EFFICIENT KNOWLEDGE DISTILLATION VIA CURRICULUM EXTRACTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Knowledge distillation is a technique used to train a small student network using the output generated by a large teacher network, and has many empirical advantages (Hinton et al., 2015). While the standard one-shot approach to distillation only uses the output of the final teacher network, recent work (Panigrahi et al., 2024a) has shown that using intermediate checkpoints from the teacher's training process as an implicit "curriculum" for progressive distillation can significantly speed up training. However, such schemes require storing these checkpoints, and often require careful selection of the intermediate checkpoints to train on, which can be impractical for large-scale training.

In this paper, we show that a curriculum can be *extracted* from just the fully trained teacher network, and that this extracted curriculum can give similar efficiency benefits to those of progressive distillation. Our extraction scheme is natural; we use a random projection of the hidden representations of the teacher network to progressively train the student network, before training using the output of the full network. We show that our scheme significantly outperforms one-shot distillation and achieves a performance similar to that of progressive distillation for learning sparse parities with two-layer networks, and provide theoretical guarantees for this setting. Additionally, we show that our method outperforms one-shot distillation even when using transformer-based architectures, both for sparse-parity learning, and language modeling tasks.

1 INTRODUCTION

In the era of large-scale models, as the cost of training state-of-the-art models increases substantially with each passing year, leveraging compute effectively for training and inference has become increasingly important. Knowledge distillation (Hinton et al., 2015) is one popular technique that is commonly used to reduce the amount of compute necessary for inference, by training a small *student* network to mimic the output of a large teacher network. Indeed, several state-of-the-art language models are distilled versions of larger models (DeepSeek-AI et al., 2025; Abdin et al., 2024; Team, 2024; OpenAI, 2024).

Despite the adoption of distillation for language model training, prior work (Panigrahi et al., 2024a; Anil et al., 2018; Mirzadeh et al., 2020) has shown that just using the output of the fully-trained teacher network to train the student can result in poor performance relative to the teacher (a "teacher-student gap" in performance), and that *progressive distillation* can significantly improve the performance of the student. Panigrahi et al. (2024a) in particular offers an explanation for this phenomenon – the intermediate checkpoints during the training of the teacher network act as an implicit *curriculum* for the training of the student network, with earlier checkpoints emphasizing simpler patterns (e.g., local syntax in the case of language models), and later checkpoints capturing complex abstractions (e.g., long-range semantics). Please see Appendix B for a more detailed overview of related work.

While progressive distillation offers significant efficiency advantages over one-shot distillation, it requires storing frequent checkpoints during the training of the teacher, which can be prohibitive for modern LLMs. Deciding on which checkpoints to make use of to train the student is often unclear; such checkpoints are found by extensive experimentation in the works listed above, which can be impractical. Moreover, in many cases, one lacks access to intermediate checkpoints during training,

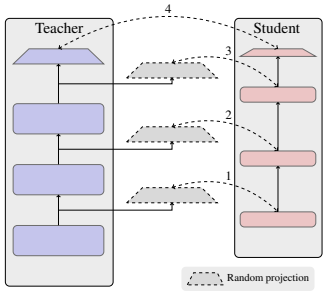

Figure 1: Our curriculum extraction method trains the student model in a layer-wise fashion. Student layers are sequentially aligned to a random projection of the corresponding teacher layer's hidden representation using the Mean Squared Error (MSE). After aligning layers, the student is trained on the teacher's output logits via the KL Divergence loss.

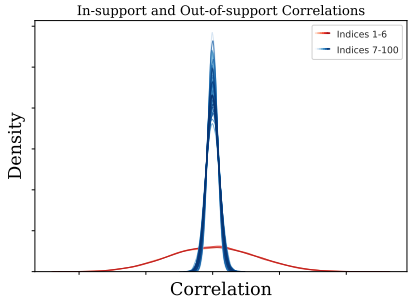

Figure 2: **In-support and out-of-support correlations.** A two-layer MLP trained on 100-dimensional 6-sparse parity data exhibits distinct in-support (red) and out-of-support (blue) correlations of $(Af_t^{(1)})(\mathbf{x})$ with x_j for the random projection $A \in \mathbb{R}^{1\times m_t}$. When j is in the support, the correlations show significantly larger standard deviations compared to when j is outside the support.

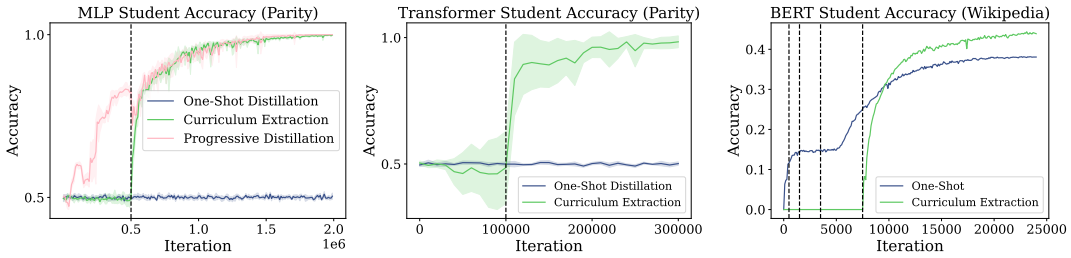

Figure 3: **Comparing Curriculum Extraction and One-Shot Distillation.** We show three tasks for which curriculum extraction outperforms one-shot distillation: (a) A two-layer MLP trained on 100-dimensional 6-sparse parity, with a teacher hidden dimension of 50k and a student hidden dimension of 100. (b) A transformer trained on 100-dimensional 6-sparse parity, using 256-dimensional embeddings, where the teacher has 32 attention heads and the student has 4. (c) A BERT-large model fine-tuned on the Wikipedia dataset, with the teacher using 768-dimensional embeddings, 12 attention heads, and 12 transformer blocks, while the student reduces embeddings to 256 dimensions and attention heads to 4. The dashed vertical lines indicate the iterations where the layer being distilled is changed in the case of curriculum extraction, and a change in teacher checkpoint in the case of progressive distillation.

even for open-source models (Jiang et al., 2023; DeepSeek-AI et al., 2025; Meta AI, 2024), making progressive distillation impossible.

