A TALE OF TWO CIRCUITS: GROKKING AS COMPETITION OF SPARSE AND DENSE SUBNETWORKS

William Merrill*, **Nikolaos Tsilivis*** **& Aman Shukla**
New York University

ABSTRACT

Grokking is a phenomenon where a model trained on an algorithmic task first overfits but, then, after a large amount of additional training, undergoes a phase transition to generalize perfectly. We empirically study the internal structure of networks undergoing grokking on the sparse parity task, and find that the grokking phase transition corresponds to the emergence of a sparse subnetwork that dominates model predictions. On an optimization level, we find that this subnetwork arises when a small subset of neurons undergoes rapid norm growth, whereas the other neurons in the network decay slowly in norm. Thus, we suggest that the grokking phase transition can be understood to emerge from *competition* of two largely distinct subnetworks: a dense one that dominates before the transition and generalizes poorly, and a sparse one that dominates afterwards.

1 INTRODUCTION

Grokking (Power et al., 2022; Barak et al., 2022) is a curious generalization trend for neural networks trained on certain algorithmic tasks. Under grokking, the network's accuracy (and loss) plot displays two phases. Early in training, the training accuracy goes to 100%, while the generalization accuracy remains near chance. Since the network appears to be simply memorizing the data in this phase, we refer to this as the *memorization phase*. Significantly later in training, the generalization accuracy spikes suddenly to 100%, which we call the *grokking transition*.

This mysterious pattern defies conventional machine learning wisdom: after initially overfitting, the model is somehow learning the correct, generalizing behavior without any disambiguating evidence from the data. Accounting for this strange behavior motivates developing a theory of grokking rooted in optimization. Moreover, grokking resembles so-called emergent behavior in large language models (Zoph et al., 2022), where performance on some (often algorithmic) capability remains at chance below a critical scale threshold, but, with enough scale, shows roughly monotonic improvement. We thus might view grokking as a controlled test bed for emergence in large language models, and hope that understanding the dynamics of grokking could lead to hypotheses for analyzing such emergent capabilities. Ideally, an effective theory for such phenomena should be able to understand the causal mechanisms behind the phase transitions, predict on which downstream tasks they could happen, and disentangle the statistical (number of data) from the computational (compute time, size of network) aspects of the problem.

While grokking was originally identified on algorithmic tasks, Liu et al. (2023) show it can be induced on natural tasks from other domains with the right hyperparameters. Additionally, grokking-like phase transitions have long been studied in the statistical physics community (Engel & Van den Broeck, 2001), albeit in a slightly different setting (online gradient descent, large limits of model parameters and amount of data etc.). Past work analyzing grokking has reverse-engineered the network behavior in Fourier space (Nanda et al., 2023) and found measures of progress towards generalization before the grokking transition (Barak et al., 2022). Thilak et al. (2022) observe a "slingshot" pattern during grokking: the final layer weight norm follows a roughly sigmoidal growth trend around the grokking phase transition. This suggests grokking is related to the magnitude of neurons within the network, though without a clear theoretical explanation or account of individual neuron behavior.

*Equal contribution

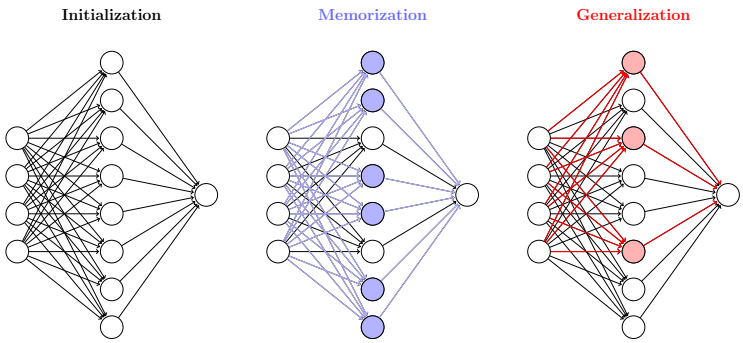

Figure 1: An illustration of the structure of a neural network during training in algorithmic tasks. Neural networks often exhibit a memorization phase that corresponds to a dense network, followed by the generalization phase where a sparse, largely disjoint to the prior one, model takes over.

