Polarity Is All You Need to Learn and Transfer Faster

Qingyang Wang [1 2] **Michael A. Powell** [3 2 4] **Eric W. Bridgeford** [5 2] **Ali Geisa** [3 2] **Joshua T. Vogelstein** [3 1 5 6 2]

Abstract

Natural intelligences (NIs) thrive in a dynamic world – they learn quickly, sometimes with only a few samples. In contrast, artificial intelligences (AIs) typically learn with a prohibitive number of training samples and computational power. What design principle difference between NI and AI could contribute to such a discrepancy? Here, we investigate the role of weight polarity: development processes initialize NIs with advantageous polarity configurations; as NIs grow and learn, synapse magnitudes update, yet polarities are largely kept unchanged. We demonstrate with simulation and image classification tasks that if weight polarities are **adequately** set ***a priori***, then networks learn with less time and data. We also explicitly illustrate situations in which *a priori* setting the weight polarities is disadvantageous for networks. Our work illustrates the value of weight polarities from the perspective of statistical and computational efficiency during learning.

1. Introduction

Natural intelligences (NIs), including those of animals and humans, are able to learn and adapt rapidly in real world environments with limited samples. Artificial intelligences (AIs), specifically deep neural networks (DNNs), can now compete with or even surpass humans in certain tasks, e.g., the game GO (Silver et al., 2017), object recognition (Russakovsky et al., 2015), protein folding analysis (Jumper et al., 2021), etc. However, a DNN is only capable of such achievements when prohibitive amounts of data and training resources are available. Such gaps in learning speed and data efficiency between NI and AI have baffled and motivated many AI researchers. A subfield of AI is dedicated to achieving few-shot learning using DNNs (Hoffer & Ailon, 2015; van der Spoel et al., 2015; Vinyals et al., 2016; Snell et al., 2017; Finn et al., 2017). Many research teams have achieved high performance on benchmark datasets (Lazarou et al., 2022; Bendou et al., 2022). However, the products of such engineering efforts greatly deviate from the brain. What are the design principle differences between NIs and AIs that contribute to such a gap in learning efficiency? In this paper, we propose two possible design differences under one theme: weight polarity.

NIs are blessed with hundreds of millions of years of optimization through evolution. Through trial and error, the most survival-advantageous circuit configurations emerge, refine, and slowly come into the form that can thrive in an ever-changing world. Such circuit configurations are embedded into genetic code, establishing a blueprint to be carried out by development. It is hypothesized that such canonical circuits provide an innate mechanism that facilitates rapid learning for NIs (Zador, 2019). Unlike transfer learning for AIs where the entirety of the weights is adopted as a form of prior knowledge, NIs pay substantial, if not exclusive, attention to polarity patterns (excluding magnitudes) when transferring such canonical circuits across generations. On the one hand, polarity is a more compressed knowledge carrier than weight, rendering it a genetic-code-saving design choice. On the other hand, is polarity alone enough to transfer knowledge between networks? NIs suggest yes; AIs' answers are yet to be explored.

Furthermore, post-development neuronal connections in the brain rarely see polarity switch (Spitzer, 2017). After development, NIs learn and adapt through synaptic plasticity – a connection between a pair of neurons can change its strength but rarely its excitatory or inhibitory nature; on the contrary, a connection (weight) between a pair of units in a DNN can freely change its sign (polarity). For the rare times polarity switching is observed in the brain, they never appeared in sensory and motor cortices (Spitzer, 2017) where visual, auditory and motor processing take place. It seems a rather rigid design choice to fix a network's connection polarity. Is it a mere outcome of an implementation-level constraint?

[1]Department of Neuroscience, Johns Hopkins University, Baltimore MD US [2]Center for Imaging Science, Johns Hopkins University, Baltimore MD US [3]Department of Biomedical Engineering, Johns Hopkins University, Baltimore MD US [4]Current location: United States Military Academy, Department of Mathematical Sciences, West Point NY US [5]Department of Biostatistics, Johns Hopkins University, Baltimore MD US [6]Institute for Computational Medicine, Kavli Neuroscience Discovery Institute, Johns Hopkins University, Baltimore MD US. Correspondence to: Qingyang Wang <qwang88@jhu.edu>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

Perhaps it is quite difficult for synapses to switch polarities. Or could it be that fixing polarity is a more efficient learning strategy? This is yet another unexplored design principle AIs may borrow from NIs.

This paper provides some thoughts and evidence in applying these two design principles to DNNs. We first discuss the trade-off between representation capacity and learning speed when weight polarity is fixed for networks (Section 2). We experimentally show that if the weight polarities are **adequately** set ***a priori***, then networks can learn with less time and data (simulated task (Section 2) + two image classification tasks (Section 3)). We also discuss how the quality of the polarity configuration affects a DNN's learning efficiency (Section 2-3). We further find transferring + fixing polarities is even superior to transferring weights (Section 4). Our results point to an unexplored direction in the machine learning community: polarity, not weight, may be the more effective and compressed medium for transferring knowledge between networks. To complete our discussion, we further discuss what we may lose when weight polarities are set *a priori* (Section 5).

By discussing both the advantages and disadvantages of setting polarity *a priori*, we provide some insights on how to make AI more statistically and computationally efficient.

2. What do we gain by setting weight polarity *a priori*?

Networks need both positive and negative weights to function (Wang et al., 2023) - a DNN with all non-negative weights is not a universal approximator. Even when both polarities are present in a network, constraining a network's weight polarity pattern limits its representation capacity: when only half of the range is available to each connection, the reduction in total possible network patterns grows exponentially with more edges in the network. It seems counterintuitive for any network to have willingly chosen to give up on a vast portion of its representation capacity. Are they gaining elsewhere? Our thought is: maybe they learn faster. We write down the representation capacity and speed trade-off concisely in Lemma 2.1 and prove it in a constrained setting. We provide experimental evidence afterwards.

Lemma 2.1 (capacity-speed trade-off)**.** *If the weight polarities are set a priori, such that the function is still representable, then the network can learn faster.*

We prove Lemma 2.1 for single-hidden-layer networks, with the following assumptions:

> Assumption 1: The weights take on discrete values. This is essentially true for all DNNs implemented on silicon chips where all continuous variables are discretized.
>
> Assumption 2: Exhaustive search is the learning algorithm.

See proof on page 19.

Next, we use simulation to show that when trained with the Freeze-SGD (Algorithm 1), networks indeed learn more quickly when polarities are set *a priori* in such a way that the function is still representable (Def 2.2).

Definition 2.2 (representability)**.** With input space $\mathcal{X}$, we say a function f is representable by a DNN F when $\forall x \in \mathcal{X}, \epsilon > 0,\ \exists F$ such that $|f(x) - F(x)| < \epsilon$.

Freeze-SGD We design our freeze training procedure to be exactly the same as SGD (Adam optimizer) except for one single step: after each batch, all weights are compared to the preset polarity template, and if any weight has switched polarity in the last batch, it is reverted back to the desired polarity/sign (see Algorithm 1). As our goal is to see the pure effect of fixing weight polarity, we did not adopt any bio-plausible learning algorithms as they may introduce confounding factors.

We compared four training procedures in general:

1. **Frozen-Net Sufficient-Polarity**: Weight polarities were set *a priori*. The polarity pattern was chosen based on a rule that ensures the configuration is adequate for learning the task, i.e., the polarity pattern carries expert knowledge about the data.

2. **Frozen-Net RAND-Polarity**: Weight polarities were set *a priori* randomly: $Bernoulli(0.5)$. The polarity pattern does not carry any prior information about the data.

3. **Fluid-Net RAND-Polarity**: Weights (containing polarities) were initialized randomly; weight polarities were free to change throughout the training procedure.

4. **Fluid-Net Sufficient-Polarity** Weight polarities (not magnitudes) were initialized with prior knowledge; weight polarities were free to change throughout the training procedure. This scenario will only be discussed in Section 3 on image classification tasks.

Controlled Experiments To see the pure effect of setting weight polarity *a priori*, we require a suite of controlled measures. We controlled the following factors across all conditions: the weight magnitude distribution (Supplementary Figure C.6), network architecture, training samples (n varies, see Figure 1), learning rate, batch sequence of the training data, and the validation samples across all scenarios. More details about the experiments can be found in the Appendix Section B.

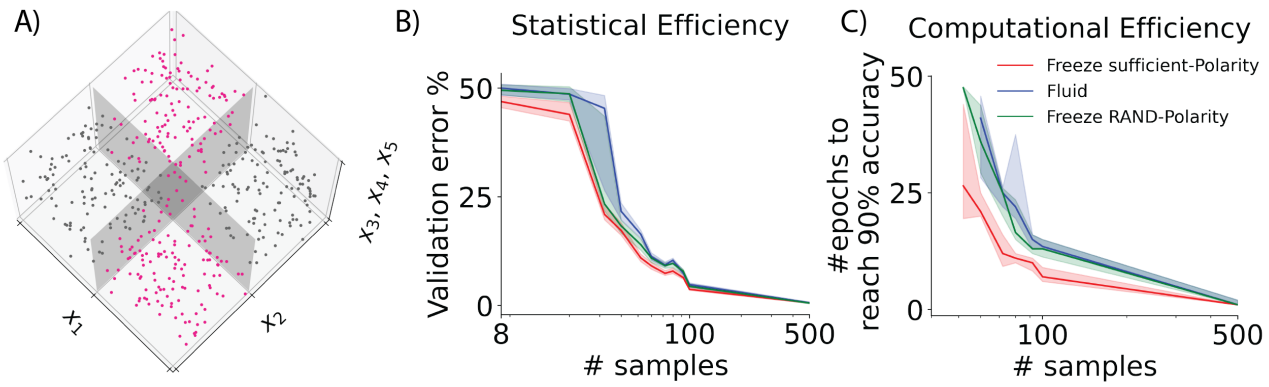

Figure 1. **Adequately setting weight polarity *a priori* allows networks learn faster (fewer epochs) and with less data (fewer training samples).** Single hidden layer networks with 64 hidden units were trained to learn XOR-5D (Section 2.1). In all scenarios, networks were trained for 100 epochs. A) XOR-5D, a binary classification problem where only the first two dimensions are relevant to the task (XOR), and the other three dimensions are noise following a normal distribution $\mathcal{N}(1, 1)$. B) Statistical efficiency: At convergence, adequately configured freeze networks (red) achieve the same level of performance (% error) with less data; randomly configured freeze networks (green) use similar and oftentimes less data than fluid (blue), e.g., to reach 80% accuracy (20% error). C) Computational efficiency: Setting weight polarity in a data-informed way (red) makes networks learn more quickly – this is true across all data-scarcity levels; when weight polarities are randomly fixed (green), networks learn as fast as their fluid counterparts (blue), and sometimes faster (e.g., training sample sizes = 80 and 92). For all experiments, n=50 trials. All curves correspond to medians with shaded regions 25th-75th percentiles.

2.1. XOR-5D Simulation Results

We used 5-dimensional XOR (XOR-5D) as the binary classification task for our simulation; only the first two dimensions are relevant to the task, and the remaining three dimensions are noise following a normal distribution $\mathcal{N}(1, 1)$ (Figure 1 panel A). For Frozen-Net Sufficient-Polarity, the polarity template is preset in this way: for each hidden unit, the polarity of the output edge is the sign product of the first two dimensions' input weights.

We first tried four different weight reset methods (details in Algorithm 1) and found they give us similar results and thus did not matter for our primary questions (Supplementary Figure C.1). For the rest of this paper, we chose the *pos-Rand* method, whereby we reset weights to a small random number of the correct sign.

