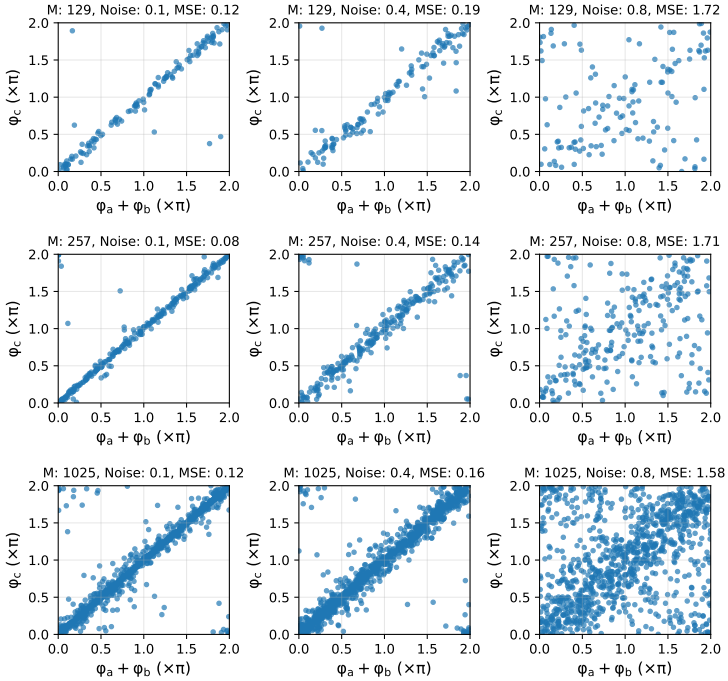


Figure 27: Scatter plot showing the relationship of phases on modular addition tasks. Each point represents a neuron, where φ_a, φ_b, and φ_c are the phases of $\boldsymbol{u}^G$, $\boldsymbol{v}^G$, and $\boldsymbol{w}^G$, respectively.

D.4 Uniform distribution of the frequency

Figure 30 shows that each frequency is covered by at least one neuron after frequency filtration (only dominant frequency left for each neuron). Figure 31 also shows that the distribution of the frequency magnitude after filtration is close to uniform at larger model width. Here, let $\tilde{\boldsymbol{W}}^G$ denote the Fourier transform of $\boldsymbol{W}^G$ with each column constituted by $\{\boldsymbol{w}_m\}_{m=1}^M$, so Figure 31 plots the norm of each row vector in $\tilde{\boldsymbol{W}}^G$ with index in $\left\{1, \cdots, \frac{P-1}{2} + 1\right\}$.

D.5 Ablations of frequency filtration

We use conjugate-pair and random-frequency filtering to test how the choice of retained frequencies affects FF. We also apply FF to randomly initialized models to test whether filtering alone can produce high test accuracy. All experiments use modular addition with $P = 113$ and widths $M \in \{513, 2049, 8193\}$. The trained-model experiments use noise ratios $\alpha \in \{0.4, 0.6, 0.8\}$ and the default configuration in Section 2, with model and data seed 42. All procedures retain the output bias.

Conjugate-pair filtering. For each neuron m, we select the smallest index k_m among those with the largest magnitude in $\tilde{\boldsymbol{w}}_m$. We retain both k_m and its conjugate frequency $(-k_m) \bmod P$ in the Fourier transforms of $\boldsymbol{u}_m$, $\boldsymbol{v}_m$, and $\boldsymbol{w}_m$. All other coefficients are set to zero. The residual retains all coefficients outside this pair. We apply the inverse Fourier transform and take the real part to obtain the filtered weights. If $k_m = 0$, only the DC component is selected. This procedure is deterministic for a fixed checkpoint.

Figure 32 shows that conjugate-pair filtering achieves test accuracy comparable to the original single-frequency FF. At $\alpha = 0.8$, test accuracy reaches 98.62% and 100.00% for widths 2049 and 8193,