In this work, we aim to better understand grokking on sparse parity (Barak et al., 2022) by studying the *sparsity* and *computational structure* of the model over time. We empirically demonstrate a connection between grokking, emergent sparsity, and competition between different structures inside the model (Figure 1). We first show that, after grokking, network behavior is controlled by a sparse subnetwork (but by a dense one before the transition). Aiming to better understand this sparse subnetwork, we then demonstrate that the grokking phase transition corresponds to accerelated norm growth in a *specific* set of neurons, and norm decay elsewhere. After this norm growth, we find that the targeted neurons quickly begin to dominate network predictions, leading to the emergence of the sparse subnetwork. We also find that the size of the sparse subnetwork corresponds to the size of a disjunctive normal form circuit for computing parity, suggesting this may be what the model is doing. Taken together, our results suggest grokking arises from targeted norm growth of specific neurons within the network. This targeted norm growth sparsifies the network, potentially enabling generalizing discrete behavior that is useful for algorithmic tasks.

2 Tasks, Models, and Methods

Sparse Parity Function. We focus on analyzing grokking in the problem of learning a sparse (n, k)-parity function (Barak et al., 2022). A (n, k)-parity function takes as input a string $x \in \{\pm 1\}^n$ returns $\pi(x) = \prod_{i \in S} x_i \in \{\pm 1\}$, where S is a fixed, *hidden* set of k indices. The training set consists of N i.i.d. samples of x and $\pi(x)$. We call the (n, k)-parity problem *sparse* when $k \ll n$, which is satisfied by our choice of $n = 40$, $k = 3$, and $N = 1000$.

Network Architecture. Following Barak et al. (2022), we use a 1-layer ReLU net $f(x) = u^\intercal \sigma(Wx + b), \tilde{y} = \operatorname{sgn}(f(x))$ where $\sigma(x) = \max(0, x)$ is ReLU, $u \in \mathbb{R}^p, W \in \mathbb{R}^{p \times n}$, and $b \in \mathbb{R}^p$. We minimize the hinge loss $\ell(x, y) = \max(0, 1 - f(x)y)$, using stochastic gradient descent (batch size B) with constant learning rate η and (potential) weight decay of strength λ (that is, we minimize the regularized loss $\ell(x, y) + \lambda \|\theta\|_2$, where θ denotes all the parameters of the model). Unless stated otherwise, we use weight decay $\lambda = 0.01$, learning rate $\eta = 0.1$ batch size $B = 32$, and hidden size $p = 1000$. We train each network 5 times, varying the random seed for generating the train set and training the model, but keeping the test set of 100 points fixed.

2.1 Active Subnetworks and Effective Sparsity

We use a variant of weight magnitude pruning (Mozer & Smolensky, 1989) to find active subnetworks that control the full network predictions. The method assumes a given support of input data $\mathcal{X}$. Let f be the network prediction function and f_k be the prediction where the $p - k$ neurons with the least-magnitude incoming edges have been pruned. We define the *active subnetwork* of f as f_k where k is minimal such that, for all $x \in \mathcal{X}$, $f(x) = f_k(x)$. We will use the active subnetwork to identify important structures within a network during grokking. We can also naturally use it to measure

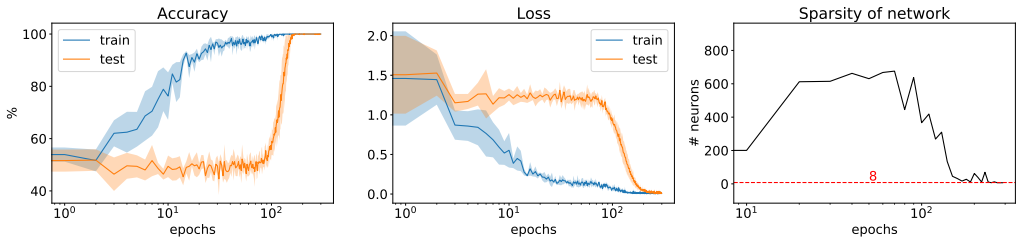

Figure 2: Accuracy (left), Average Loss (middle) and Effective Sparsity (right) during training of an FC network on $(40, 3)$ parity. Generalization accuracy suddenly jumps from random chance to flawless prediction concurrent with sparsification of the model. Shaded areas show randomness over the training dataset sampling, model initialization, and stochasticity of SGD (5 random seeds).

sparsity: we define the *effective sparsity* of f as the number of neurons in the hidden layer of the active subnetwork of f.