This raises a natural question – can we design a scheme that maintains the advantages of progressive distillation without suffering from its drawbacks? Specifically, can we leverage the final fully-trained teacher model more effectively to train the student model efficiently?

Curriculum Extraction. We propose a scheme to *extract* a curriculum from the fully-trained teacher network. Our key insight is that the layer-wise hierarchy of a fully trained network naturally encodes a progression from simple to complex features. To operationalize this, we train the student's hidden layers sequentially on *random projections* of the teacher's hidden layers, starting from shallow (layer l, say) to deep (the final layer L), before training the full student network on the output of the full teacher network. See Figure 1 for a visual description of our extraction scheme.

By progressively training on projections from shallower to deeper layers, the student learns incrementally—mirroring the coarse-to-fine learning in progressive distillation, without having to store intermediate checkpoints. Beyond circumventing checkpoint storage, our approach can potentially be applied to efficiently distill from open-source models (e.g., Llama (Meta AI, 2024), Mistral (Jiang et al., 2023), and Deepseek models (DeepSeek-AI et al., 2025)), where only the final model is available. In addition, our method is computationally cheaper than one-shot or progressive distillation per training iteration – during early stages, only a subset of student and teacher layers are active, reducing memory and FLOPs per iteration.

1.1 Our Results

Sparse Parity Learning. We show that our curriculum extraction scheme is *significantly* more efficient than one-shot distillation for the task of learning sparse parities using a two-layer MLP, and provide a theoretical analysis for this setting. See Section 3 for a formal description of sparse parity learning and two-layer MLPs. Here, we state an informal version of our main theorem, with the formal theorem stated in Section 3.2.

Theorem 1.1 (Main, Informal)**.** *Consider learning d-dimensional k-sparse parity with a student model of size $\tilde{\Theta}(2^{O(k)})$, where $\tilde{O}, \tilde{\Theta}$ hides polylog factors in d, k. Suppose the teacher (of size $2^{O(k)}\mathrm{poly}(d,k)$) has a loss $O(\epsilon)$ for some small $\epsilon > 0$. Then, the total sample complexity needed for the student to reach ϵ-loss using curriculum extraction based on random projection is: $\tilde{\Theta}\left(2^{O(k)}\mathrm{poly}(d,k)\epsilon^{-2}\right)$. However, one-shot distillation requires at least $\Omega\left(d^{k-1}\epsilon^{-2}\right)$ samples.*

Thus, one-shot distillation *requires* $\Omega(d^{O(k)})$ samples to learn sparse parties, while our curriculum extraction scheme can learn using only $O(2^{O(k)}\mathrm{poly}(d))$ samples. We show in Figure 3 (a) that our curriculum extraction scheme significantly outperforms one-shot distillation empirically, as predicted by our theory – our scheme succeeds in learning, while one-shot distillation fails after training using $2 \cdot 10^6$ samples. Furthermore, it has similar performance as progressive distillation for a carefully chosen checkpoint – we choose the checkpoint during training of the teacher network whose output is most correlated with the support of the parity function, as proposed by Panigrahi et al. (2024b).

We also show empirically that our scheme continues to outperform one-shot distillation when using a transformer-based architecture for learning sparse parities in Figure 3 (b).

Masked Language Modeling (BERT). In addition to learning sparse parities, we empirically study our curriculum extraction scheme for language modeling, focusing on BERT-style *masked language modeling*. We study two settings with different kinds of data: (i) Synthetic data generated by a Probabilistic Context-free Grammar (PCFG), and (ii) Real-world language data from Wikipedia.

In the case of PCFGs, we show that our scheme outperforms one-shot distillation, both in terms of computational efficiency (number of FLOPs), and in terms of sample efficiency (number of iterations), in Figures 4 (a) and (b). We also show that our curriculum scheme outperforms just using the final hidden representation of the teacher to distill before distilling using the full network; this suggests that the efficiency benefits of our scheme do indeed come from the fact that the layers of the teacher network implicitly act as a curriculum, rather than merely from the increased dimensionality of the distilled features.

For Wikipedia data, we show in Figure 3 (c) that curriculum extraction has a significant accuracy advantage over one-shot distillation when training for $24 \cdot 10^3$ iterations with a batch size of 128 – for extraction, we use 4 intermediate layers to distill, before distilling with the full teacher network.

2 Curriculum Extraction

We now describe our curriculum extraction scheme formally.

Definition 2.1 (Curriculum Extraction Scheme)**.** *Given a pre-trained teacher network T and a student network S, both having the same number of layers L, let T_i and S_i denote the network up to layer i ($T_i : \mathbb{R}^d \to \mathbb{R}^{m_i}$ and $S_i : \mathbb{R}^d \to \mathbb{R}^{n_i}$). Suppose also that for some $\ell \in [0, L)$ we are given a sequence $\{t_\ell, \ldots, t_L\}$ such that $t_i \in \mathbb{Z}$ indicates the number of iterations we train S_i for. The Layer-Wise Curriculum Extraction Scheme proceeds as follows:*

1. ***Initialization:*** *Initialize the student network S with random weights.*
2. ***Layer-Wise Training:*** *For each $i \in [\ell, L-1]$, such that $t_i > 0$:*
   (a) *Define a random projection matrix $P_i : \mathbb{R}^{m_i} \to \mathbb{R}^{n_i}$ for layer i.*
   (b) *Train S_i for t_i iterations to reduce the MSE loss between $S_i(x)$ and $P_i(T_i(x))$: $\mathcal{L}_i = \frac{1}{T}\sum_{t=1}^{T} \|S_i(x^{(t)}) - P_i(T_i(x^{(t)}))\|_2^2$ where T is the number of training samples.*
3. *Train the entire student network S for t_L iterations to reduce the KL-divergence loss between $S(x)$ and $T(x)$.*

3 Learning Sparse Parities via Curriculum Extraction

To demonstrate the effectiveness of our curriculum extraction scheme, we study its performance in learning sparse parities. We compare our curriculum extraction scheme to one-shot distillation, where the student is trained directly on samples generated by the teacher.