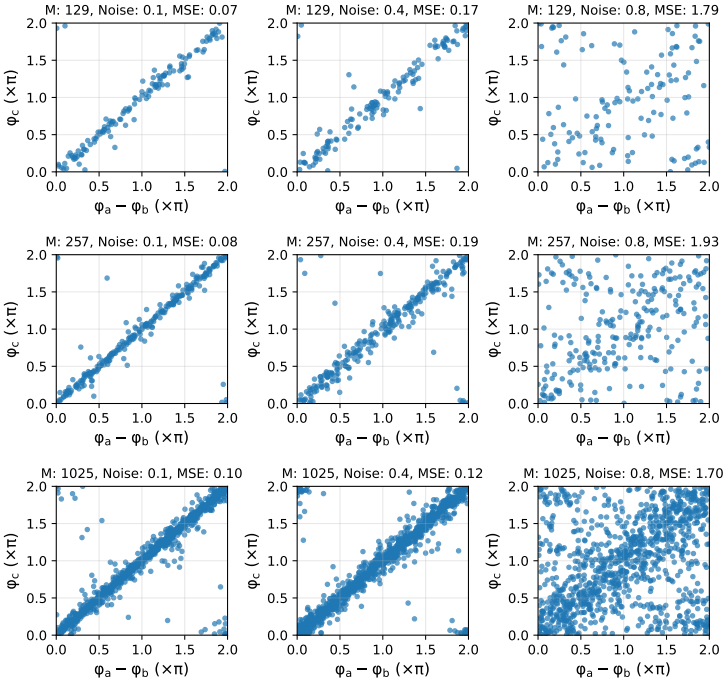


Figure 28: Scatter plot showing the relationship of phases on modular subtraction tasks. Each point represents a neuron, where φ_a, φ_b, and φ_c are the phases of $\boldsymbol{u}^G$, $\boldsymbol{v}^G$, and $\boldsymbol{w}^G$, respectively.

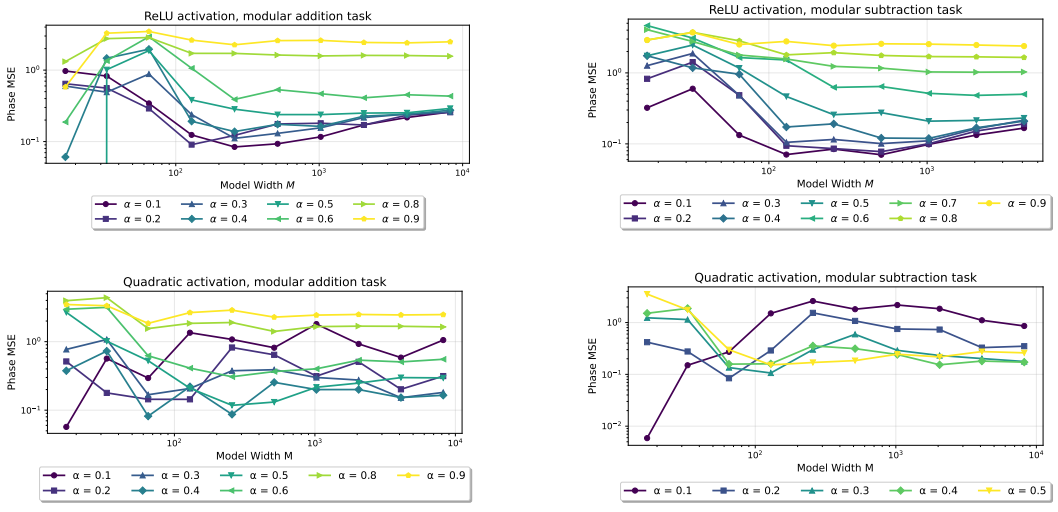


Figure 29: Phase MSE across different tasks, activation functions, model widths, and noise ratios. Higher noise ratios increase Phase MSE for ReLU models, and a similar trend is observed for quadratic models on the modular addition task.