3 Results

We see in Figure 2 that our sparse parity task indeed displays grokking, both in accuracy and loss. We now turn to analyzing the internal network structure during training. We refer to Appendix C for additional configurations (smaller weight decay or larger parity size) that support our findings[1].

3.1 Grokking Corresponds to Sparsification

Figure 2 (right) shows the effective sparsity (number of active neurons; cf. Section 2.1) of the network over time. Noticeably, it becomes orders of magnitude sparser as it starts generalizing to the test set, and crucially, this phase transition happens at the same time as the loss phase transition. This can be directly attributed to the norm regularization being applied in the loss function, as it kicks in right after we reach (almost) zero in the data-fidelity part of the loss. Interestingly, this phase transition can be calculated solely from the training data but correlates with the phase transition in the test accuracy. Nanda et al. (2023) observe sparsity in the Fourier domain after grokking, whereas we have found it in the conventional network structure as well. Motivated by the discovery of this sparse subnetwork, we now turn our attention to understanding why this subnetwork emerges and its structure.

3.2 Selective Norm Growth Induces Sparsity During Grokking

Having identified a sparse subnetwork of neurons that emerges to control network behavior after the grokking transition, we study the properties of these neurons throughout training (before the formation of the sparse subnetwork). Figure 3 (left) plots the average neuron norm for three sets of neurons: the neurons that end up in the sparse subnetwork, the complement of those neurons, and a set of random neurons with the same size as the sparse subnetwork. We find that the 3 networks have similar average norm up to a point slightly before the grokking phase transition, at which the generalizing subnetwork norm begins to grow rapidly.

In Figure 3 (right), we measure the faithfulness of the neurons in the sparse subnetwork over time: in other words, the ability of these neurons alone to reconstruct the full network predictions on the test set, measured as accuracy. The grokking phase transition corresponds to these networks emerging to fully explain network predictions, and we believe this is likely a causal effect of norm growth.[2] The fact that the performance of its complement degrades after grokking supports the conclusion that the sparse network is "competing" with another network to inform model predictions, and that grokking corresponds to a sudden switch where the sparse network dominates model output. The element of

[1]Code available on `https://github.com/Tsili42/parity-nn`

[2]Conventional machine learning wisdom associates small weight norm with sparsity, so it may appear counterintuitive that *growing* norm induces sparsity. We note that the growth of selective weights can lead to effective sparsity because the large weights dominate linear layers (Merrill et al., 2021).

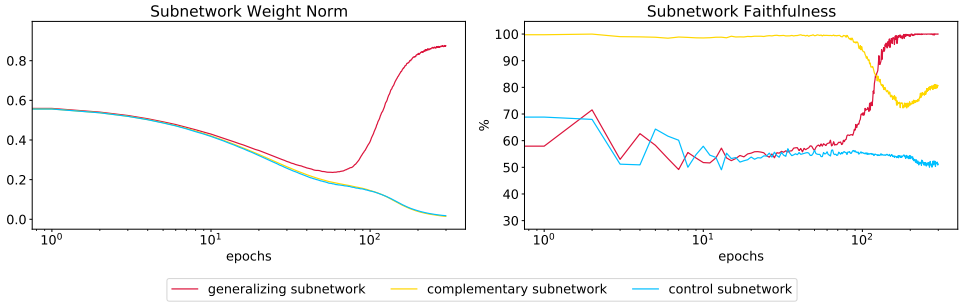

Figure 3: Left: Average norm of different subnetworks during training. Right: Agreement between the predictions of a subnetwork and the full network on the test set. The *generalizing* subnetwork is the final sparse net, the *complementary* subnetwork is its complement, and the *control* subnetwork is a random network with the same size as the generalizing one.

competition between the different circuits is further evident when plotting the norm of individual neurons over time. Figure 5 in the Appendix shows that neurons active during the memorization phase slightly grow in norm before grokking but then "die out", while the the neurons of sparse subnetwork are inactive during memorization and then explode in norm. The fact that the model is overparameterized allows this kind of competition to take place.