3.1 Preliminaries

For our arguments in this section, we will assume WLOG that the support of the unknown parity is $S = [k]$. We will learn two-layer MLP networks of the form $f(\mathbf{x}) := \mathbf{a} \cdot \sigma(\mathbf{W}\mathbf{x} + \mathbf{b}) = \sum_{i=1}^{m} a_i \sigma(\mathbf{w}_i \cdot \mathbf{x} + b_i)$. where $\mathbf{x} \in \mathbb{R}^d, \mathbf{W} \in \mathbb{R}^{d \times m}$ and $\mathbf{b}, \mathbf{a} \in \mathbb{R}^m$, and $\sigma(t) := \max(0, t)$ is applied coordinate-wise when applied to a vector. We will denote the student network by f_s and the teacher network by f_t with hidden dimensions m_s and m_t respectively. In general $m_s \leq m_t$. Let $\ell_f(\mathbf{x}, y) := \max(0, 1 - f(\mathbf{x})y)$ be the hinge loss. Our main task will be to find the best fitting two-layer MLP to an unknown sparse parity function.

Problem 3.1 (Learning Sparse Parities)**.** Let $S \subset [d]$ with $|S| = k$ and $k < d$ denote the support of our unknown sparse parity. For $\mathbf{x} \in \{\pm 1\}^d$, we define $\chi_S(\mathbf{x}) := \prod_{i \in S} x_i$ to be a *sparse parity* supported on S. Given a tolerance $\epsilon \in \mathbb{R}$ and n samples $\{(\mathbf{x}_i, \chi_S(\mathbf{x}_i)) \mid \mathbf{x}_i \sim_{u.a.r} \{\pm 1\} \text{ for } i \in [0, n]\}$ for an unknown support S, the task of learning a sparse parity function using a two-layer MLP, is to find a two-layer MLP that achieves loss $\mathbf{E}_{\mathbf{x} \sim \{\pm 1\}}[\ell_f(\mathbf{x}, y)] \leq \epsilon$.

For our theoretical analysis, our training setup differs slightly from Definition 2.1 when it comes to our losses – we use the hinge loss to train the teacher as well as the student's top layer, and a correlation-based distillation loss (defined below) to train the student's hidden layer.

Initialization Prior to training, we will initialize the network using the following symmetric initialization from Barak et al. (2022).

Definition 3.2 (Symmetric Initialization)**.** *Let $f(\mathbf{x}) := \sum_{i=1}^{m} a_i \sigma(\mathbf{w}_i \cdot \mathbf{x} + b_i)$ be a two-layer MLP with input dimension d and hidden dimension m. For each $1 \leq i \leq m/2$, we initialize the parameters $\{\mathbf{w}_i\}_{i=1}^m$, $\{b_i\}_{i=1}^m$ and $\{a_i\}_{i=1}^m$ as follows: $\mathbf{w}_i \sim U(\{\pm 1\}^d)$, $b_i \sim U\left(\left\{-1 + \frac{1}{k}, \cdots, 1 - \frac{1}{k}\right\}\right)$, $a_i \sim U\left(\left\{\frac{\pm 1}{m}\right\}\right)$, $m/2 < i \leq m$ are set to $\mathbf{w}_i = -\mathbf{w}_{i-m/2}$, $b_i = b_{i-m/2}$, $a_i = -a_{i-m/2}$.*

Training Algorithms: We train the teacher network f_t by minimizing the hinge loss in two stages. In the first stage of training, we freeze the top layer weights ($\mathbf{a}$) and train the network with a regularized version of $\ell_{f_t}(\mathbf{x}, y)$, given by $\ell_{f_t}(\mathbf{x}, y) - \lambda\|\mathbf{W}\|^2$ updating only $\mathbf{W}, \mathbf{b}$. In the second stage of training, we freeze the bottom layer weights and biases $\mathbf{W}, \mathbf{b}$ and only update $\mathbf{a}$.

For the student network f_s, we define a *distillation loss* instead. Let $f_t^{(1)} := \sigma(\mathbf{W} \cdot \mathbf{x} + \mathbf{b}) : \mathbb{R}^d \to \mathbb{R}^{m_t}$ denote the output of the first layer of the teacher and suppose $A \in \mathbb{R}^{m_t \times m_s}$ is a random symmetric projection which mimics the initialization, i.e. for $i \leq m_t/2$, each $A_{ij} \sim U(\{\pm 1/m_t\})$ and for $i > m_t/2$, $A_{ij} = -A_{(i-m/2)\ j}$. In the first stage, the first layer of the student (i.e. $f_s^{(1)}(\mathbf{x}) := \sigma(\mathbf{W}_s\mathbf{x} + \mathbf{b})$) is trained using the a similarly regularized version of the following distillation loss: $\ell_{DL}(\mathbf{x}, f_s^{(1)}, Af_t^{(1)}) = -f_s^{(1)}(\mathbf{x}) \cdot (Af_t^{(1)}(\mathbf{x}))$, i.e. $\ell_{DL}(\mathbf{x}, f_s^{(1)}, Af_t^{(1)}) - \lambda\|\mathbf{W}\|^2$

The second layer of the student ($\mathbf{a}_s$) is then trained using the standard hinge loss.

3.2 THEORETICAL RESULTS

We prove that, compared to one-shot distillation—where the student needs at least $\Omega\left(d^{\min(2c,k-1)}\right)$ samples to learn the unknown sparse parity—our curriculum extraction method reduces this requirement to $\tilde{O}(2^{O(k)}\text{poly}(k,d))$. We state this formally below:

Theorem 3.3 (Curriculum Extraction Requires Fewer Samples)**.** *Suppose the teacher model (of size $O(2^{O(k)}\text{poly}(d,k))$) has been trained with 2-stage training in Algorithm 1, and achieves a loss of $O(d^{-c})$ for some constant $c \geq 1$ at the end of the second stage. Suppose we train a student model f_s of size $\tilde{m} = \tilde{\Theta}(2^k k)$ using the following two strategies:*

1. ***Random-projection curriculum extraction:*** *Train the first layer of the student with a random projection of the first layer of the teacher to the right output dimension, and then train the entire student network with the final teacher network.*
2. ***One-shot Distillation:*** *Train with the teacher network throughout.*

Then,

1. *Under our distillation scheme, the total sample complexity to reach a loss of ϵ with probability $1-\delta$ is $\Theta(2^{O(k)}\text{poly}(d,k)\epsilon^{-2}\log(k/\delta\epsilon))$.*
2. *The necessary sample complexity under distillation is at least $\Omega\left(d^{\min(2c,k-1)}\right)$.*

The key difference between the two is that, in one-shot distillation, the student must identify one of $\Omega(d^k)$ possible parity functions from scratch. In contrast, our scheme splits the learning into two phases: identifying the support and learning the final function. Initially, the gradients of the distillation loss guide the student in detecting the support of the sparse parity via the bottom layer. With the support identified, the student only needs to select from $O(2^k)$ possible parities.