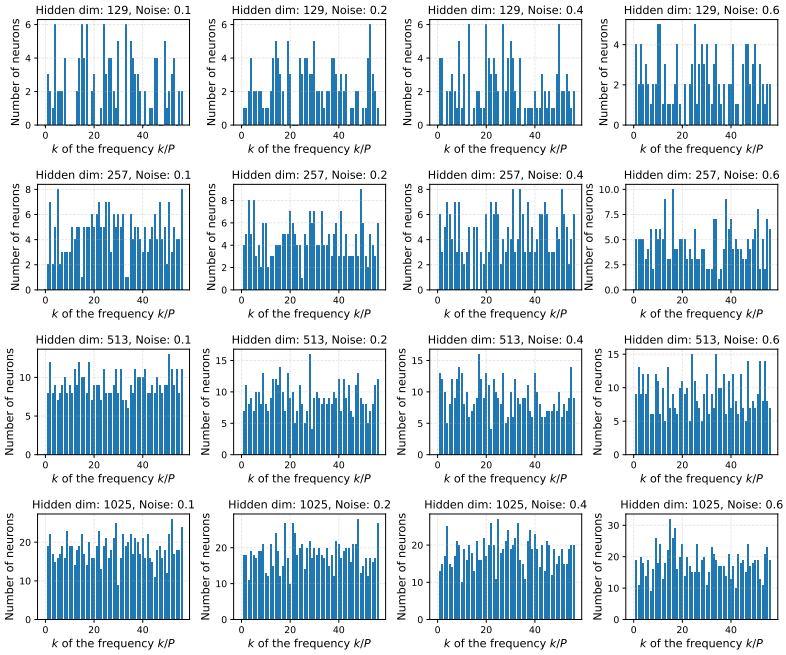


Figure 30: Each frequency $\omega \in \left\{1, \cdots, \frac{P-1}{2}\right\}$ is covered by at least one neuron when the model width is no less than 513.

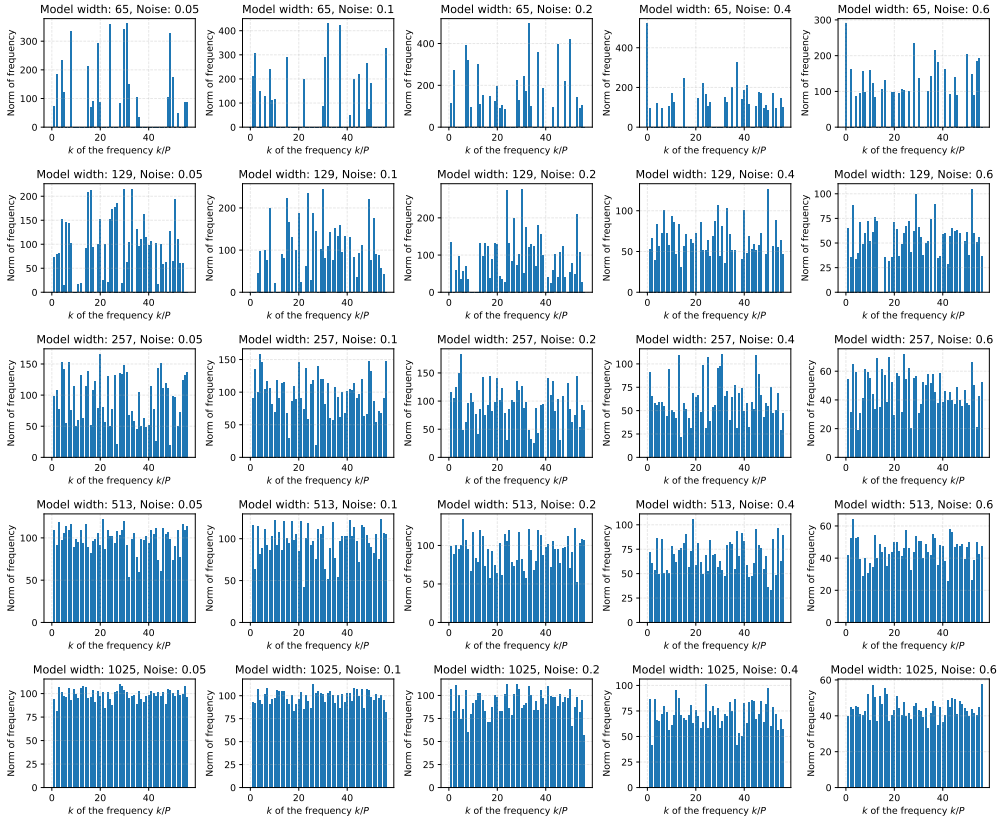


Figure 31: The magnitude (norm) of each frequency $\omega \in \left\{1, \cdots, \frac{P-1}{2}\right\}$ of $\boldsymbol{W}^G$ after frequency filtration to each neuron.

respectively. The filtered model has low accuracy on corrupted labels. The residual still fits most corrupted labels, although its accuracy decreases slightly at $M = 513$. Its test accuracy remains low.