When the polarities are fixed in such a way that it is sufficient to learn the task (red), networks always learn faster than networks without polarity constrains (blue) (Fig 1 panel B). This advantage is true across all data scarcity levels and is particularly valuable when data is scarce. When only 60 or 72 training samples were available, Frozen-Net Sufficient-Polarity on average takes 58% and 48% of the standard Fluid-Net training time, respectively, to reach the same level of accuracy. When weight polarities are randomly chosen (green), networks learn as fast as their Fluid-Net counterparts (blue), and sometimes faster (e.g., training samples = 80 and 92).

Frozen-Net Sufficient-Polarity not only saves time, but it also requires less data (Figure 1 panel C). At convergence, Frozen-Net Sufficient-Polarity (red) takes fewer samples to reach the same level of accuracy when compared to the standard Fluid-Net (blue). Randomly configured Frozen-Net (green) uses similar and oftentimes less data than Fluid-Net (blue) (e.g., to reach 80% accuracy, Frozen-Net RAND-Polarity uses less data than Fluid-Net).

We showed that setting weight polarity *a priori* makes networks learn in less time, with less data, provided the function is still representable. Even randomly configured Frozen-Nets show comparable and sometimes better performance than Fluid-Nets. Such a result is striking from an optimization perspective: Frozen-Net is at a disadvantage by design because the weight resetting step in Freeze-SGD (see Algorithm 1) fights against the gradient update, and part of the error update information is lost in this process. Regardless of this disadvantage during optimization, our Frozen-Net Sufficient-Polarity consistently outperforms Fluid-Net; even Frozen-Net RAND-Polarity is never worse than Fluid-Net. Combined, these results show we may be able to help AIs learn quickly and with fewer samples by doing two things: 1) fix weight polarities, and 2) choose the polarity pattern wisely. We will tease apart the effect of these two factors in the next section.

3. Effectiveness of setting weight polarity *a priori* in image classification tasks

In this section, we extend the experiments in Figure 1 to image classification tasks (Figure 2). Such complex learning

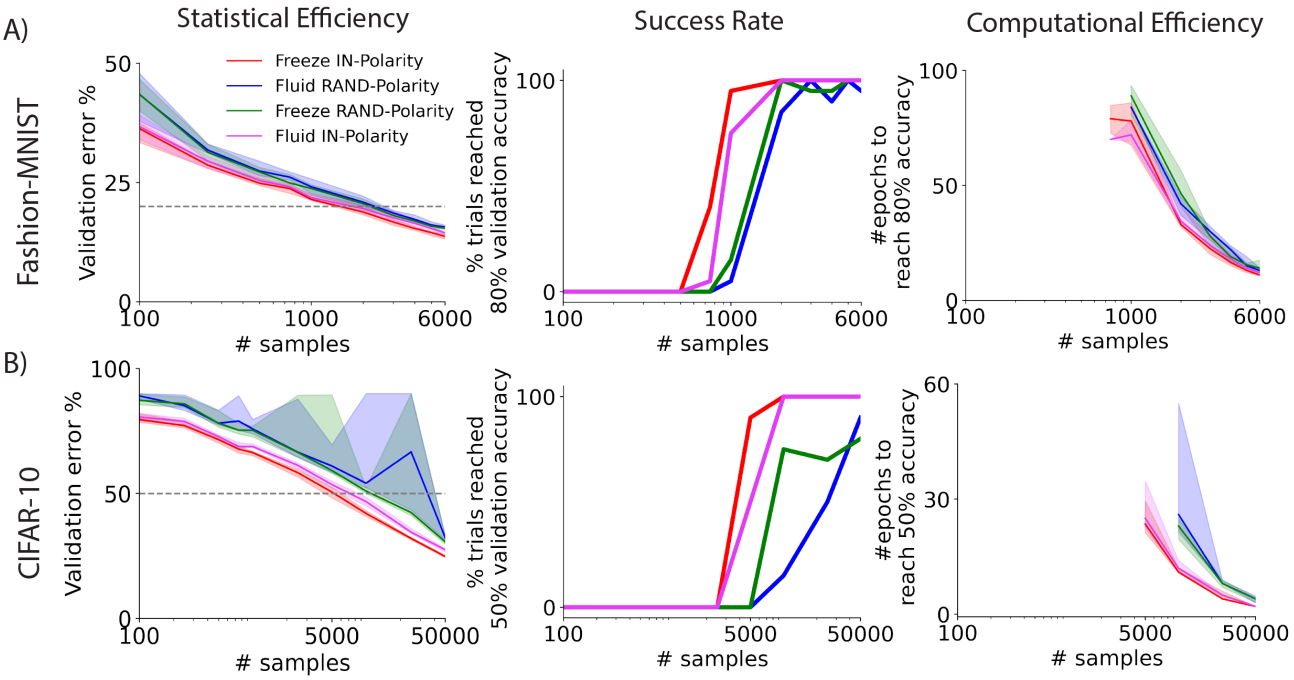

Figure 2. **DNNs with frozen ImageNet-Polarity (IN-Polarity) learn more quickly and with less data in image classification tasks.** A) Experiments on Fashion-MNIST image classification dataset. From left to right: 1) Statistical efficiency: Frozen-Nets with IN-Polarity (red) always learn with fewer samples than Fluid-Net (blue); the majority of the gain is contributed by the knowledge transferred from the initial polarity configuration (pink vs. blue); Frozen-Net RAND-Polarity (green) never performs worse than Fluid-Net (blue). 2) Frozen-Net IN-Polarity always has a higher chance of reaching 80% validation accuracy than Fluid-Net; Frozen-Net RAND-Polarity has comparable and sometimes a higher chance of reaching 80% validation accuracy than Fluid-Net. 3) Computational efficiency: Frozen-Net IN-Polarity always takes less time than Fluid-Net to reach 80% validation accuracy; again, effective knowledge transfer from the preset polarity pattern is the major contributing factor (pink vs. blue); Frozen-Net RAND-Polarity takes a similar number of computational iterations as Fluid-Net. B) Same as A except experiments were on the CIFAR-10 dataset, and the validation accuracy threshold is at 50%. Gray lines in the first column correspond to the validation accuracy thresholds used to plot the next two columns. For a comprehensive view of performance at different thresholds, see Supplementary Figure C.2. For statistical significance of the difference, see Figure 3. Both datasets: n=20 trials, 100 epochs, lr=0.001. No data augmentation was performed.

tasks do not have simple and explicit polarity configuration rules as in XOR-5D. We exploited an alternative strategy by using ImageNet trained polarities (IN-Polarity). A Frozen-Net IN-Polarity has its weight *magnitudes* initialized randomly, with its weight *polarities* initialized and fixed to match the IN-Polarity pattern. We also tested Fluid IN-Polarity where networks were initialized to the exact same pattern as Frozen-Net IN-Polarity, except polarities are free to switch while learning the task. This comparison helps us to understand which of the two factors contributed more to the performance gain: fixing polarities or knowledge transfer through the initial polarity pattern. We trained and tested networks on the Fashion-MNIST (grayscale) and CIFAR-10 (RGB-color) datasets, using AlexNet network architecture (Krizhevsky et al., 2017). For both datasets, we trained for 100 epochs, with lr=0.001 (best out of [0.1,0.03,0.01,0.001]). The AlexNet IN-weights were obtained here. Of note, we controlled all conditions to follow the same weight magnitude distribution at initialization Supplementary Figure C.6). Specifically, we randomly initialized the networks following conventional procedures: we used Glorot Normal for conv layers and Glorot Uniform for fc layers; we then either fixed the polarities as-is (RAND-Polarity) or flipped the polarities according to the ImageNet template (IN-Polarity), introducing no change to the magnitude distributions.

Across the board, Frozen-Net IN-polarity (red) always learns with fewer samples than Fluid-Net (blue) (Figure 2 first column). When only 100 training images were available (across the 10 classes), Frozen-Net IN-Polarity (red) yields 7% less validation error at convergence compared to Fluid-Net (blue) in the Fashion-MNIST task and 9.4% less error for CIFAR-10. Such a gain is mostly brought by knowledge transferred through the polarity pattern (pink vs. blue, 6% gain for Fashion-MNIST; 8.4% gain for CIFAR-10). Fixing the polarities can further bring performance gain: for CIFAR-10, 1% gain at 100 training samples, and up to 3% at 50000 training samples. The improvement on valida-

tion accuracy (less validation error) from Fluid IN-Polarity to Frozen-Net IN-Polarity is often statistically significant (Figure 3, column 1, pink line). When the polarities are fixed randomly (RAND-Polarity, green), networks never perform worse than Fluid-Net (blue).

Across the board, Frozen-Net IN-Polarity always learns with less time than Fluid-Net (Figure 2 third column). The majority of such a gain is brought by polarity pattern knowledge transfer. Frozen-Net RAND-Polarity takes a comparable number of computational iterations as Fluid-Net. Furthermore, not every network is able to reach the specified accuracy threshold (Fashion-MNIST: 80%, CIFAR-10: 50%; Figure 2 second column). Across the board, Frozen-Net IN-Polarity has a higher chance of passing the specified accuracy threshold than Fluid-Net; Frozen-Net RAND-Polarity has an equal and sometimes higher chance than Fluid-Net (Fashion-MNIST 1000 & 2000 samples; CIFAR-10 10000 & 25000 samples). These observations are true across different validation accuracy thresholds (Supplementary Figure C.2).

IN-Polarity-initialized networks in general show more consistent performance across trials compared to the Fluid setting for both Frozen-Net and Fluid-Net. This is especially obvious for the statistical efficiency plots: across the board, IN-Polarity always shows less variation in its performance on validation error (shaded area marks 25th-75th percentiles).

To provide a lens into the dynamics of polarity switching throughout learning, we analyzed the ratio of weight parameters (excluding bias terms) that switched polarity between two epochs for Fluid RAND-Polarity for the first 50 epochs (first half of training) - indeed, there are more polarity switches early on in training, and the ratio decays throughout training (Figure C.5). Such a trend is true across layers and across training sample sizes. This suggests polarities are mostly learned early on during training but also remain dynamic throughout the learning process.

Taken together, Frozen-Net IN-polarity consistently learns with less data and time and does so with a higher probability of success compared to Fluid-Net; the majority of the performance gain is brought by knowledge embedded in the initialized polarity pattern, with further gain possible by fixing weight polarities; Frozen-Net RAND-Polarity performs as well as Fluid-Net RAND-Polarity, sometimes better.

4. Transferring and fixing polarity is superior to transferring weights

From a transfer learning perspective, Frozen-Net IN-Polarity essentially transfers weight polarities instead of weights per se. How does polarity transfer compare to the traditional finetune (weight transfer) strategy? This time, instead of randomly initializing the weight magnitudes, we initialized the weight magnitudes based on the ImageNet-trained weights (IN-Weight). We compared them with Frozen-Net IN-Polarity by plotting their differences (Δ={LEGEND} - {Freeze IN-Polarity}) in Figure 3. The original curves before taking the differences can be found in Supplementary Figure C.3.