3.2.1 PROOF OVERVIEW

Without loss of generality, suppose that the support of the unknown parity $S = [k]$. The lower bound on the sample complexity for one-shot distillation (Item 2 in Theorem 3.3) follows from the exact same item in the analogous result in Panigrahi et al. (2024a) (Theorem B.1). The rest of this section will focus on a high-level sketch of a sample complexity upper bound of $\tilde{O}(2^{O(k)}\text{poly}(k,d))$ for our curriculum extraction scheme (i.e. Item 1 in Theorem 3.3).

As stated earlier, our scheme aims to separate *support recovery* from *loss minimization*. By extracting the support of the unknown sparse parity from the teacher using (exponentially) fewer samples than what the teacher requires to learn the support, the student can focus on optimizing the top layer after recovering the support.

For the sake of illustration, consider the unknown sparse parity $\chi_{[k]}(\mathbf{x}) := \prod_{i=1}^{k} x_i : \{\pm 1\}^d \to \{\pm 1\}$. Our goal is to learn this function using a two-layer MLP. First, note that the parity function $\chi_{[k]}(\mathbf{x})$ can be *represented* by a reasonably sized two-layer MLP. Since $\chi_{[k]}(\mathbf{x})$ depends only on the sum $\sum_{i=1}^{k} x_i$ due to its symmetry in $x_1, \ldots, x_k$, it can be rewritten as $\chi_{[k]}(\mathbf{x}) = g\left(\sum_{i=1}^{k} x_i\right)$ for a univariate function $g(t)$ mapping integers between $-k$ and k to $\{\pm 1\}$. A two-layer MLP with $O(k)$ ReLU activations can approximate this function by constructing a piecewise linear function over the $2k+1$ possible values.

If the ambient dimension is $d \gg k$, then this neural network may be realized by simply setting the out-of-support coordinates of the weight vectors to be 0. Hence, if you know the parity apriori, it is easy to construct the two-layer MLP that represents it.

In fact, it is possible to approximate the unknown sparse parity when much less is known — Theorem 4 from Barak et al. (2022) shows that the k-sparse parity can be well-approximated by training only the top layer, given a hidden dimension of size $\tilde{\Omega}(2^k)$ and random weights, provided there is a gap between in-support and out-of-support variables of the bottom layer weights. We restate the version of this theorem from Panigrahi et al. (2024b) in Lemma D.3.

Returning to our setting, we want the student's hidden weights to meet the conditions in Lemma D.3. From Lemma D.2 (Panigrahi et al., 2024b; Barak et al., 2022), we know that the weights of the first teacher layer $f_t^{(1)}$ satisfy this condition. The first student layer $f_s^{(1)}$ can only obtain support information through the gradients of the distillation loss. It turns out that it is possible to select an appropriate regularization parameter (λ) such that the student's first layer weights become proportional to the gradients after the first step, hence it suffices to demonstrate such a gap between the magnitudes of the in-support and out-of-support coordinates of the gradient (Barak et al., 2022; Panigrahi et al., 2024a). Lemma E.1 shows that the gradient of the distillation loss with respect to $\mathbf{w}_i$ is proportional to $\mathbf{E}_{\mathbf{x}}[(Af_t^{(1)}(\mathbf{x}))_i\mathbf{x} + (Af_t^{(1)}(\mathbf{x}))_i \operatorname{Maj}(\mathbf{w}_i \odot \mathbf{x})\mathbf{x}]$, and so to estimate the gap between different gradient coordinates, it suffices to estimate the gap between the in-support and out-of-support coordinates of the vector above.

For the rest of this discussion, we will focus on the j-th coorindate of the first term $\mathbf{E}_{\mathbf{x}}[(Af_t^{(1)}(\mathbf{x}))_i x_j]$. It will turn out that the second term is controlled by the first term. Note that , $\mathbf{E}_{\mathbf{x}}[(Af_t^{(1)}(\mathbf{x}))_i x_j] = \sum_{\ell=1}^{m_s} A_{i\ell}\, \mathbf{E}_{\mathbf{x}}[\sigma(\mathbf{x}\cdot\mathbf{w}_\ell + b_\ell)x_j]$, which, after a rearrangement, can be viewed as a sum of scaled Rademacher random variables $\{A_{i\ell}\}_{\ell=1}^{m_s}$. Arguments from Panigrahi et al. (2024a) show that the scaling of these random variables is significantly larger for $j \in S$ rather than $j \notin S$ (see Figure 2 for the distributions of these variables in our trained network).

By applying anticoncentration and concentration inequalities for sums of Rademacher random variables to the terms above, we show that with reasonable probability over the randomness of the top-layer weights this variance gap (over randomness of A) translates to a gap in the coordinates of $\mathbf{E}_{\mathbf{x}}[(Af_t^{(1)}(\mathbf{x}))_i\mathbf{x}]$ for a given draw of A. We state an informal version of this below, the formal version of which is in Lemma E.3.

Lemma 3.4 (Correlation Gap (Informal))**.** *As long as* $m_t = 2^{O(k)}\mathrm{poly}(d,k)/\delta^2 \geq (m_s^2\, k^4 \log(d)^2/\delta^2)$*, with probability* $1-\delta$*, every coordinate* $i \in [m_s]$ *of the projected teacher network satisfies* $|\mathbf{E}_{\mathbf{x}}[(Af_t)_i(\mathbf{x})x_j]| > \tilde{\Omega}((m_t k)^{-1})$ *for all* j *in the support of the unknown sparse parity, and* $\max_{j>k} |\mathbf{E}_{\mathbf{x}}[(Af_t)_i(\mathbf{x})x_j]| \leq \tilde{O}((m_t k d)^{-1})$ *for* j *that are out-of-support.*

To ensure the population gap in Lemma 3.4 is witnessed by the empirical distribution, we need only $\tilde{O}(2^{O(k)}\mathrm{poly}(d,k))$ samples. This translates to a gap in the gradient, which in turn translates to a similar gap in the student weights after the first stage of training. Lemma D.3 then applies, which allows us to learn the parity in a small number of samples overall.