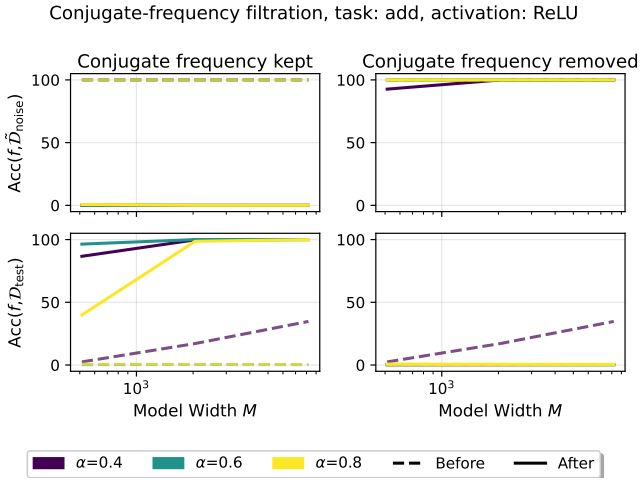


Figure 32: Conjugate-pair filtering on trained ReLU models. The left column retains the selected pair, and the right column removes it. The top row reports accuracy on corrupted training labels, and the bottom row reports test accuracy. Dashed lines show the original model, and solid lines show the filtered models.

Random-frequency filtering. For each neuron m, we independently sample one frequency k_m uniformly from $\{0, \ldots, P-1\}$, including the DC component. We retain this coefficient in the Fourier transforms of $\boldsymbol{u}_m$, $\boldsymbol{v}_m$, and $\boldsymbol{w}_m$ and set all others to zero. The conjugate frequency is not added. The residual removes the same selected coefficient. We then apply the inverse Fourier transform and take the real part. We repeat the selection with random-filter seeds 0, 1, 2, 3, and 4. The model weights and data remain fixed. We report the mean and standard deviation across these five selections.

Figure 33 shows that random-frequency filtering does not consistently improve test accuracy. It can produce large gains in some settings: at $M = 8193$ and $\alpha = 0.6$, mean test accuracy increases from 0.31% to 88.39%. However, at $\alpha = 0.4$, it decreases from 34.73% to 15.77%. At $\alpha = 0.8$, it reaches only 30.82%, compared with 100.00% for conjugate-pair filtering. The residual largely preserves the original model's accuracy on both corrupted labels and test data. These results show that the choice of retained frequency matters, although non-dominant frequencies can also contain useful information.

Filtering randomly initialized models. We initialize an untrained model of each width using the default PyTorch initialization with seed 42. We then apply the original single-frequency FF procedure. For each neuron, we select the smallest index among those with the largest magnitude in $\tilde{\boldsymbol{w}}_m$ and retain the matching coefficient in $\tilde{\boldsymbol{u}}_m$, $\tilde{\boldsymbol{v}}_m$, and $\tilde{\boldsymbol{w}}_m$. We do not add its conjugate frequency. After the inverse Fourier transform, we take the real part. We evaluate test accuracy before and after filtering on the same test split used for the trained models. Since the untrained model does not depend on the training noise ratio, we report one result per width.

Table 7 shows that test accuracy remains low before and after FF. All values are below 2%, while chance accuracy is approximately $100/113 \approx 0.88\%$. Thus, filtering alone does not reproduce the high test accuracy observed in trained models. This supports the role of learned weights in FF recovery, although the filter still uses a task-specific Fourier prior.

D.6 Frequency filtration across random seeds

We repeat the FF experiments with seeds 0 and 123. Figure 34 shows similar recovery patterns in both runs. At $\alpha = 0.8$, FF recovers near-perfect test accuracy at widths 4097 and 8193, despite the low test accuracy of the original models. The filtered models have low accuracy on corrupted labels, while the residual models retain high corrupted-label accuracy and low test accuracy. Together with the seed-42 results, these runs show that recovery under 80% label noise is not specific to one seed.

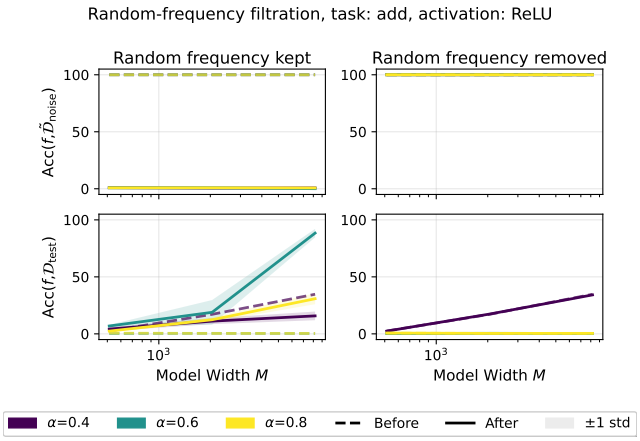


Figure 33: Random-frequency filtering on trained ReLU models. The left column retains one randomly selected frequency per neuron, and the right column removes the same frequency. The top row reports accuracy on corrupted training labels, and the bottom row reports test accuracy. Dashed lines show the original model. Solid lines and shaded bands show the mean and one standard deviation over five random-filter seeds.