The orange curves compare weight transfer (Fluid-Net IN-Weight) with our polarity transfer strategy (Frozen-Net IN-Polarity). Across almost all data scarcity levels, our polarity transfer strategy achieves lower validation error (higher accuracy) than finetune (first column, orange $\geqslant 0$). Such a superiority is statistically significant (Mann-Whitney U two-tail; multiple comparisons corrected with Holm–Bonferroni method) when training data is limited (Fashion-MNIST: 250 and 1000 samples; CIFAR-10: $\leqslant 1000$ samples except 250). Polarity transfer also allows the networks to learn with higher probability (second column) and fewer epochs (third column, curve $\geqslant 0$). Such faster learning occurs regardless of the performance threshold value (Supplementary Figure C.4). In sum, transferring and fixing polarity is almost always superior to weight transfer in terms of both statistical and computational efficiency. Such an observation suggests polarity configuration is an effective medium, if not superior to weight pattern, for transferring knowledge between networks.

When networks were transferred with polarities but not frozen (Fluid IN-Polarity pink), they almost always perform better than Fluid IN-Weight (orange, Figure 3 & Supplementary Figure C.3 to see variance), this is true except in rare cases (e.g., CIFAR-10 2500 and 5000 samples) where the difference is not significant due to the wide performance variation of Fluid IN-Weight.

Surprisingly, when we initialized the Frozen-Net with IN-Weight (cyan), there is some gain in performance, but to a limited extent; in fact, it can sometimes be worse. When training data is limited (Fashion-MNIST $\leqslant 1000$, CIFAR-10 $\leqslant 1000$), Frozen-Net IN-Weight gained little in performance (cyan ≈ 0) and could be worse at times (CIFAR-10 500 and 1000 samples). When training data is more abundant, there is a more consistent accuracy gain by initializing Frozen-Net with IN-Weight (Fashion-MNIST gain $\sim 1\%$; CIFAR-10 gain $\sim 4\%$). Such a gain is discounted by more training iterations (third column), and often a lower likelihood of reaching a high level of performance (second column).

Furthermore, similar to random initialization, weight transfers tend to have wider performance variations compared to polarity transfers, for both Frozen and Fluid networks (Supplementary Figure C.3). The exact reason behind this observation remains to be explored; our current hypothesis is the stochasticity of sample batching: by only transferring polarity while initializing the magnitudes randomly, the learning process is more robust against such stochasticity.

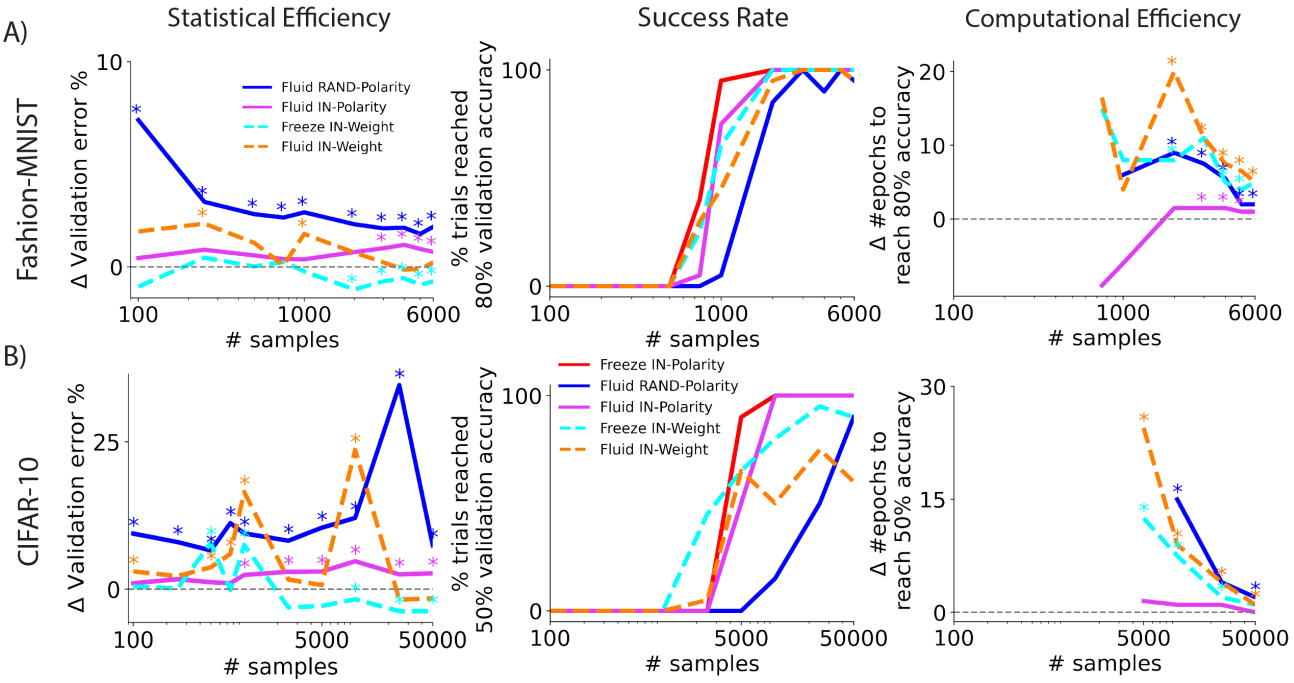

Figure 3. **Across all sample sizes, Frozen-Net with ImageNet-Polarities (IN-Polarities) learns more quickly than Fluid-Net with ImageNet-Weight (IN-Weight) initialization.** In the first and third columns, curves are the **median difference Δ={LEGEND} - {Freeze IN-Polarity}** (the second term is the red curve in Figure 2). Asterisks (*) indicate that the validation error is significantly higher than Freeze IN-Polarity (Mann-Whitney U two-tailed test, α=0.05). Multiple comparisons are corrected with Holm–Bonferroni method. 1) First column: Transferring and fixing polarity is more effective than transferring weights in terms of data efficiency. This is indicated by the orange curve above zero across most sample sizes, meaning Fluid-Net IN-Weight on average has higher validation error than Freeze IN-Polarity, and such differences are statistically significant for small sample sizes. Note the zero line here means LEGEND = Freeze IN-Polarity. 2) Second column: Frozen-Net IN-Polarity has a higher chance of reaching 80% (Fashion-MNIST) / 50% (CIFAR-10) validation accuracy compared to weight transfer (orange curve); 3) Third column: Frozen-Net IN-Polarity always takes fewer epochs to reach high validation accuracy than weight transfer (orange curve). Both datasets: n=20 trials, 100 epochs, lr=0.001. For the right two columns, the validation accuracy thresholds are the same as in Figure 2.

An alternative contributing factor that cannot be ruled out yet is the difference in initialized magnitude distributions between polarity transfer vs. weight transfer (Supplementary Figure C.6).

In sum, polarity transfer is superior to weight transfer in most of the scenarios we tested, and such a superiority is further secured by fixing the polarities throughout training. To a large extent, weight polarity alone, not weight per se, is enough to transfer knowledge between networks and across tasks. Giving the additional magnitude information to Frozen-Net can give some performance gain, but only when data and time is abundant; in all other scenarios (i.e., data-limited or time-limited), initializing Frozen-Net with stereotypical weight magnitudes could be detrimental to the learning performance.

5. What do we lose by setting weight polarity *a priori*?

Intelligent agents have limited resources in 1) data, 2) time, 3) space (number of hidden units or other network size parameters), and 4) power ($\propto$ time$\times$space, number of flops). In Sections 2, 3, and 4, we showed that fixing weight polarity helps to save on two of these resources, time and data, but with a condition – the polarity configuration must be adequately set such that the network can still represent the function. With fixed polarity, certain configurations will result in networks never being able to learn the task. What is the probability of such unfortunate events happening? We investigate this direction with our simulated task XOR-5D.

Assuming unlimited resources (i.e., perfect learning algorithm, unlimited data and time), we deduced the theoretical probability limit for a single-hidden-layer network to be able to represent XOR (and its high-dimensional variants with task-irrelevant higher dimensions) as a function of the

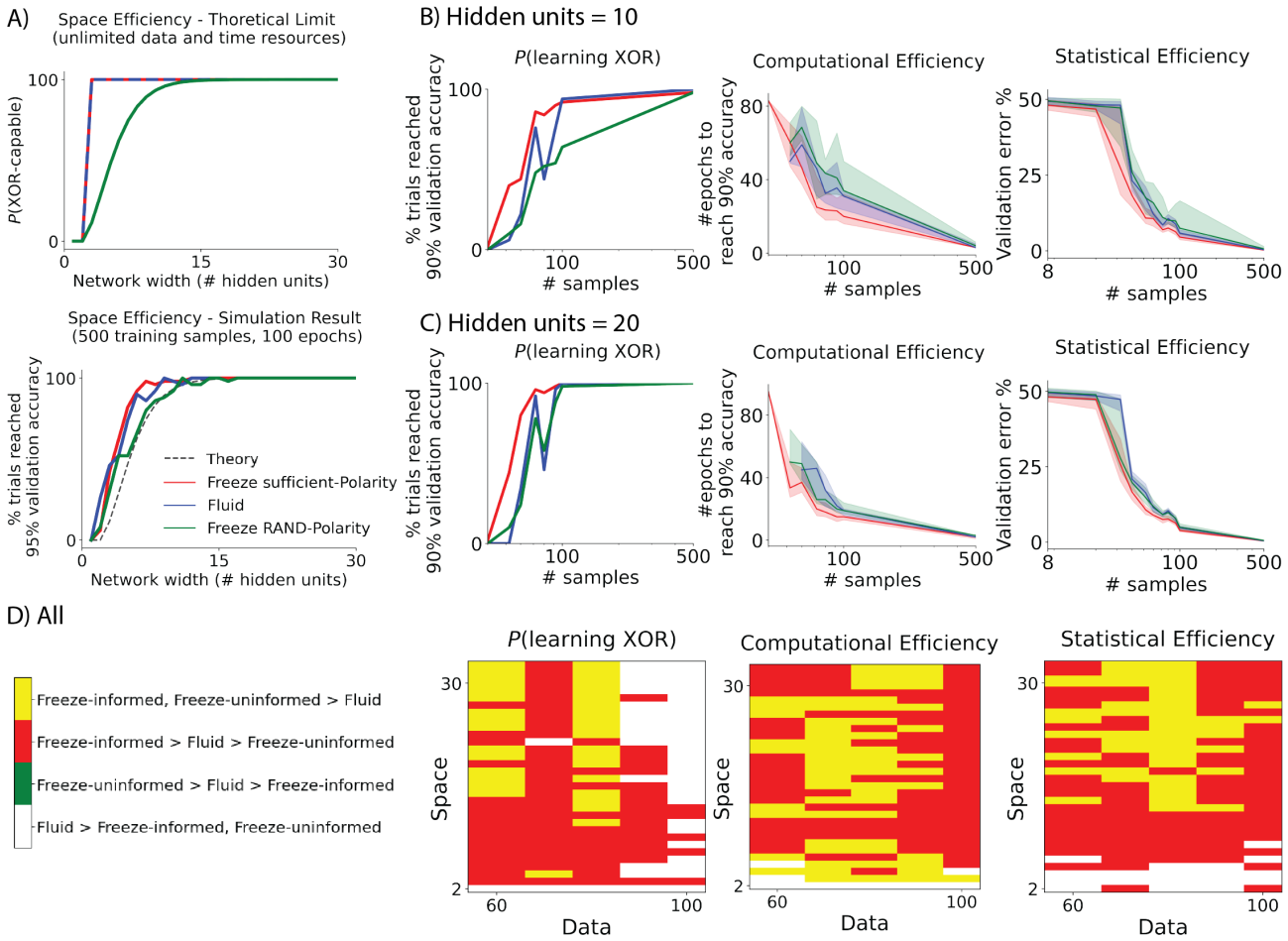

Figure 4. **A network has to be sufficiently large for it to learn more quickly, with less data, if its weight polarity is configured randomly.** A) For a randomly configured Frozen-Net to learn XOR-5D, it has to be sufficiently large. In both the theoretical limit and simulation results, it takes at least 15 hidden units for a Frozen-Net RAND-Polarity (green) to have $> 99\%$ chance of learning XOR-5D; it takes three hidden units, in theory, for an adequately configured Frozen-Net (red) and Fluid-Net (blue) to learn XOR-5D. B) When a Frozen-Net RAND-Polarity is not large enough, it has a much lower chance of learning XOR-5D, and when it does learn, it uses more time and data. C) When Frozen-Net RAND-Polarity is sufficiently large, it shows at least the same level of performance as standard Fluid-Nets. D) Frozen-Net Sufficient-Polarity shows an advantage over Fluid-Net (red + yellow) in almost all training sample sizes (especially when data is limited) and across all network sizes. Randomly configured Frozen-Nets have an advantage over Fluid-Nets (green + yellow), mostly when the network is sufficiently sized (> 15 hidden units). We ran 50 trials for all experiments. All curves represent medians with shaded regions denoting 25th-75th percentiles.