At this point, we note an important difference between our proof and the one for progressive distillation in Panigrahi et al. (2024a). In their work, the weights after the first stage of training are able to adjust to the current top layer, this dependence allows them to more easily demonstrate a gap. In fact, we also observe the effect of being able to tune to the current top layer in Figure 3 (a), where we see that progressive distillation is able to make some progress even during the stage where we only tune the bottom layer weights. In our setting, in the first phase of training, we train the bottom layer of the student independently of the top layer, requiring us to rely on a different argument.

3.3 Experiments

We investigate curriculum extraction for the problem of learning sparse parities for a Multi-Layer Perceptron (MLP) and Transformer architectures, which we describe below:

Student and Teacher Architectures For the *Multi-Layer Perceptron,* both the student and teacher are two-layer MLPs with the teacher network having a hidden dimension of 5×10^4 and the student having hidden dimension 100. For the *transformer architecture*, the transformer configuration has matching embedding dimensions (256 dimensions for both teacher and student); however, the teacher has 32 attention heads and the student has only 4. Both student as well as teacher architectures use two decoder blocks followed by a linear projection layer.

Training and Evaluation Our distillation loss is the Mean Squared Error (MSE) and the final checkpoint training is done using the Cross-Entropy loss. We measure performance of our model by looking at the accuracy.

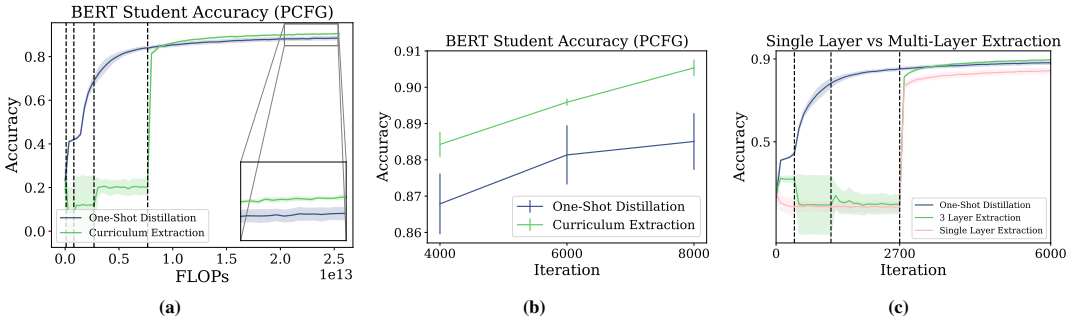

Figure 4: **PCFG Experiments on BERT.** The dashed vertical lines indicate iterations where the layer being distilled from is changed. (a) Soon after the final checkpoint, curriculum extraction achieves a larger accuracy in the same number of FLOPs, when compared to one-shot distillation. (b) We compare curriculum extraction to one-shot distillation across three models trained with two, three, and four-stage curricula at 4000, 6000, and 8000 iterations, respectively. Curriculum extraction consistently outperforms one-shot distillation at all scales. (c) We compare curriculum extraction performance by varying the number of layers. With a fixed budget of 6000 iterations (2700 for extraction, 3300 for full network training), extracting from three layers outperforms one-shot distillation, and using a single layer.

3.3.1 Discussion

Multi-Layer Perceptron: In Figure 3 (a), we compare the performance of MLPs trained using one-shot distillation, layer-wise curriculum extraction, and the progressive distillation approach from Panigrahi et al. (2024a). In both curriculum extraction and progressive distillation, we switch to the final checkpoint at iteration 5×10^5.

We observe that after 10^6 iterations, both methods perform comparably, with progressive distillation achieving slightly higher accuracy. Notably, progressive distillation shows early improvement before the 5×10^5 iteration mark, likely due to the MLP's bottom layer quickly tuning to the top layer. In the second phase of training, the top layer benefits from a well-optimized starting point. With curriculum extraction on the other hand, the bottom layer starts randomly initialized and uncorrelated with the top layer, limiting early gains. However, once we begin training the top layer, performance improves rapidly, matching progressive distillation. In contrast, one-shot distillation shows no significant improvement, even after 2×10^6 iterations. In fact, we observe in Figure 6 that our curriculum extraction method leads to the more information about the support being transferred to the underlying network than progressive distillation.

Transformer: In Figure 3 (b), we compare the performance of the transformer model trained using one-shot distillation and our curriculum extraction scheme, with a checkpoint at 10^5 iterations. We see that this leads to significantly improved student performance over the one-shot distillation case.

4 BERT Language Modeling

We use BERT models trained for masked prediction, a task that involves predicting the tokens hidden (masked) in an input sequence. Similar to Panigrahi et al. (2024a) we study this task for learning probabilistic context-free grammars (PCFGs) and natural language modeling on the Wikipedia dataset. We define the masked token prediction task below.

Problem 4.1 (Masked Token Prediction)**.** Let v be a vocabulary (including the token $[mask]$), and let x be a sequence of length h. We randomly choose a fraction (30%) of the positions $M \subseteq [h]$ to mask, with each position included independently with probability p. We create a masked input $x \setminus M$ by replacing tokens in M with $[mask]$, a random token, or leaving them unchanged with respective probabilities 80%, 10%, 10%. The model is then trained via a cross-entropy objective to predict the original tokens at those masked positions.

For sequence-to-sequence modelling our teacher and student networks map from sequences to real vectors $f_t, f_s \colon v^h \to \mathbb{R}^{h \times C}$. The teacher's output distribution at position i is given by $p_T^{(i)}(\mathbf{x}; \tau) = \mathrm{softmax}\big([f_t(\mathbf{x})]_i / \tau\big)$, and $p_S^{(i)}$ is defined analogously for the student. We set the temperature τ to 10^{-4} for experiments on sparse parity and PCFG, and to 10^{-20} for Wikipedia.