Table 7: Test accuracy (%) before and after single-frequency FF on randomly initialized models. Initialization seed: 42. No training is performed.

Model width M	Before FF	After FF
513	1.15	1.17
2049	0.78	0.76
8193	0.78	1.72

E Supplementary Materials for Section 5

E.1 Performance of Algorithm 1 using Str. or IPR

In this section, we examine generalization improvements across different tasks achieved through neuron selection. Figure 35 reports the improvement obtained using IPR for neuron selection, while Figure 36 presents the corresponding results for Str.

Comparing the results on the modular multiplication task illustrates the generality of the method using Str. This task lacks periodic patterns relevant for generalization, so measuring neuron importance via IPR offers limited improvement.

E.2 Effect of the output bias

Both FF and neuron selection retain the original output bias ($\boldsymbol{\mu}$) when evaluating the selected and residual models. To check its effect, we evaluate these models again without the bias. Table 8 reports the results with weight decay 0.1 and seed 42. Removing the bias changes accuracy by at most 0.94 percentage points. The performance gap between FF and neuron selection remains large.

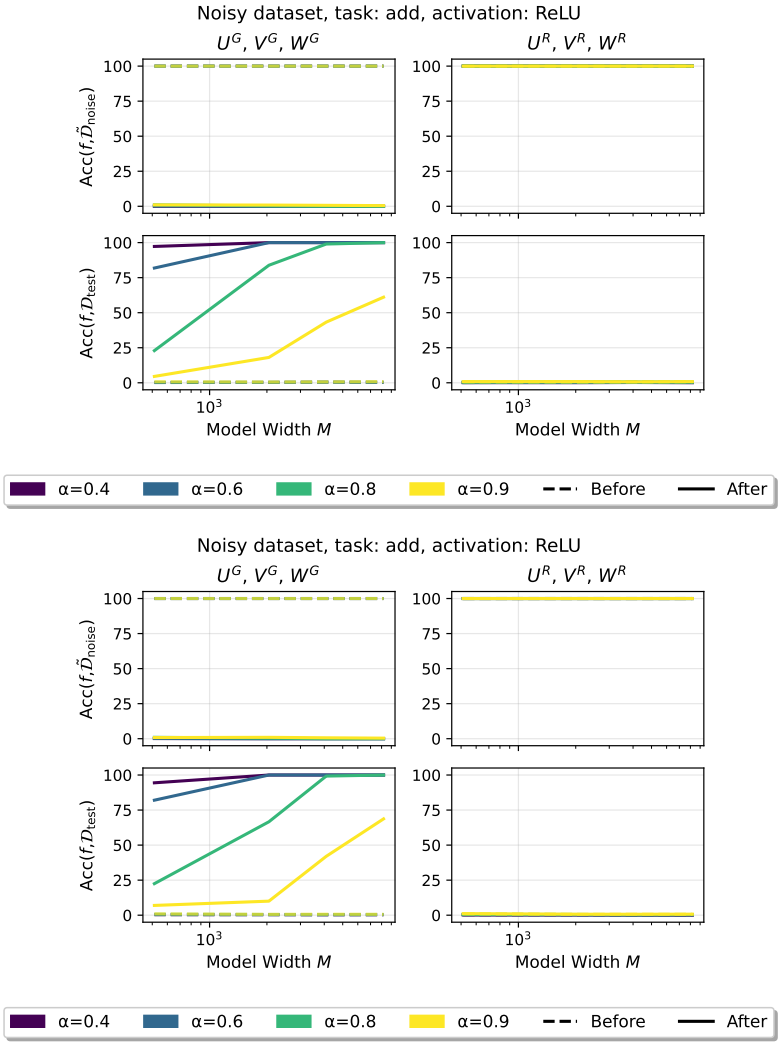


Figure 34: Frequency filtration with seed 0 (top) and seed 123 (bottom) on modular addition. Within each plot, the left column shows the dominant-frequency model, and the right column shows the residual model. The upper row reports accuracy on corrupted training labels, and the lower row reports test accuracy.