number of hidden units (Supplementary Theorem D.3). The theoretical results are plotted in Figure 4 panel A (top). In theory, it takes at least 3 units for Fluid-Net and Frozen-Net Sufficient-Polarity to be able to learn XOR-5D on every single trial. Such a probability will never be 100% for Frozen-Net RAND-Polarity, no matter how large the networks are. Luckily, the probability grows exponentially with the network size: having 15 hidden units is already sufficient for a randomly configured Frozen-Net to learn XOR with $> 99\%$ probability.

This is exactly what we see in our simulation results (Figure 4 panel A bottom). For a generous amount of data (500 training samples) and learning time (100 epochs), the Frozen-Net RAND-Polarity curve nicely matches the theoretical curve - networks with more than 15 hidden units learn XOR-5D with very high probability, even though their polarities are fixed randomly.

Both theory and simulation results show that we will lose all advantages if weight polarities are fixed but not configured adequately; an example of such is a small, randomly configured Frozen-Net (e.g., 10 hidden units, Figure 4 panel B). Notice that for the same network size, if we ensure the configuration is adequately set (red), then the network learns quickly and is data-efficient. By allowing more space (net-

work size), Frozen-Net RAND-Polarity starts to pick up the advantages in time and data efficiency (Figure 4 panel C). In summary, we gain from setting weight polarity *a priori* only if the polarity is configured adequately (Figure 4 panel D); the adequacy can either be made more probable by having a larger network or can be guaranteed by an explicit configuration rule (e.g., through development for NIs, or explicit rules in our simulations, or transferred polarities from previously trained networks).

6. Discussion

We showed in this paper that 1) if the weight polarities are adequately set *a priori*, then networks can learn with less data and time; 2) polarity may be a more effective and compressed medium of network knowledge transfer; and 3) while we lose all advantages when the polarities are fixed but not set adequately, we can regain these advantages by increasing the size of the networks. Below we discuss the novelty of our work and some future directions.

Transfer Learning In the transfer learning literature, to the best of our knowledge, there has been no indication that transferring polarity alone is sufficient for knowledge transfer – ours is the first demonstration of such sufficiency. Indeed, in a statistical learning framework, it is counter-intuitive to disassemble a single parameter into magnitude and sign. Previous work that vaguely took on a connectionist's view mostly focused on the existence or range of connections (e.g., adding skip connections (He et al., 2016)), but the polarity of such connections were essentially left out of the discussion. Our work broke the stereotypical view and studies weight polarity as an important factor all by itself. A lesson we learned from this research is that when designing network architectures, we should not only focus on the *existence* of connections, but also pay attention to the *polarities* of connections.

Lottery Ticket Hypothesis Our work also agrees with the results in the lottery ticket hypothesis (LTH) literature. LTH states that for a large, randomly initialized neural network, there exists small sub-networks (lottery tickets) that can be trained in isolation to match the test accuracy of the original large network (Frankle & Carbin, 2019). It has been proven that such lottery tickets exist with high probability provided the large networks are wide enough (Burkholz et al., 2022). In our work, instead of pruning connections, we pruned half of the polarity for all connections by randomly initializing and fixing polarities. Similar to LTH theory, we also observe that larger networks enjoy a high probability of learning XOR-5D (sec 5). Interestingly, it has been specifically shown that polarity is important in the pruning process Zhou et al. (2019). In essence, Zhou et al. (2019) experimentally showed that lottery tickets can no longer reach the original test accuracy if their polarity information is randomly initialized during the pruning process. Our study agrees with them that polarity patterns contain important task-related information and should be properly initialized. Our work differs from Zhou et al. (2019) in key ways. Most fundamentally, Zhou et al. (2019) demonstrate that properly presetting a subset of the initial polarity pattern is sufficient for a subnetwork to **learn at all**; we demonstrate that an adequate set of polarity patterns are sufficient for a network to **learn and transfer efficiently**. Furthermore, results in Zhou et al. (2019) can only demonstrate that polarity alone contains sufficient information to transfer knowledge across networks within **a given task**; we show that polarity contains sufficient information to transfer knowledge across **different tasks** (Sec 4).

Constrained Weight Distribution In the computational neuroscience literature, the discussion on sign-constrained weights is done under the frame of constraining weight distributions. The discussion focuses on the memory capacity of recurrent neural networks, where the memory capacity is measured using random input-output pattern associations (Amit et al., 1989; Brunel et al., 2004). A more recent treatment of the topic of constrained weight distribution and capacity is done on a single-hidden-unit network and the network capacity is measured by packing number (Zhong et al., 2022). None of the existing work can yet relate weight distribution constraints to representation capacity of general feedforward neural networks in general task settings. This is an interesting line of work that may be useful to define the limitations of setting weight polarity *a priori* in more general task settings.

Bio-Plausible AI In the literature of bio-plausible artificial neural networks (ANNs), the most related work is on Dale's principle: a single unit's output weights are exclusively excitatory or inhibitory. An exciting attempt at applying such a principle to ANNs achieved performance comparable to multi-layer perceptrons (gray-scale image classification tasks) and VGG16 (CIFAR-10) (Cornford et al., 2021). Our approach differs in several ways; the most fundamental one is ours does *not* require exclusivity of a unit's weight polarity, we only ask the polarity configuration to stay fixed throughout training. Because we made fewer assumptions on the architecture and network properties, we were able to reveal the true power of weight polarities - polarity-fixed networks can not only perform as well as the traditional approaches when the polarities are set adequately, but they can also learn more quickly with smaller training samples. Additionally, we revealed that polarities, not weights, may be a more effective and compressed knowledge transfer medium. Furthermore, our Freeze-SGD Algorithm 1 is easily applicable to any existing network architecture and any learning mode, be it transfer learning or de novo learning,

thus enjoying a wider application range.

Realization of the Bias-Variance Trade-off Our work is a realization of learning theory. Our way of explicitly setting the weight polarities provides a strong prior for the networks. This is in line with the bias-variance trade-off theorem (Geman et al., 1992; Tibshirani & Friedman, 2001). The Frozen-Net sufficient-Polarity strategy has a high inductive bias – by trading representation capacity, they gain in time and data efficiency through discounting parameter variances. Indeed, the performance gain we observe from fixing polarities might be explained by preventing over-fitting and thus achieving better generalization to the validation data.

Future Direction: Frozen-Polarity-Compatible Learning Algorithm We engineered our Freeze-SGD algorithm entirely based on SGD because we are interested in the pure effect of fixing polarity during learning. As discussed in Section 2, such an approach intrinsically put Frozen-Net in a disadvantageous position by resetting weights to the correct polarity after each learning iteration, effectively fighting against the gradient update direction. As quantified in Supplementary Figure C.5, we indeed observe polarities are dynamic throughout the learning process for a Fluid network trained *de novo*. It is an interesting next step to adopt a more polarity-freeze-compatible learning algorithm, possibly allowing us to further improve learning performance. One possibility is adapting the primal-dual interior-point method (Vogelstein et al., 2010) or Hebbian learning (Amit, 2019), as well as a host of bio-plausible learning algorithms (Miconi, 2017; Boopathy & Fiete, 2022; Dellaferrera & Kreiman, 2022).

Future Direction: Theories and Beyond In this work, we showed experimental evidence (simulation and image classification) that if the weight polarities are adequately set *a priori*, then networks can learn with less data and time. The experimental results are interesting yet largely lack theoretical explanations. As a future direction, we are planning to work on theoretically proving polarity, compared to weights, to be a more efficient knowledge transfer medium. Our demonstration that polarity is potentially all you need to transfer knowledge across networks and tasks was done in two settings: ImageNet to Fashion-MNIST and ImageNet to CIFAR-10. It is an important next step to test more exhaustively in different transfer settings; it is especially interesting to apply a 'stress test' to the polarity transfer idea and calibrate how dissimilar two tasks have to be for polarity transfer to not work well. We also look forward to applying our Frozen-Net approach to more tasks to see if we can empirically extend our results to more diverse scenarios. This paper is a first step toward the development of supporting theory and diverse experimental tests, potentially illuminating the structure of the hypothesized "innate circuit" that enables NIs to learn rapidly.

Software and Data

Code for running all experiments and analysis may be found on GitHub. All data presented in the paper may be found here.

Acknowledgements

This work is supported by grants awarded to J.T.V: NSF CAREER Award (Grant No. 1942963), NSF NeuroNex Award (Grant No. 2014862), NSF AI Institute Planning Award (Grant No. 2020312), and NSF Collaborative Research: THEORI Networks (Grant No. 2031985). Q.W. was also partially supported by Johns Hopkins neuroscience T32 training grant (Grant no. 091018). The authors would like to thank Dr. Carey Priebe, Dr. Jeremias Sulam, and Dr. Konrad Kording for the helpful discussion and direction to related literatures, the NeuroData lab members for helpful feedback.

References

Amit, D., Wong, K. Y. M., and Campbell, C. Perceptron learning with sign-constrained weights. *Journal of Physics A: Mathematical and General*, 22(12), 1989. ISSN 13616447. doi: 10.1088/0305-4470/22/12/009.

Amit, Y. Deep learning with asymmetric connections and hebbian updates. *Frontiers in Computational Neuroscience*, 13:18, 2 2019. ISSN 16625188. doi: 10.3389/FNCOM.2019.00018/BIBTEX.

Bendou, Y., Hu, Y., Lafargue, R., Lioi, G., Pasdeloup, B., Pateux, S., and Gripon, V. Easy: Ensemble augmented-shot y-shaped learning: State-of-the-art few-shot classification with simple ingredients. *arXiv*, 1 2022. doi: 10.48550/arxiv.2201.09699. URL http://arxiv.org/abs/2201.09699.

Boopathy, A. and Fiete, I. How to train your wide neural network without backprop: An input-weight alignment perspective. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S. (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 2178–2205. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/boopathy22a.html.

Brunel, N., Hakim, V., Isope, P., Nadal, J. P., and Barbour, B. Optimal information storage and the distribution of synaptic weights: Perceptron versus Purkinje cell. *Neuron*, 43(5), 2004. ISSN 08966273. doi: 10.1016/j.neuron.2004.08.023.

Burkholz, R., Laha, N., Mukherjee, R., and Gotovos, A. On the existence of universal lottery tickets. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=SYB4WrJql1n.