Training and Evaluation For the masked prediction task, we use $\ell(\mathbf{x}; f_S) = \mathbb{E}_M \Big[\frac{1}{|M|} \sum_{i \in M} \mathrm{KL}\big(e_{x_i} \,\|\, p_S^{(i)}(\mathbf{x} \setminus M; \tau)\big) \Big]$, where e_y is a one-hot vector with a 1 at index y for the final layer training; and the MSE for the indetermediate layers as our distillation loss. Our final performance is measured using the top-1 accuracy on the masked tokens.

Student and Teacher Architectures For the *PCFG tasks*, the teacher model uses a BERT-style architecture with a 256-dimensional embedding, 32 attention heads, and 4 transformer blocks, trained with a batch size of 512. The student model retains the teacher's 4-block architecture and 512 batch size but reduces the embedding dimension to 64 and the number of attention heads to 8.
For *Wikipedia* language modeling, the teacher model follows a standard BERT-large configuration, with 768-dimensional embeddings, 12 attention heads, and 12 transformer blocks, trained with a batch size of 256. The student model preserves the teacher's 12-block depth and 256 batch size but reduces the embedding dimension to 256 and the number of attention heads to 4.

Probabilistic Context-Free Grammar A Probabilistic Context-Free Grammar (PCFG) generates sentences using a hierarchical tree structure, defined by non-terminal symbols, rules, a probability distribution over the rules, and a vocabulary of terminal symbols. PCFGs have been used as mechanistic proxies for language data (Panigrahi et al., 2024a; Allen-Zhu & Li, 2023; Zhao et al., 2023). We focus on masked token prediction for synthetic data from the cfg3b PCFG (Allen-Zhu & Li, 2023) (defined in Appendix F). Our layer-wise curriculum schedule outperforms one-shot distillation. To confirm this improvement is due to the curriculum and not increased training bandwidth, we compare models trained with different curricula but the same time steps.

4.1 Discussion

PCFG Experiments: In Figure 4 (a), curriculum extraction shows FLOPS savings compared to one-shot distillation, outperforming it after $\approx 0.8 \times 10^{13}$ FLOPS. Additionally, its accuracy improves at every checkpoint.

To establish that the reason for our improved performance is our scheme, In Figure 4 (b), we implement two distinct curriculum strategies for BERT distillation to train the student network while maintaining equivalent *bandwidth* across our experiments. We compare single-layer and multi-layer extraction, both with 6000 training steps and 2700 dedicated to curriculum extraction. The single-layer model uses one checkpoint at the end, while the three-layer model has checkpoints at 400, 1200, and 2700 steps. Extracting from three layers improves performance, while single-layer extraction appears to perform worse than one-shot distillation, possibly due to overfitting between the student's bottom layer and the teacher's upper layers.

In Figure 4 (c) we see similar performance gains to those in Figure 4 (b) for higher bandwidth experiments – for two, three and four-stage curricula. The two-stage curriculum skips the linear projection and first transformer layer for iterations 1 through 500, skips just the top linear layer for iterations 501 to 1500, and finally trains the entire network for iterations 1501 through 4000. The three-stage curriculum skips two encoder blocks for iterations 1 through 400, skips one encoder block for iterations 401 through 1200, and skips the final projection layer for iterations 1201 through 2700, finally training the entire network for iterations 2701 to 6000. The four-stage curriculum skips four encoder blocks for iterations 1 through 200, then three blocks for iterations 201 through 700, then two blocks for iterations 701 through 1500, one block for iteration 1501 through 3000 and finally just the final linear layer until iteration 8000.

Wikipedia In Figure 3 (c) we see that our extraction scheme also works extremely well on real-world data. We train our BERT model for 500 iterations while skipping 4 encoder blocks, a further 1000 iterations while skipping 3 encoder blocks, a further 2000 iterations skipping one encoder block, 4000 iterations skipping the final projection layer and finally the full network for 16500 iterations.

REFERENCES

Marah Abdin, Jyoti Aneja, Harkirat Behl, Sébastien Bubeck, Ronen Eldan, Suriya Gunasekar, Michael Harrison, Russell J Hewett, Mojan Javaheripi, Piero Kauffmann, et al. Phi-4 technical report. *arXiv preprint arXiv:2412.08905*, 2024.

Gustavo Aguilar, Yuan Ling, Yu Zhang, Benjamin Yao, Xing Fan, and Chenlei Guo. Knowledge distillation from internal representations. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 7350–7357, 2020.

Zeyuan Allen-Zhu and Yuanzhi Li. Physics of language models: Part 1, context-free grammar. *arXiv preprint arXiv:2305.13673*, 2023.

Rohan Anil, Gabriel Pereyra, Alexandre Passos, Robert Ormandi, George E Dahl, and Geoffrey E Hinton. Large scale distributed neural network training through online distillation. *arXiv preprint arXiv:1804.03235*, 2018.

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. URL https://openreview.net/forum?id=8XWP2ewX-im.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *ICML*, pp. 41–48, 2009.

Jang Hyun Cho and Bharath Hariharan. On the efficacy of knowledge distillation. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4793–4801, 2019. doi: 10.1109/ICCV.2019.00489.

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jiawei Wang, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Shengfeng Ye, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wanjia Zhao, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanhong Xu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Hrayr Harutyunyan, Ankit Rawat, Aditya Menon, Kim Seungyeon, and Sanjiv Kumar. Supervision complexity and its role in knowledge distillation, 01 2023.

John Hewitt and Christopher D Manning. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4129–4138, 2019.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *ArXiv*, abs/1503.02531, 2015. URL https://api.semanticscholar.org/CorpusID:7200347.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. Tinybert: Distilling BERT for natural language understanding. *EMNLP Findings*, pp. 4163–4174, 2020.

Tom Kocmi and Ondrej Bojar. Curriculum learning and minibatch bucketing in neural machine translation. 2017.

M Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. *Advances in neural information processing systems*, 23, 2010.