3.3 Subnetwork Computational Structure

Sparse Subnetwork. Across 5 runs, the sparse subnetwork has size $\{6, 6, 6, 8, 8\}$. This suggests that the network may be computing the parity via a representation resembling disjunctive normal form (DNF), via the following argument. A standard DNF construction uses $2^k = 8$ neurons to compute the parity of $k = 3$ bits (Proposition 1). We also derive a modified DNF net that uses only 6 neurons to compute the parity of 3 bits (Proposition 2). Since our sparse subnetwork always contains either 6 or 8 neuron, we speculate it may always be implementing a variant of these constructions. However, there is an even smaller network computing an $(n, 3)$-parity with only 4 neurons via a threshold-gate construction, but it does not appear to be found by our networks (Proposition 3).

Dense Subnetwork. The network active during the so-called memorization phase is not exactly memorizing. Evidence for this claim comes from observing grokking on the binary operator task of Power et al. (2022). For the originally reported division operator task, the network obtains near zero generalization prior to grokking (Figure 4, right). However, switching the operator to addition, the generalization accuracy is above chance before grokking (Figure 4, left). We hypothesize this is because the network, even pre-grokking, can generalize to unseen data since addition (unlike division) is commutative. In this sense, it is not strictly memorizing the training data.

4 Conclusion

We have shown empirically that the grokking phase transition, at least in a specific setting, arises from competition between a sparse subnetwork and the (dense) rest of the network. Moreover, grokking seems to arise from selective norm growth of this subnetwork's neurons. As a result, the sparse subnetwork is largely inactive during the memorization phase, but soon after grokking, fully controls model prediction. We speculate that norm growth and sparsity may facilitate emergent behavior in large language models similar to their role in grokking. As preliminary evidence, Merrill et al. (2021) observed monotonic norm growth of the parameters in T5, leading to "saturation" of the network in function space.[3] More promisingly, Dettmers et al. (2022) observe that a targeted subset of weights in pretrained language models have high magnitude, and that these weights overwhelmingly explain model predictions. It would also be interesting to extend our analysis of grokking to large language models: specifically, does targeted norm growth subnetworks of large language models (Dettmers et al., 2022) facilitate emergent behavior?

[3]Saturation measures the discreteness of the network function, but may relate to effective sparsity.

5 ACKNOWLEDGEMENTS

This material is based upon work supported by the National Science Foundation under NSF Award 1922658.

REFERENCES

Boaz Barak, Benjamin L. Edelman, Surbhi Goel, Sham M. Kakade, Eran Malach, and Cyril Zhang. Hidden progress in deep learning: SGD learns parities near the computational limit. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. GPT3.int8(): 8-bit matrix multiplication for transformers at scale. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022.

A. Engel and C. Van den Broeck. *Statistical Mechanics of Learning*. Cambridge University Press, 2001.

Ziming Liu, Eric J Michaud, and Max Tegmark. Omnigrok: Grokking beyond algorithmic data. In *International Conference on Learning Representations*, 2023.

William Merrill, Vivek Ramanujan, Yoav Goldberg, Roy Schwartz, and Noah A. Smith. Effects of parameter norm growth during transformer training: Inductive bias from gradient descent. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 1766–1781, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.133. URL https://aclanthology.org/2021.emnlp-main.133.

Michael C. Mozer and Paul Smolensky. *Skeletonization: A Technique for Trimming the Fat from a Network via Relevance Assessment*, pp. 107–115. Morgan Kaufmann Publishers Inc., 1989. ISBN 1558600159.

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

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

Vimal Thilak, Etai Littwin, Shuangfei Zhai, Omid Saremi, Roni Paiss, and Joshua M. Susskind. The slingshot mechanism: An empirical study of adaptive optimizers and the \emph{Grokking Phenomenon}. In *Has it Trained Yet? NeurIPS 2022 Workshop*, 2022.

Barret Zoph, Colin Raffel, Dale Schuurmans, Dani Yogatama, Denny Zhou, Don Metzler, Ed H. Chi, Jason Wei, Jeff Dean, Liam B. Fedus, Maarten Paul Bosma, Oriol Vinyals, Percy Liang, Sebastian Borgeaud, Tatsunori B. Hashimoto, and Yi Tay. Emergent abilities of large language models. *TMLR*, 2022.

A BINARY OPERATOR EXPERIMENTS

We trained a decoder only transformer with 2 layers, width 128, and 4 attention heads (Power et al., 2022). In both operator settings, we used the AdamW optimizer, with a learning rate of 10^{-3}, $\beta_1 = 0.9$ and $\beta_2 = 0.98$, weight decay equal to 1, batch size equal to 512, 9400 sample points and an optimization limit of 10^5 updates. We repeated the experiments for both operators with 3 random seeds and aggregated the results.