Cornford, J., Kalajdzievski, D., Leite, M., Lamarquette, A., Kullmann, D. M., and Richards, B. A. Learning to live with dale's principle: {ANN}s with separate excitatory and inhibitory units. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=eU776ZYxEpz.

Dellaferrera, G. and Kreiman, G. Error-driven input modulation: Solving the credit assignment problem without a backward pass. In Chaudhuri, K., Jegelka, S., Song, L., Szepesvari, C., Niu, G., and Sabato, S. (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 4937–4955. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/dellaferrera22a.html.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1126–1135. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/finn17a.html.

Frankle, J. and Carbin, M. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *7th International Conference on Learning Representations, ICLR 2019*, 2019. URL https://openreview.net/forum?id=rJl-b3RcF7.

Geman, S., Bienenstock, E., and Doursat, R. Neural networks and the bias/variance dilemma. *Neural Computation*, 4:1–58, 1 1992. ISSN 0899-7667. doi: 10.1162/neco.1992.4.1.1. URL https://doi.org/10.1162/neco.1992.4.1.1.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016. doi: 10.1109/CVPR.2016.90.

Hoffer, E. and Ailon, N. Deep metric learning using triplet network. In Feragen, A., Pelillo, M., and Loog, M. (eds.), *Similarity-Based Pattern Recognition*, pp. 84–92, Cham, 2015. Springer International Publishing. ISBN 978-3-319-24261-3.

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., Bridgland, A., Meyer, C., Kohl, S. A., Ballard, A. J., Cowie, A., Romera-Paredes, B., Nikolov, S., Jain, R., Adler, J., Back, T., Petersen, S., Reiman, D., Clancy, E., Zielinski, M., Steinegger, M., Pacholska, M., Berghammer, T., Bodenstein, S., Silver, D., Vinyals, O., Senior, A. W., Kavukcuoglu, K., Kohli, P., and Hassabis, D. Highly accurate protein structure prediction with alphafold. *Nature*, 596:583–589, 2021. ISSN 14764687. doi: 10.1038/s41586-021-03819-2. URL http://dx.doi.org/10.1038/s41586-021-03819-2.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60, 2017. ISSN 15577317. doi: 10.1145/3065386.

Lazarou, M., Stathaki, T., and Avrithis, Y. Iterative label cleaning for transductive and semi-supervised few-shot learning. *ICCV*, pp. 8731–8740, 2022. ISSN 15505499. doi: 10.1109/iccv48922.2021.00863.

Miconi, T. Biologically plausible learning in recurrent neural networks reproduces neural dynamics observed during cognitive tasks. *eLife*, 6, 2017. ISSN 2050084X. doi: 10.7554/eLife.20899.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., and Fei-Fei, L. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115:211–252, 2015. ISSN 15731405. doi: 10.1007/s11263-015-0816-y.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., Chen, Y., Lillicrap, T., Hui, F., Sifre, L., Driessche, G. V. D., Graepel, T., and Hassabis, D. Mastering the game of go without human knowledge. *Nature*, 550:354–359, 2017. ISSN 14764687. doi: 10.1038/nature24270. URL http://dx.doi.org/10.1038/nature24270.

Snell, J., Swersky, K., and Zemel, R. Prototypical networks for few-shot learning. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/cb8da6767461f2812ae4290eac7cbc42-Paper.pdf.

Spitzer, N. C. Neurotransmitter switching in the developing and adult brain. *Annual Review of Neuroscience*, 40, 2017. ISSN 15454126. doi: 10.1146/annurev-neuro-072116-031204.

Tibshirani, R. and Friedman, J. H. *The elements of statistical learning [electronic resource]: data mining, inference, and prediction: with 200 full-color illustrations*. Springer, 2001.

van der Spoel, E., Rozing, M. P., Houwing-Duistermaat, J. J., Slagboom, P. E., Beekman, M., de Craen, A. J. M., Westendorp, R. G. J., and van Heemst, D. Siamese neural networks for one-shot image recognition. *ICML - Deep Learning Workshop*, 7, 2015. ISSN 19454589.

Vinyals, O., Blundell, C., Lillicrap, T., kavukcuoglu, k., and Wierstra, D. Matching networks for one shot learning. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper/2016/file/90e1357833654983612fb05e3ec9148c-Paper.pdf.

Vogelstein, J. T., Packer, A. M., Machado, T. A., Sippy, T., Babadi, B., Yuste, R., and Paninski, L. Fast nonnegative deconvolution for spike train inference from population calcium imaging. *Journal of Neurophysiology*, 104:3691–3704, 12 2010. ISSN 00223077. doi: 10.1152/JN.01073.2009/ASSET/IMAGES/LARGE/Z9K0081002840010.JPEG. URL https://journals.physiology.org/doi/full/10.1152/jn.01073.2009.

Wang, Q., Powell, M. A., Geisa, A., Bridgeford, E., Priebe, C. E., and Vogelstein, J. T. Why do networks have inhibitory/negative connections?, 2023.

Zador, A. M. A critique of pure learning and what artificial neural networks can learn from animal brains. *Nature Communications*, 10, 2019. ISSN 20411723. doi: 10.1038/s41467-019-11786-6. URL http://dx.doi.org/10.1038/s41467-019-11786-6.

Zhong, W., Sorscher, B., Lee, D., and Sompolinsky, H. A theory of weight distribution-constrained learning. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 14113–14127. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/5b2db6dfda4d7362b2101b2d12dac029-Paper-Conference.pdf.

Zhou, H., Lan, J., Liu, R., and Yosinski, J. Deconstructing lottery tickets: Zeros, signs, and the supermask. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/1113d7a76ffceca1bb350bfe145467c6-Paper.pdf.

A. Freeze-SGD Algorithm

Algorithm 1 Freeze-SGD

for $l = 1, 2, \ldots, L$ **do**
  Get weight polarity template $T^{(l)}$ based on the configuration rules. Match $W^{(l)}$ to $T^{(l)}$.
end for

for $epoch = 1, 2, \ldots$ **do**
  for $batch = 1, 2, \ldots$ **do**
    SGD updates all weights.
    for $l = 1, 2, \ldots, L$ **do**
      Compare signs of the weights $W^{(l)}$ to the template $T^{(l)}$,
      get $checker = T^{(l)} * \text{sign}(W^{(l)})$.
      for (i, j) where $T^{(l)}(i, j) < 0$ **do**
        Make $W_l(i, j)$ in compliance with $T^{(l)}(i, j)$ by one of the following four ways:
        Case posCon $W^{(l)}(i, j) = T^{(l)}(i, j) * \epsilon$ where $\epsilon > 0$
        Case posRand $W^{(l)}(i, j) = T^{(l)}(i, j) * rand([0, \epsilon])$ where $\epsilon > 0$
        Case zero $W^{(l)}(i, j) = 0$
        Case flip $W^{(l)}(i, j) = -W^{(l)}(i, j)$
      end for
    end for
  end for
end for

B. Methods

XOR-5D Data were prepared by sampling from the XOR-5D distribution as described in the main text, the training data size varied, validation set is always 1000 samples across all scenarios. A single hidden layer network (64 hidden units for Figure 1, various for Figure 4) was trained for 100 epochs; in total 50 randomly seeded initializations (trials) were ran. To have the best controlled experiments, we fixed the magnitude distribution, architecture, training samples (n varies), learning rate, batch sequence of the training data, and the validation samples across all tested scenarios.

Image classification Experiments were done essentially the same as XOR-5D in a tightly controlled fashion. Across all scenarios, we controlled for the same magnitude distribution, batch sequence (batch size = 1000), learning rate, and training data. lr=0.001 was chosen from [0.1,0.03,0.01,0.001] based on the most accuracy gain after 50 epochs across all scenarios. Networks were trained for 100 epochs, 20 trials in total. We did not do any image augmentation.

Statistical efficiency was quantified by plotting validation error rate *at convergence* across different training data scarcity levels.

Computational efficiency was quantified by plotting the number of epochs to reach certain level of validation accuracy. We note not all trails could reach the same level of validation accuracy cut-off, to present the whole picture, we 1) plotted the percentage of trials that reached the validation accuracy cut-off (**success rate**); 2) showed results across a range of cut-off selections (Supplementary Figure C.2, C.3, C.4).

All experiments were run on 2 RTX-8000 GPUs.

C. Supplementary Figures

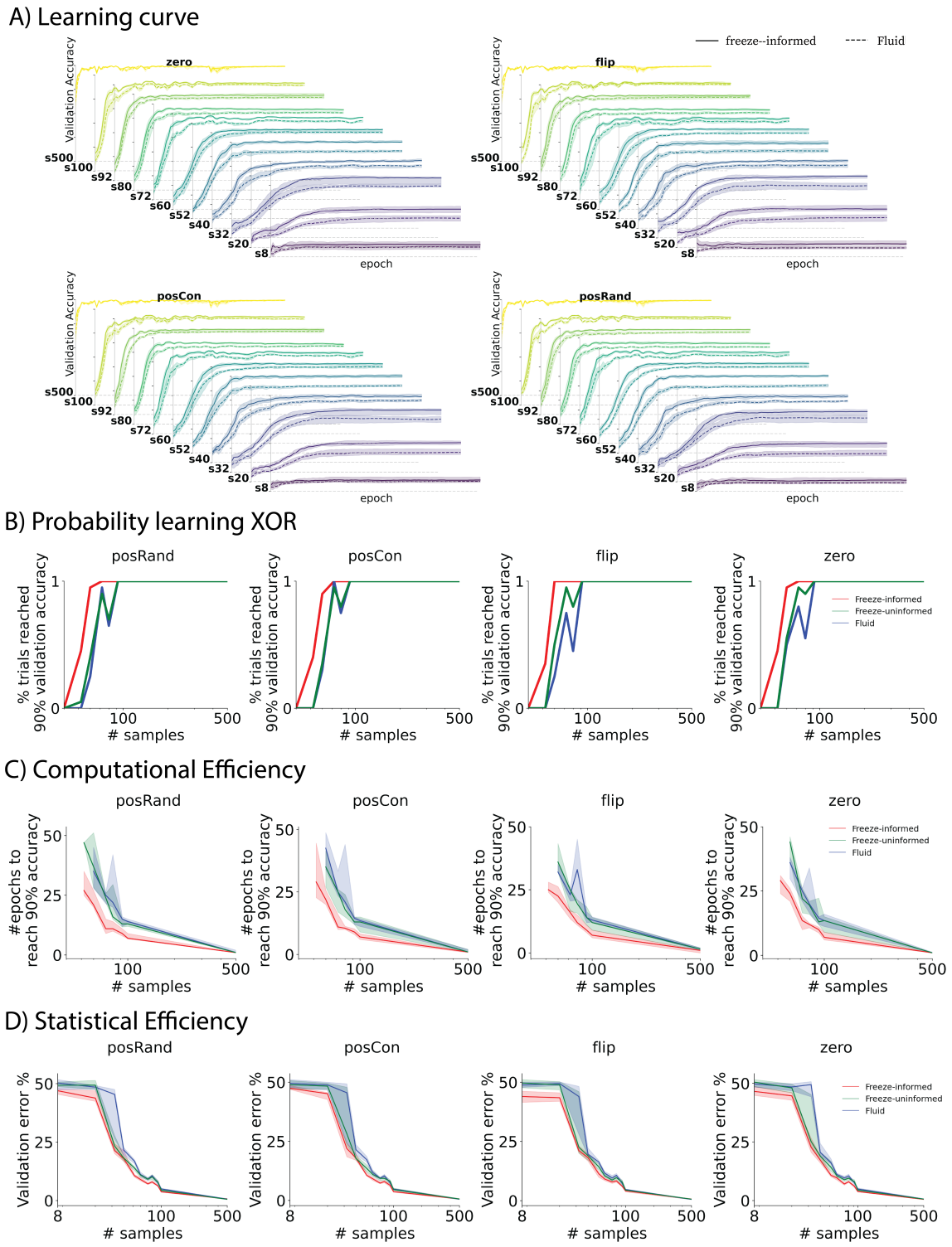

Figure C.1. **Different reset methods in the Freeze-SGD algorithm yield similar results.** The meaning of different reset methods is described in Algorithm 1. A) Freeze-informed consistently reaches higher validation accuracy after the same amount of training time. This is especially evident when training sample is scarce. B) Same curves as in Figure 4, panels B & C first column, plotted for different reset methods. C) & D) Same curves as in Figure 1, panels B & C, plotted for different reset methods. We run 20 trials for all experiments in this figure. All curves correspond to medians with shaded regions representing the 25th and 75th percentiles.