Chen Liang, Simiao Zuo, Qingru Zhang, Pengcheng He, Weizhu Chen, and Tuo Zhao. Less is more: Task-aware layer-wise distillation for language model compression. In *International Conference on Machine Learning*, pp. 20852–20867. PMLR, 2023.

Meta AI. Llama 3. 2024. https://ai.meta.com/blog/meta-llama-3/.

Seyed Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, Nir Levine, Akihiro Matsukawa, and Hassan Ghasemzadeh. Improved knowledge distillation via teacher assistant. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 5191–5198, 2020.

OpenAI. Openai o1-mini, 2024. URL https://openai.com/index/openai-o1-mini-advancing-cost-efficient-reasoning/.

Abhishek Panigrahi, Bingbin Liu, Sadhika Malladi, Andrej Risteski, and Surbhi Goel. Progressive distillation induces an implicit curriculum. *arXiv preprint arXiv:2410.05464*, 2024a.

Abhishek Panigrahi, Nikunj Saunshi, Kaifeng Lyu, Sobhan Miryoosefi, Sashank Reddi, Satyen Kale, and Sanjiv Kumar. Efficient stagewise pretraining via progressive subnetworks. *arXiv preprint arXiv:2402.05913*, 2024b.

Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, 2017.

S. Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. Patient knowledge distillation for bert model compression. In *Conference on Empirical Methods in Natural Language Processing*, 2019. URL https://api.semanticscholar.org/CorpusID:201670719.

Terence Tao and Van H. Vu. *Additive Combinatorics*. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2006.

Gemini Team. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context, 2024. URL https://arxiv.org/abs/2403.05530.

I Tenney. Bert rediscovers the classical nlp pipeline. *arXiv preprint arXiv:1905.05950*, 2019.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. *arXiv preprint arXiv:1905.09418*, 2019.

Haoyu Zhao, Abhishek Panigrahi, Rong Ge, and Sanjeev Arora. Do transformers parse while predicting the masked word? *arXiv preprint arXiv:2303.08117*, 2023.

A Organization

Appendix B gives an overview of the related work. Appendices C to E aim to provide a proof of Theorem 3.3. In Appendix C we recall some basic definitions and facts. In Appendix D we recall the analysis of the teacher in Panigrahi et al. (2024a); Barak et al. (2022). In Appendix E we analyze the number of samples required for our student to perform well. In Appendix F we formally define a PCFG and recall the definition of the **cfg3b** grammar from Allen-Zhu & Li (2023).

B Related Work

Knowledge Distillation Knowledge distillation (KD), pioneered by Hinton et al. (2015), transfers knowledge from computationally expensive teacher models to lightweight students by aligning output distributions. This paradigm has been widely adopted in modern language models for inference cost reduction. Examples of models trained via distillation include ChatGPT O1-mini (OpenAI, 2024), Gemini Flash (Team, 2024), and the Phi series of models (Abdin et al., 2024). Despite the success of distillation, there have been a number of works (Mirzadeh et al., 2020; Cho & Hariharan, 2019; Harutyunyan et al., 2023; Panigrahi et al., 2024a) that have observed that simply distilling using the output of the fully trained teacher network can be suboptimal, resulting in a "teacher-student gap" in capabilities. To circumvent this gap, these works have proposed *progressive distillation*, a technique that trains the student using intermediate checkpoints during the teacher training progressively, before training on the output of the fully-trained teacher. Panigrahi et al. (2024a) proposes that the intermediate checkpoints act as an implicit *curriculum*, allowing the student to learn simpler functions before moving to the final complex one, and shows theoretical and empirical evidence to support these claims.

Despite its promise, progressive distillation faces some challenges – (1) storing intermediate checkpoints for large models incurs prohibitive costs, (2) finding an effective checkpoint schedule to use is done heuristically (Panigrahi et al., 2024a), requiring costly trial-and-error, (3) Most models, including open-source ones (e.g., Llama 3 (Meta AI, 2024), Mistral (Jiang et al., 2023)) only release final weights, making progressive distillation impossible.

Our method builds a curriculum using just the fully-trained teacher network, and thus, avoids the shortcomings or progressive distillation. While our curriculum extraction method is related to layer-wise distillation methods proposed in the literature (Aguilar et al., 2020; Liang et al., 2023; Jiao et al., 2020; Sun et al., 2019), these methods typically require the teacher and student to have the same embedding dimension. In contrast, our method uses a *random projection* to embed the teacher's hidden representation into the student's embedding dimension, allowing us to accommodate differing embedding dimensions. Our method also crucially relies on the stage-wise curriculum extraction of each layer individually, while the aforementioned works typically distill teacher layers simultaneously with the final output. Moreover, all of these works are heuristic in nature; we provide rigorous guarantees showing the correctness of our method, albeit in a stylized setting.

Curriculum Learning Curriculum learning, formalized by Bengio et al. (2009), structures training data by difficulty to improve learning efficiency. Early NLP work relied on handcrafted curricula (Kocmi & Bojar, 2017), while modern approaches automate this process through self-paced learning (Kumar et al., 2010). Recent advances, such as those by (Panigrahi et al., 2024b), demonstrate that stagewise pretraining via incremental subnetworks achieves computational efficiency without compromising model quality, further supporting the benefits of progressive learning paradigms. Additionally, studies on BERT reveal that its layers encode linguistic hierarchies (Tenney, 2019; Hewitt & Manning, 2019), suggesting that intermediate representations can scaffold learning—though this typically requires access to intermediate checkpoints (Shwartz-Ziv & Tishby, 2017; Voita et al., 2019).

C Preliminaries

We define the Fourier expansion of a boolean function below.

Definition C.1 (Fourier Expansion of a Boolean Function)**.** *Let $f : \{-1,1\}^n \to \mathbb{R}$ be a Boolean function. The **Fourier coefficients** of f are defined as:*

$$\hat{f}(S) = \mathbb{E}_{\mathbf{x}\sim\{-1,1\}^n}\left[f(\mathbf{x}) \cdot \chi_S(\mathbf{x})\right],$$

where:

- *$S \subseteq [n]$ is a subset of the input coordinates,*
- *$\chi_S(\mathbf{x}) = \prod_{i\in S} x_i$ is the **parity function** corresponding to S,*

The function f can then be expressed in terms of its Fourier expansion:

$$f(\mathbf{x}) = \sum_{S\subseteq[n]} \hat{f}(S) \cdot \chi_S(\mathbf{x}).$$

C.1 Properties of the ReLU Function

We will need the following properties of the ReLU function.