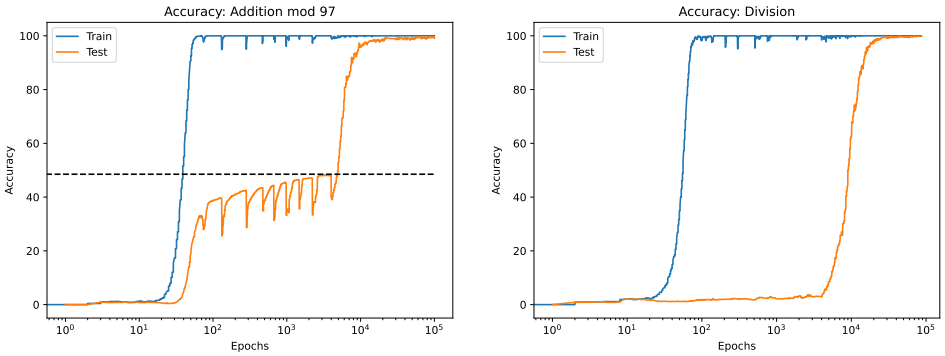

Figure 4: Accuracy curves for addition (left) and division (right). For the addition operator, the dashed line represents the % of dataset that can be solved by commuting test points and then looking them up in the memorized training set. The generalization accuracy before grokking matches this level, suggested that the network has learned to generalize the commutative property of addition before it learns to generalize fully.

B Additional Plots

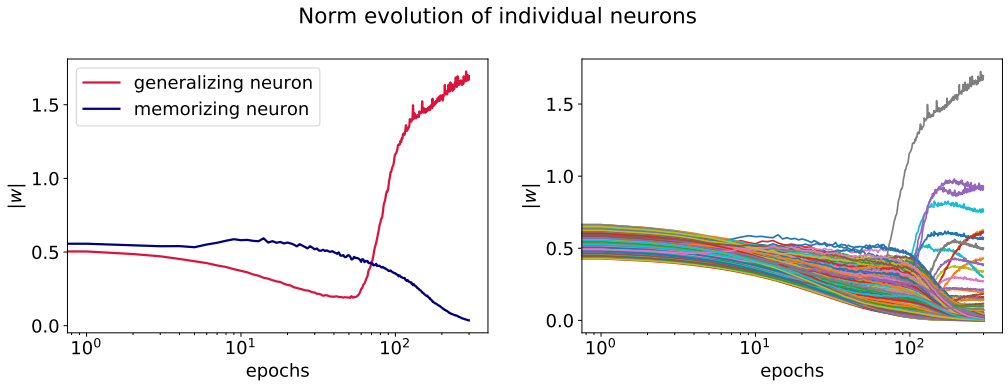

Figure 5: Weight norm of individual neurons during training. Left: Evolution of the dominant neurons during the memorization epoch (first time we hit $> 98\%$ train accuracy) and final epoch (that corresponds to the generalizing subnetwork). Right: Weight norm over time for all neurons. Notice that most of them are driven to 0.

C Additional Configurations

We provide accuracy, loss, sparsity, subnetwork norm and subnetwork faithfulness plots for smaller weight decay (Figures 6 and 7), and for larger parity size (Figures 8 and 9). The experimental observations are consistent with those of the main body of the paper.

D Computing Parity with Neural Nets

We say that a neural net of the form defined in section 2 computes parity iff its output is positive when the parity is 1 and negative otherwise.

We first show a general way to represent parity in ReLU networks for any parity size k. This construction requires 2^k hidden neurons.

Proposition 1. *For any n, there exists a 1-layer ReLU net with 2^k neurons that computes (n, k)-parity.*

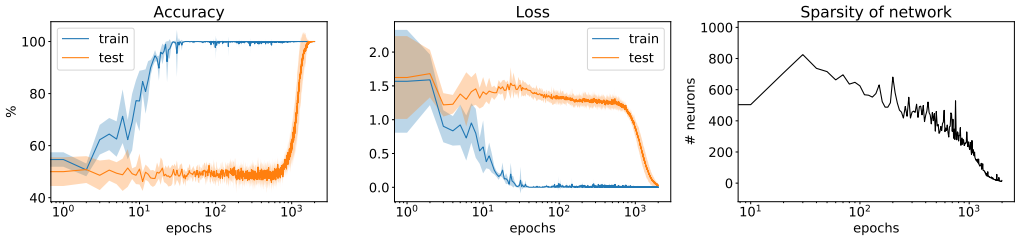

Figure 6: Reproduction of Figure 2 for smaller weight decay $\lambda = 0.001$ (the rest of the hyperparameters are the same as in the standard setup). Accuracy (left), Average Loss (middle) and Effective Sparsity (right) during training of an FC network on $(40, 3)$-parity.