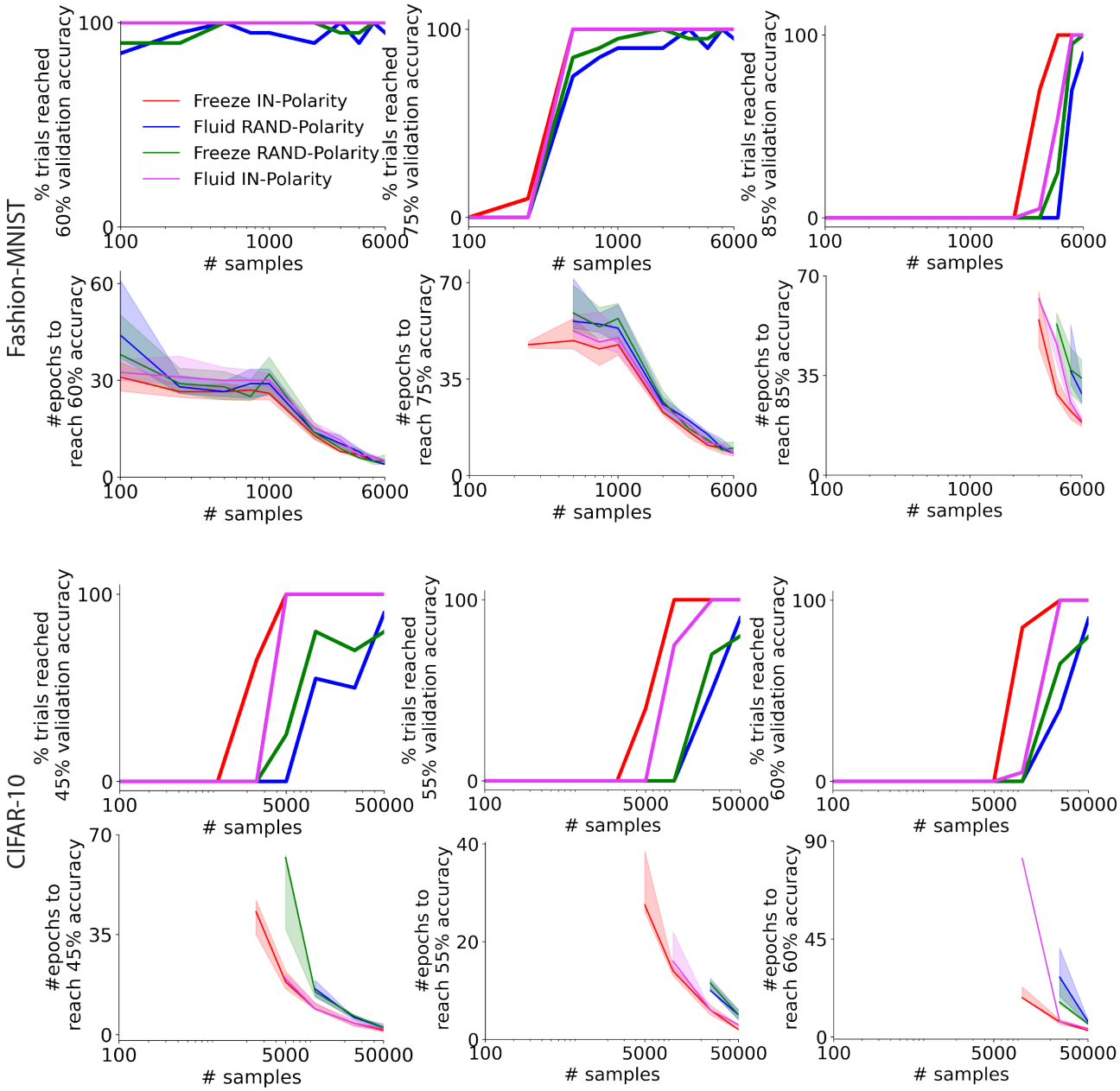

Figure C.2. **Related to Figure 2. Regardless of validation accuracy threshold, Frozen-Net IN-Polarity always learns more quickly.** Same curves as in Figure 2 right two columns, plotted for different validation accuracy thresholds.

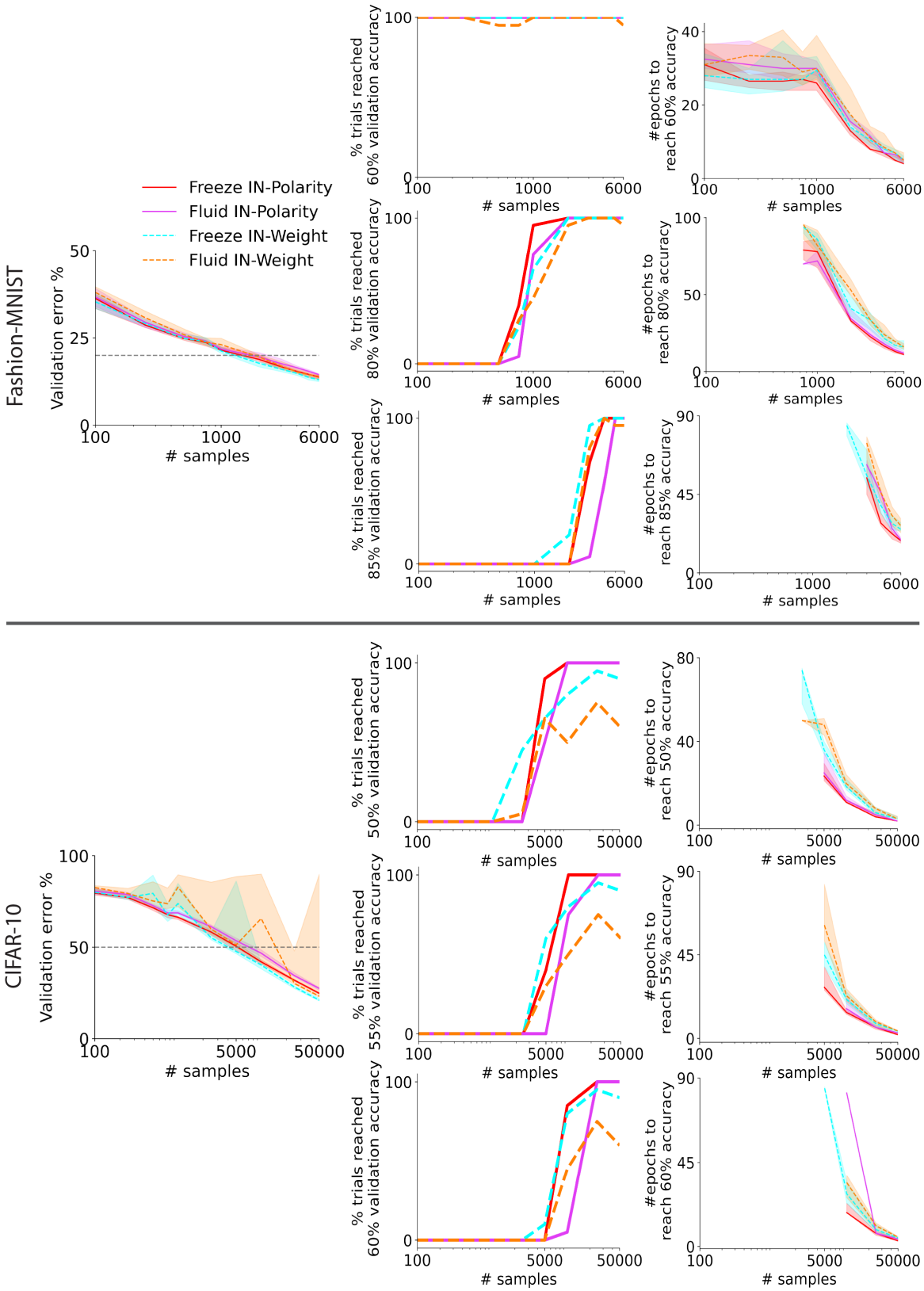

Figure C.3. **Related to Figure 3. All scenarios plotted separately instead of plotting only for the differences.**