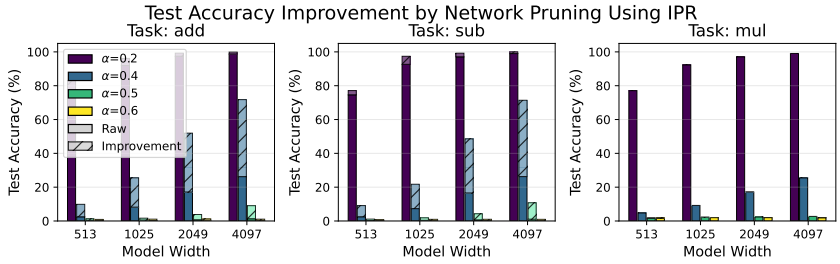


Figure 35: Generalization improvement across various tasks by neuron selection/pruning using IPR. No noticeable improvement is observed for the modular multiplication task.

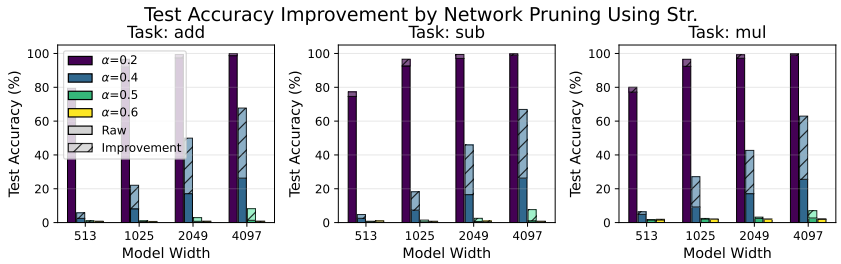


Figure 36: Generalization improvement across various tasks by neuron selection/pruning using Str. Significant improvement is also observed for the modular multiplication task.

Table 8: Effect of removing the output bias. Values are accuracies (%) with weight decay 0.1 and seed 42 at $\alpha = 0.4$. For FF, we report test accuracy of the selected model and corrupted-label accuracy of the residual model. For neuron selection, we report test accuracy of f^G and corrupted-label accuracy of f^R.

		FF		Neuron selection	
Width M	Bias	Selected (Test)	Residual (Noisy)	f^G (Test)	f^R (Noisy)
1025	With	97.81	100.00	22.08	99.10
	Without	96.87	100.00	21.79	99.10
4097	With	100.00	100.00	67.75	98.39
	Without	100.00	100.00	67.04	98.71
8193	With	100.00	100.00	83.43	99.22
	Without	100.00	100.00	82.62	99.61

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes], [No], or [N/A].
- [N/A] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for [N/A]).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will also be asked to include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While [Yes] is generally preferable to [No], it is perfectly acceptable to answer [No] provided a proper justification is given (e.g., error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering [No] or [N/A] is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

1. **Claims**

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

   Answer: [Yes]

   Justification: See Page 1-2.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: See the conclusion paragraph.

   Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [N/A]

   Justification: No theoretical results.

   Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: We have provided all the hyperparameters needed to reproduce the results, including the random seed.

   Guidelines:

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

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

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

   Answer: [No]

Justification: This setup is easy to implement and all the hyperparameters have been provided.

Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: See Section 2.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [N/A]

   Justification: The experiments are implemented across varying setups (optimizers, training ratios, etc.) to validate the conclusions.

   Guidelines:

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

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

8. **Experiments compute resources**

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

   Answer: [No]

   Justification: The setups are small enough to run on the CPU.

   Guidelines:

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: N/A.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [N/A]

    Justification: No social impact.

    Guidelines:

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

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

11. **Safeguards**

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

    Answer: [N/A]

    Justification: The paper poses no such risks.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [N/A]

    Justification: The paper does not use existing assets.

    Guidelines:

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

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

13. **New assets**

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

    Answer: [N/A]

    Justification: The paper does not release new assets.

    Guidelines:

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

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

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

    Answer: [N/A]

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

    Guidelines:

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

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

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

    Answer: [N/A]

    Justification: N/A.

    Guidelines:

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

16. **Declaration of LLM usage**

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

Answer: [N/A]

Justification: N/A.

Guidelines:

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