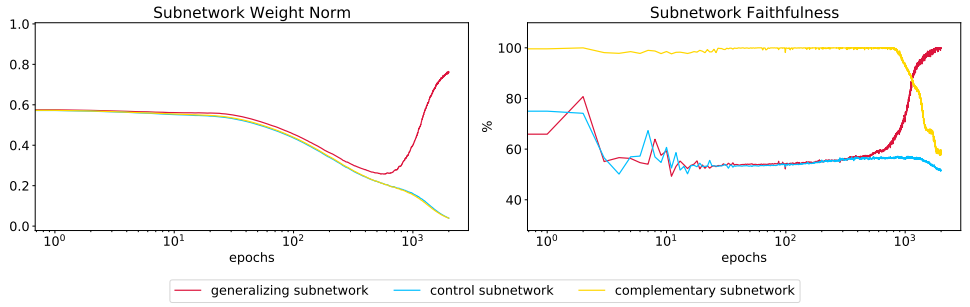

Figure 7: Reproduction of Figure 3 for smaller weight decay $\lambda = 0.001$ (the rest of the hyperparameters are the same as in the standard setup). Left: Average norm of different subnetworks during training. Right: Agreement between the predictions of a subnetwork and the full network on the test set.

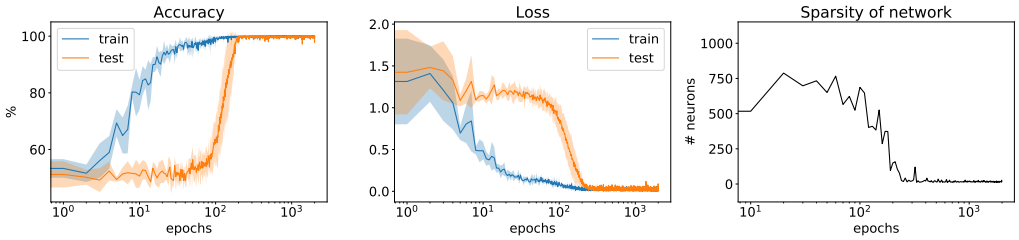

Figure 8: Reproduction of Figure 2 for larger parity size $k = 4$ (the rest of the hyperparameters are the same as in the standard setup). Accuracy (left), Average Loss (middle) and Effective Sparsity (right) during training of an FC network on $(40, 4)$ parity.

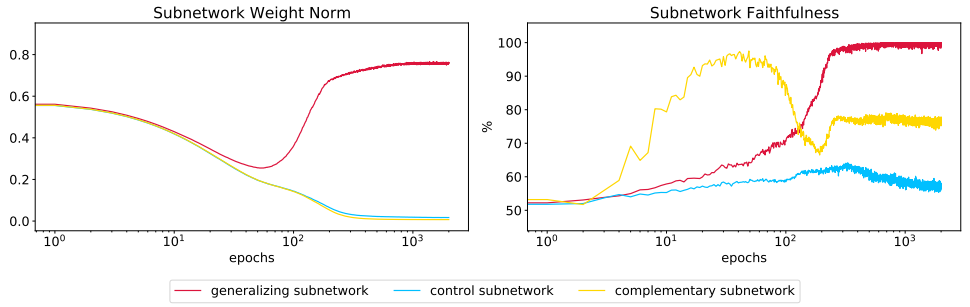

Figure 9: Reproduction of Figure 3 for larger parity size $k = 4$ (the rest of the hyperparameters are the same as in the standard setup). Left: Average norm of different subnetworks during training. Right: Agreement between the predictions of a subnetwork and the full network on the test set.