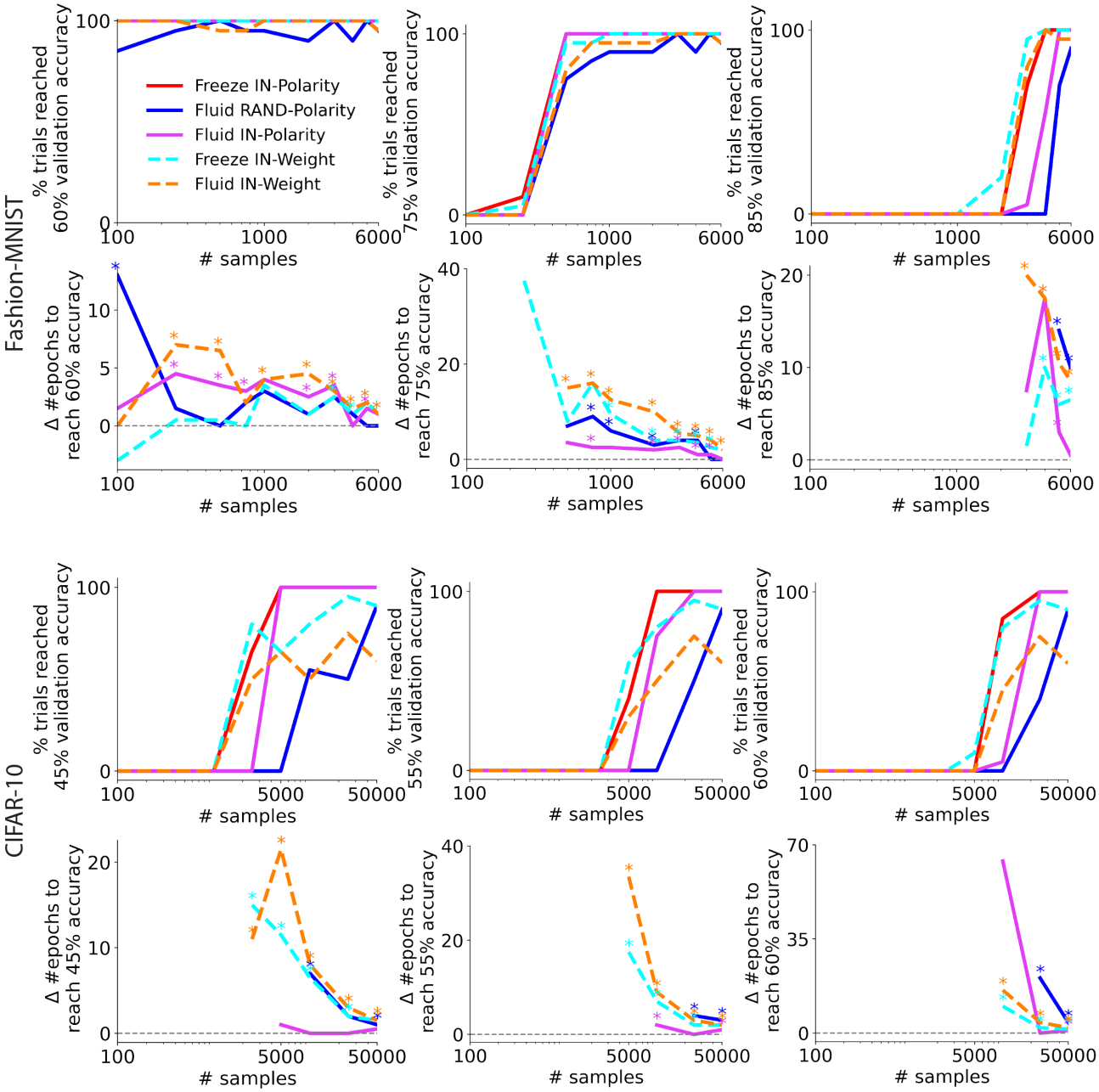

Figure C.4. **Related to Figure 3. Regardless of validation accuracy threshold, transferring and fixing polarities help networks learn faster than traditional weight transfer strategy.** Same curves as in Figure 3 right two columns, plotted for different validation accuracy thresholds.

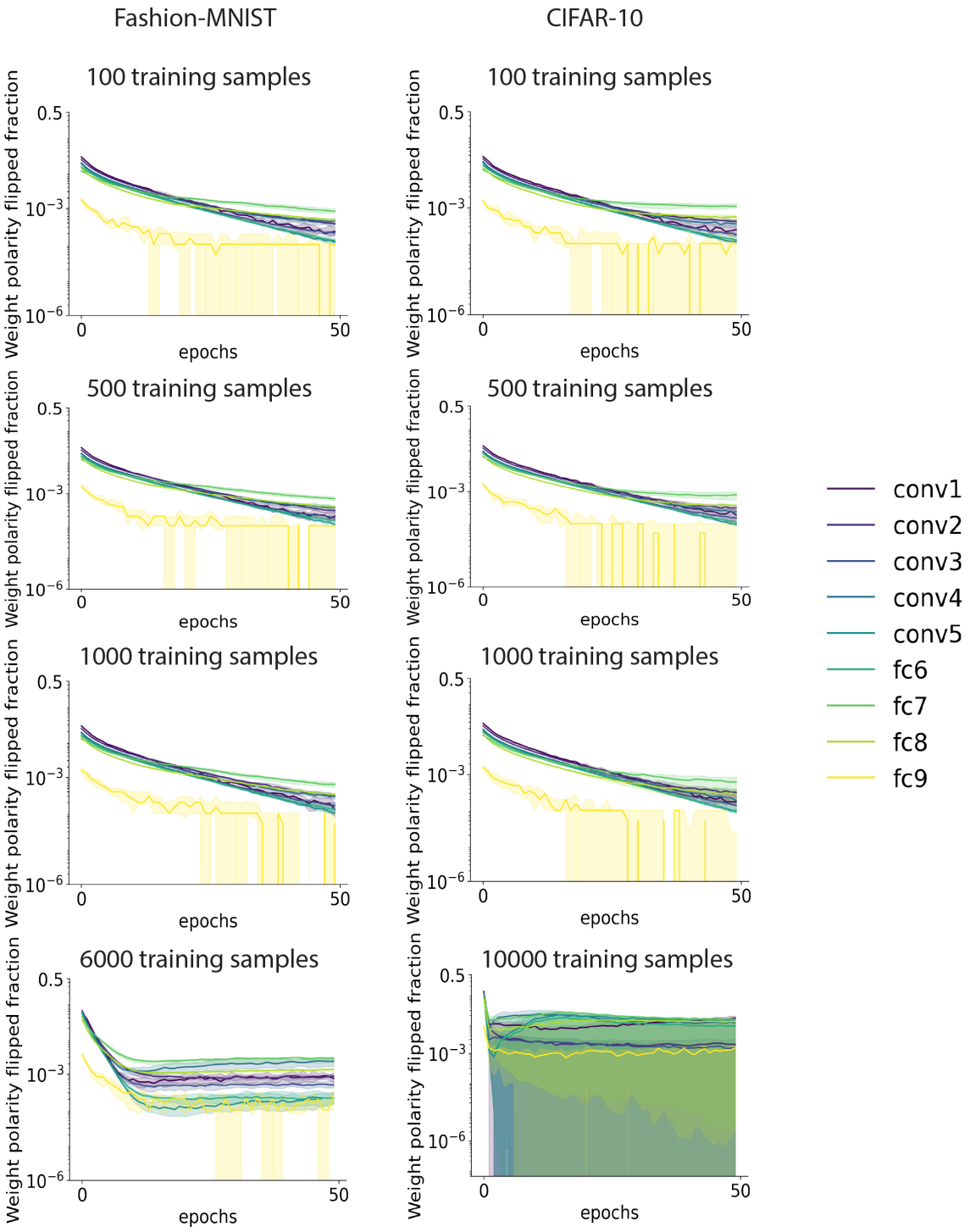

Figure C.5. **Tracking number of polarity flips.** Weight polarity flips were analyzed for Fluid RAND-Polarity (SGD with random initialization) and measured the ratio of weight parameters (excluding bias terms) that flipped sign between two consecutive epochs. The first 50 epochs were analyzed and plotted, separately across layers and training data size. Curves are median with shaded area representating the 25th and 75th percentiles out of the 20 trials.

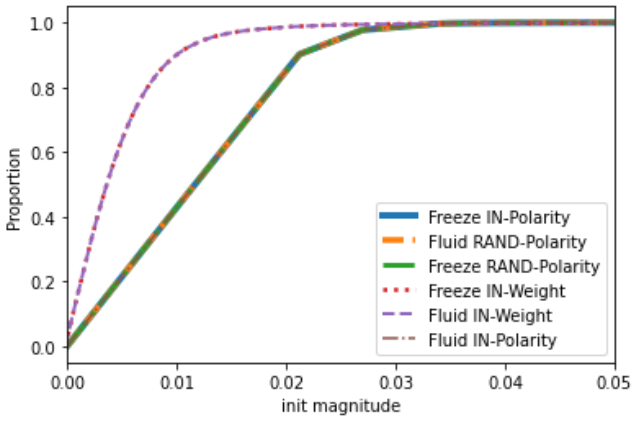

Figure C.6. **Weight magnitude distribution at initialization across all experimental conditions for image classification tasks.** All scenarios follow the exact same magnitude initialization magnitude except IN-weight transfer.

D. XOR Theorems related to Sec 5

Lemma D.1 (minimum XOR solution, 2D)**.** *For any single-hidden-layer network to be able to solve XOR, it is sufficient to have 3 different XOR-compatible hidden units. The weight pattern for each unit can be given by a triplet* $(w^{(1)}_{1,j}, w^{(1)}_{2,j}, w^{(2)}_{j,1})$*, where* j *is the unit index,* $j \in \{1, \ldots, n\}$*,* n *is the number of hidden units. An XOR compatible hidden unit is one where the weight pattern triplet have their polarities satisfy* $sign(w^{(2)}_{j,1}) = sign(w^{(1)}_{1,j}) \times sign(w^{(1)}_{2,j})$*.*

See proof on page 20.

Lemma D.2 (can learn XOR probability, 2D)**.** *For a single hidden layer network with* n *hidden units (*$n \in \mathbb{Z}^+$*), randomly initialize each weight with its polarity following* $P(polarity) \sim Bernoulli(0.5)$*, then the probability that this randomly initialized network can learn XOR without changing any of the weight polarity is lower bounded by*

$$\Omega(P(n)) = 1 - \frac{\sum_{k=0}^{2} Q(k,4,4,n)}{8^n}, \quad n \in \mathbb{Z}^+$$

$$Q(k,m,M,n) = [(M+k)^n - \sum_{p=0}^{k-1} Q(p,k,M,n)]\binom{m}{k}$$

Q here is a helper function for counting, it is defined in more detail in the proof.

See proof on page 20.

Theorem D.3 (can learn XOR probability, high dimensional)**.** *For a single hidden layer network with* n *hidden units, randomly initialize each weight with its polarity following* $P(polarity) \sim Bernoulli(0.5)$*, then the probability that this randomly initialized network can learn high-dimensional XOR (first two dimensions are relevant, the rest* $(d-2)$ *dimensions are irrelevant) without changing any of the weight polarities is lower bounded by*

$$\Omega(P(n,d)) = 1 - \frac{\sum_{k=0}^{2} Q(k,2^d,2^d,n)}{(2^{d+1})^n}, \quad n,d \in \mathbb{Z}^+, d \geq 2$$

$$Q(k,m,M,n) = [(M+k)^n - \sum_{p=0}^{k-1} Q(p,k,M,n)]\binom{m}{k}$$

Q here is a helper function for counting, it is defined in more detail in the proof.

See proof on page 21.

E. Proofs

Lemma 2.1 (capacity-speed trade-off)**.** *If the weight polarities are set a priori, such that the function is still representable, then the network can learn faster.*

Proof of Lemma 2.1. A feedforward DNN can be described as a graph

$$G = (V, E), w : E \to \mathbb{D}.$$

Nodes of the graph correspond to neurons (units), where each neuron is a function $\sigma^{(l)}_j(x) = \sigma(W^{(l)}_j x + b^{(l)}), j \in [n_l]$. All the weights for the network take on values from the set $\mathbb{D} = \{-d, \ldots, 0, \ldots, d\}$ for some $d \in \mathbb{N}$. The network is organized in *layers*. That is, the set of nodes can be decomposed into a union of (nonempty) disjoint subsets, $V = \dot{\cup}_{l=0}^{L} V_l$, such that every edge in E connects some node in V_{l-1} to some node in V_l, for some $l \in [L]$. Assume we have fully connected layers. Then the number of incoming edges per node is $|V_{l-1}|$. Let $|V_0|$ be the input space dimensionality. All nodes in V_0 (input layer) and V_L (output layer) are distinct.
Let $|G|$ denote the total number of distinct weight patterns of the graph G. Then for a single hidden layer network where $L = 2$, we have:

$$|G| = |\mathbb{D}|^{|E|} = |\mathbb{D}|^{(|V_0|*|V_1|+|V_1|*|V_2|)} \tag{1}$$

Then

$$\frac{|G|}{|G_{polarityFrozen}|} = (\frac{2d+1}{d+1})^{(|V_0|*|V_1|+|V_1|*|V_2|)} \tag{2}$$

Assume a different weight pattern represents a different function (trivially holds in linear + full rank weight case), then for every single representable function, there always exists a set of weight polarity configurations $G_{correct}$ such that $G_{correct} \subseteq G_{polarityFrozen}$, therefore

$$\frac{|G_{correct}|}{|G_{\text{polarityFrozen}}|} \ll \frac{|G_{correct}|}{|G|} \tag{3}$$

This means setting weight polarity *a priori* in an adequate way constraints the combinatorial search space to have much higher proportion of correct solutions, hence easier to learn under exhaustive search algorithm (Lemma 2.1). □

Lemma D.1 (minimum XOR solution, 2D)**.** *For any single-hidden-layer network to be able to solve XOR, it is sufficient to have 3 different XOR-compatible hidden units. The weight pattern for each unit can be given by a triplet* $(w^{(1)}_{1,j}, w^{(1)}_{2,j}, w^{(2)}_{j,1})$*, where* j *is the unit index,* $j \in \{1, \ldots, n\}$*,* n *is the number of hidden units. An XOR compatible hidden unit is one where the weight pattern triplet have their polarities satisfy* $sign(w^{(2)}_{j,1}) = sign(w^{(1)}_{1,j}) \times sign(w^{(1)}_{2,j})$*.*

Proof of Lemma D.1. For each hidden unit, we can enumerate all of its 8 possible weight polarity configurations, with the index set $A_{Polarities} = \{1, 2, 3, 4, 5, 6, 7, 8\}$:

#	$w^{(1)}_{1,j}$	$w^{(1)}_{2,j}$	$w^{(2)}_{j,1}$
1	+	+	+
2	+	+	-
3	+	-	+
4	+	-	-
5	-	+	+
6	-	+	-
7	-	-	+
8	-	-	-

The set $B_{XOR-Polarities} = \{2, 3, 5, 8\} \subset A_{Polarities}$ contains indexes of polarity patterns that follow XOR. To prove Lemma D.1, we first consider the case of 3 hidden unit single layer network. To prove a network of certain polarity pattern is capable of solving XOR, it suffices to give a working solution. Below, we exhaust all possible 3-unit network that satisfy the statement of Lemma D.1, i.e. having 3 different XOR-compatible units; and list out their corresponding working network solutions.

{}	#	$w^{(1)}_{1,j}$	$w^{(1)}_{2,j}$	$b^{(1)}_{j}$	$w^{(2)}_{j,1}$	working network $F(x, y)$=
{2,3,5}	2	+1	+1	0	-1	sigmoid$(-\sigma(x+y)+\sigma(x)+\sigma(y))$
	3	1	0	0	1	
	5	0	1	0	1	
{2,3,8}	2	+1	0	0	-1	sigmoid$(-\sigma(x)+\sigma(x-y)-\sigma(-y))$
	3	+1	-1	0	+1	
	8	0	-1	0	-1	
{2,5,8}	2	0	+1	0	-1	sigmoid$(-\sigma(y)+\sigma(y-x)-\sigma(-x))$
	5	-1	+1	0	+1	
	8	-1	0	0	-1	
{3,5,8}	3	0	-1	0	+1	sigmoid$(\sigma(-y)+\sigma(-x)-\sigma(-x-y))$
	5	-1	0	0	+1	
	8	-1	-1	0	-1	

For the case that single-hidden-layer network has more than 3 hidden units, if it satisfies the rule in Lemma D.1, we can always construct a XOR-solvable network solution by setting all other weights to zero except for 3 different XOR-compatible units, then we arrive at one of the four situations in the above table and we just proved they are XOR solutions.

Therefore, for any single hidden layer network to solve XOR, it is sufficient to have at least 3 of the 4 XOR polarity patterned units. □

Lemma D.2 (can learn XOR probability, 2D)**.** *For a single hidden layer network with n hidden units ($n \in \mathbb{Z}^+$), randomly initialize each weight with its polarity following $P(polarity) \sim Bernoulli(0.5)$, then the probability that this randomly initialized network can learn XOR without changing any of the weight polarity is lower bounded by*

$$\Omega(P(n)) = 1 - \frac{\sum_{k=0}^{2} Q(k, 4, 4, n)}{8^n}, \quad n \in \mathbb{Z}^+$$

$$Q(k, m, M, n) = [(M+k)^n - \sum_{p=0}^{k-1} Q(p, k, M, n)]\binom{m}{k}$$

Q here is a helper function for counting, it is defined in more detail in the proof.

Proof of Lemma D.2. For any network randomly initialized, its weight polarity pattern is a set of n draws with replacement from the polarities indexed by the set $A_{Polarities}$, $|A_{Polarities}| = 8$. Define $H = (h_1, \ldots, h_m, \ldots, h_n), h_m \in A_{Polarities}, |H| = n$, to be the tuple of the indices of the observed weight patterns for the hidden layer. h_m is the index of the weight polarity pattern for unit m. For any network to be able to solve XOR, it needs to have at least 3 units whose weight patterns are distinct members of set $B_{XOR-Polarities}$ (Lemma D.1). That is, let $J = \{h_m : h_m \in H, h_m \in B_{XOR-Polarities}\}$. Then we need that $|J| \geq 3$. We can define the probability of having exactly k of the 4 XOR compatible weight patterns present within the hidden layer as following (for brevity, $A = A_{Polarities}, B = B_{XOR-Polarities}$):
None of the members in set B appears is given by:

$$P(|J| = 0) = \frac{|B|^n}{|A|^n} = (\frac{4}{8})^n = (\frac{1}{2})^n$$

Only one of the member in set B appeared in H, and that member can appear more than once in H is given by:

$$P(|J| = 1) = \frac{\binom{|B|}{1}((|B|+1)^n - |B|^n)}{|A|^n} = \frac{\binom{4}{1}(5^n - 4^n)}{8^n}$$

This is explained by choosing one of 4 members of B $\binom{|B|}{1}$, then multiply by the chosen member appears at least once $((|B|+1)^n - |B|^n)$

Only two of the members in set B appeared in H, and both can appear more than once in H:

$$P(|J| = 2) = \frac{\binom{|B|}{2}((|B|+2)^n - \binom{2}{1}((|B|+1)^n - |B|^n) - |B|^n)}{|A|^n}$$
$$= \frac{\binom{4}{2}(6^n - \binom{2}{1}(5^n - 4^n) - 4^n)}{8^n}$$

To count two members appear at least once in H, we have $\binom{|B|}{2}$ ways of choosing the 2 members from set B, and there are $(|B|+2)^n$ ways of choosing with replacement to populate the tuple H, where each hidden unit can choose from in total $(|B|+2)$ possible patterns, and subtract the

situation where only one of the two members appeared $\binom{2}{1}((|B|+1)^n - |B|^n)$ and the situation where neither appeared $|B|^n$
The above equations can be put into a compact form

$$P(|J| = k) = \frac{Q(k, |B|, |A| - |B|, n)}{|A|^n}$$

where $Q(k, m, M, n) = [(M + k)^n - \sum_{p=0}^{k-1} Q(p, k, M, n)]\binom{m}{k}, k \in \{0, \ldots, m\}, m \in 1, \ldots, |B|, M \in 1, \ldots, |A| - |B|, n \in \mathbb{Z}^+$ is a helper function that gives exactly k of the m different XOR-compatible polarity patterns appeared in n units, and in total $m + M$ options are considered for each unit.

Then, $P(n)$ is counting at least 3 of the 4 set B patterns appear:

$$\begin{aligned}
&\Omega(P(n)) \\
&= P(|J| \geq 3) \\
&= 1 - P(|J| = 0) - P(|J| = 1) - P(|J| = 2) \\
&= 1 - \frac{\sum_{k=0}^{2} Q(k, 4, 4, n)}{8^n} \\
&= 1 - (\frac{1}{2})^n - \frac{\binom{4}{1}(5^n - 4^n)}{8^n} - \frac{\binom{4}{2}(6^n - \binom{2}{1}(5^n - 4^n) - 4^n)}{8^n}.
\end{aligned}$$

□

Theorem D.3 (can learn XOR probability, high dimensional)**.** *For a single hidden layer network with n hidden units, randomly initialize each weight with its polarity following $P(polarity) \sim Bernoulli(0.5)$, then the probability that this randomly initialized network can learn high-dimensional XOR (first two dimensions are relevant, the rest $(d-2)$ dimensions are irrelevant) without changing any of the weight polarities is lower bounded by*

$$\Omega(P(n, d)) = 1 - \frac{\sum_{k=0}^{2} Q(k, 2^d, 2^d, n)}{(2^{d+1})^n}, \quad n, d \in \mathbb{Z}^+, d \geq 2$$

$$Q(k, m, M, n) = [(M + k)^n - \sum_{p=0}^{k-1} Q(p, k, M, n)]\binom{m}{k}$$

Q here is a helper function for counting, it is defined in more detail in the proof.

Proof of Theorem D.3. We solve Theorem D.3 with the exact same counting algorithm as in Lemma D.2, the only difference is now $|A_{Polarities}| = 2^{d+1}$ and $|B_{XOR-Polarities}| = 2^d$. We prove these two equalities below.

$|A_{Polarities}| = 2^{d+1}$ because we have d input weights and 1 output weight for each unit.

The weight pattern of a single hidden unit in this case is given by a tuple $(w_{1,j}^{(1)}, \ldots, w_{d,j}^{(1)}, w_{j,1}^{(2)}), j \in \{1, \ldots, n\}$ and n is the number of hidden units. Our conclusion in Lemma D.1 trivially extends to the high dimensional case. This is because high-dimensional-XOR-solvable network solutions can be trivially constructed from Lemma D.1 by setting the irrelevant input dimension weights to 0. We can restate Lemma D.1 for high-dimensional XOR as following:

For any single-hidden-layer network to be able to solve high dimensional XOR (only first two dimensions are relevant), it is sufficient to have 3 different XOR-compatible hidden units. A high dimensional XOR compatible unit polarity pattern is ruled by $sign(w_{j,1}^{(2)}) = sign(w_{1,j}^{(1)}) \times sign(w_{2,j}^{(1)})$.

Therefore the irrelevant dimension input weights can be of either polarities and there are 2^{d-2} different combinations of them. Therefore $|B_{XOR-Polarities}| = 4 \times 2^{d-2} = 2^d$.

Follow the steps in Lemma D.2, we have (for brevity, $A = A_{Polarities}, B = B_{XOR-Polarities}$)

$$\begin{aligned}
P(|J| = 0) &= \frac{|B|^n}{|A|^n} = (\frac{2^d}{2^{d+1}})^n \\
P(|J| = 1) &= \frac{\binom{|B|}{1}((|B|+1)^n - |B|^n)}{|A|^n} \\
&= \frac{\binom{2^d}{1}((2^d+1)^n - (2^d)^n)}{(2^{d+1})^n} \\
P(|J| = 2) &= \frac{\binom{|B|}{2}((|B|+2)^n - \binom{2}{1}((|B|+1)^n - |B|^n) - |B|^n)}{|A|^n} \\
&= \frac{\binom{2^d}{2}((2^d+2)^n - \binom{2}{1}((2^d+1)^n - (2^d)^n) - (2^d)^n)}{(2^{d+1})^n}
\end{aligned}$$

Therefore, we have

$$\begin{aligned}
\Omega(P(n, d)) =& 1 - \frac{\sum_{k=0}^{2} Q(k, 2^d, 2^d, n)}{(2^{d+1})^n} \\
=& 1 - (\frac{1}{2})^n \\
&- \frac{\binom{2^d}{1}((2^d+1)^n - (2^d)^n)}{(2^{d+1})^n} \\
&- \frac{\binom{2^d}{2}((2^d+2)^n - \binom{2}{1}((2^d+1)^n - (2^d)^n) - (2^d)^n)}{(2^{d+1})^n}, \\
& n, d \in \mathbb{Z}^+, d \geq 2
\end{aligned}$$

□