Proof. We use each 2^k neurons to match a specific configuration of the k parity bits by using the first affine transformation to implement an AND gate (note that the bias term is crucial here). In the output layer, we add positive weight on edges from neurons corresponding to configurations with parity 1 and negative weight for neurons corresponding to configurations with parity -1. □

In the case where $k = 3$, we show that there is a simpler construction with 6 neurons.

Proposition 2. *For any n, there exists a 1-layer ReLU net with 6 neurons that computes $(n, 3)$-parity.*

Proof. Let x_1, x_2, x_3 be the 3 parity bits. We construct $\mathbf{h} \in \mathbb{R}^6$ as follows, where σ is ReLU:

$$
\begin{aligned}
h_1 &= \sigma(-x_1 + -x_2 + 10x_3 - 9) \\
h_2 &= \sigma(-x_1 - x_2 + x_3 - 2) \\
h_3 &= \sigma(x_1 + x_2 + x_3 - 2) \\
h_4 &= \sigma(x_1 - x_2 - 10x_3 - 9) \\
h_5 &= \sigma(-x_1 + x_2 - x_3 - 2) \\
h_6 &= \sigma(x_1 - x_2 - x_3 - 2).
\end{aligned}
$$

In the final layer, we assign h_1 and h_4 a weight of -1, and h_2, h_3, h_5, and h_6 a weight of $+10$.

To show correctness, we first characterize the logical condition that each neuron encodes:

$$
\begin{aligned}
h_1 > 0 &\iff (x_1 = -1 \vee x_2 = -1) \wedge x_3 = 1 \\
h_2 > 0 &\iff x_1 = -1 \wedge x_2 = -1 \wedge x_3 = 1 \\
h_3 > 0 &\iff x_1 = 1 \wedge x_2 = 1 \wedge x_3 = 1 \\
h_4 > 0 &\iff (x_1 = 1 \vee x_2 = -1) \wedge x_3 = -1 \\
h_5 > 0 &\iff x_1 = -1 \wedge x_2 = 1 \wedge x_3 = -1 \\
h_6 > 0 &\iff x_1 = 1 \wedge x_2 = -1 \wedge x_3 = -1.
\end{aligned}
$$

In the final layer, h_1 and h_4 contribute a weight of -1 whenever the parity is negative (and in two other cases). But in the four cases when the true parity is positive, one of the other neurons contributes a positive weight of $+10$. Thus, the sign of the network output is correct in all 8 cases. We conclude that this 6-neuron network correctly computes the parity of x_1, x_2, and x_3. □

However, there is a 4-neuron construction computing parity,[4] which, interestingly, our networks do not find:

Proposition 3. *For any n, there exists a 1-layer ReLU net with 4 neurons that computes $(n, 3)$-parity.*

Proof. Let $X = x_1 + x_2 + x_3$ be the sum of the parity bits. We construct $\mathbf{h} \in \mathbb{R}^4$ as follows:

$$
\begin{aligned}
h_1 &= 1 \\
h_2 &= \sigma(X - 1) \\
h_3 &= \sigma(X + 1) \\
h_4 &= \sigma(-X - 1).
\end{aligned}
$$

In the final layer, we assign h_1 a weight of 1, h_2 a weight of 2, and h_3 and h_4 a weight of -1. We proceed by cases over the possible values of $X \in \{\pm 1, \pm 3\}$, which uniquely determines the parity:

1. $\underline{X = -3}$: Then there are three input bits with value -1, so the parity is -1. We see that $h_1 = 1, h_2 = 0, h_3 = 0$, and $h_4 = 2$. So the output is $h_1 - h_4 = -1$.
2. $\underline{X = -1}$: Then there are two input bits with value -1, so the parity is 1. We see that $h_1 = 1$, $h_2 = 0, h_3 = 0$, and $h_4 = 0$. So the output is $h_1 = 1$.
3. $\underline{X = 1}$: Then there is one input with value -1, so the parity is -1. We see that $h_1 = 1$, $h_2 = 0, h_3 = 2$, and $h_4 = 0$. So the output is $h_1 - h_3 = -1$.

[4] We thank anonymous reviewer 3kdq for demonstrating this construction.

4. $\underline{X = 3}$: Then there are no inputs with value -1, so the parity is 1. We see that $h_1 = 1$, $h_2 = 2$, $h_3 = 4$, and $h_3 = 0$. So the output is $h_1 + 2h_2 - h_3 = 1$.

We conclude that this 4-neuron network correctly computes the parity of x_1, x_2, and x_3. □