Lemma C.2 (Properties of $\phi_b(t)$)**.** *Let $\sigma(t) := \max(0,t)$, and let $a, b \in \mathbb{R}$. Then $\phi_b(a) := \sigma(a+b) - \sigma(-a+b)$ satisfies the following:*

1. *$\phi_b(0) = 0$*
2. *$\phi_b(-t) = -\phi_b(t)$*
3. *$\phi_b(t)$ is monotonically non-decreasing in t.*

Proof. The proofs follow by the definition of $\phi_b(a)$.

1. $\phi_b(0) = \sigma(b) - \sigma(b)$.
2. $\phi_b(-t) = \sigma(-t+b) - \sigma(t+b) = -\phi_b(t)$.
3. This follows from the fact that $\sigma(t)$ is monotonically non-decreasing. If $t_1 \le t_2$, observe that

$$\begin{aligned}\phi_b(t_1) &= \sigma(t_1+b) - \sigma(-t_1+b)\\ &< \sigma(t_2+b) - \sigma(-t_2+b)\\ &= \phi_b(t_2).\end{aligned}$$

□

C.2 Probability Facts

We will need the following anticoncentration and concentration inequalities for sums of scaled Rademacher random variables.

Lemma C.3 (Littlewood-Offord Anticoncentration Lemma (Tao & Vu, 2006))**.** *Let $X_1, X_2, \ldots, X_n$ be independent random variables, each taking values in $\{-1,+1\}$ with equal probability $\frac{1}{2}$. Let $a_1, a_2, \ldots, a_n$ be real coefficients. Define the random sum*

$$S = \sum_{i=1}^{n} a_i X_i.$$

The probability that S lies in an interval of length t satisfies

$$\mathbb{P}[|S| \le t] \le \frac{C}{\sqrt{|\{a_i \mid |a_i| > t\}|}},$$

where $C > 0$ is an absolute constant.

We will also need the standard Hoeffding's inequality.

Lemma C.4 (Hoeffding's Inequality)**.** *Let $X_1, X_2, \ldots, X_n$ be independent random variables such that $a_i \leq X_i \leq b_i$ almost surely for each $i \in \{1, 2, \ldots, n\}$. Define the sample mean*

$$\overline{X} = \frac{1}{n} \sum_{i=1}^{n} X_i,$$

and let $\mu = \mathbb{E}[\overline{X}]$ be the expected value of the sample mean. Then for any $t > 0$,

$$\mathbb{P}\left(|\overline{X} - \mu| \geq t\right) \leq 2 \exp\left(-\frac{2n^2 t^2}{\sum_{i=1}^{n}(b_i - a_i)^2}\right).$$

D TEACHER TRAINING

In this section, we recall the teacher training analysis from Panigrahi et al. (2024a). Some lemmas from this review will be used in the analysis of our student model.

In Appendix D.1, we restate Lemma D.2, which shows that after one gradient descent step, the in-support weights become larger than the out-of-support weights. In Appendix D.2, we recall Lemma D.3, which establishes that if the conditions of Lemma D.2 are met, the teacher can learn the top-level weights within $O(d^{O(k)}\epsilon^{-2}\log(dk/\epsilon\delta))$ samples, achieving a loss of at most ϵ.

Recall that we use the teacher loss is given by the hinge loss $\ell_{f_t}(\mathbf{x}, y) = \max(0, 1 - f_t(\mathbf{x})y)$.

Algorithm 1 2-stage training for teacher

Require: Timestep T_2, Learning rates η_1, η_2, batch sizes B_1, B_2, weight decay λ_1.

1: **Inner Layer Training:**
2: **for** $t = 1$ **do**
3: Sample B_1-samples $\{(\mathbf{x}^{(j)}, y^{(j)})\}_{j=1}^{B_1}$.
4: Update the inner layer weights $\{\mathbf{w}_1^{(t)}, \ldots, \mathbf{w}_{m_t}^{(t)}\}$ as:

$$\mathbf{w}_i^{(t)} \leftarrow \mathbf{w}_i^{(t-1)} - \eta_1 \mathbb{E}_{(\mathbf{x},y)\in\{(\mathbf{x}^{(j)},y^{(j)})\}_{j=1}^{B_1}} \left[\nabla_{\mathbf{w}_i}\left(\ell_{f_t}(\mathbf{x}, y) + \lambda_1 \|\mathbf{w}_i^{(t-1)}\|^2\right)\right]$$

5: **end for**
6: **Outer Layer Training:**
7: **for** $t \in [0, T_2]$ **do**
8: Sample B_2-samples $\{(\mathbf{x}^{(j)}, y^{(j)})\}_{j=1}^{B_2}$.
9: Update the outer layer weights:

$$\mathbf{a}^{(t)} \leftarrow \mathbf{a}^{(t-1)} - \eta_2 \mathbb{E}_{(\mathbf{x},y)\in\{(\mathbf{x}^{(i)},y^{(i)})\}_{i=1}^{B_2}} \left[\nabla_{\mathbf{a}} \ell_{f_t}(\mathbf{x}, y)\right]$$

10: **end for**

Our teacher is trained in exactly the same way as in Panigrahi et al. (2024a), using Algorithm 1. For completeness, we recall conditions required for teacher training to succeed. Before we continue, we set up some notation.

Notation

- In what follows, B_1, B_2 are batch sizes, i.e. number of samples drawn to estimate the gradients in the first and second stages of training respectively. δ will denote the probability of failure, and λ_1 will denote a regularization parameter (as seen in Algorithm 1).
- τ_g is the error estimate of the gradient, i.e. for a network f, $\left|\mathbb{E}_{\mathbf{x},y\sim U(\{\pm 1\}^d)}\left[\nabla_{w_{ij}} f(\mathbf{x})\right] - \mathbb{E}_{\{(\mathbf{x}_k,y_k)\}_{k=1}^{B_1}}\left[\nabla_{w_{ij}} f(\mathbf{x})\right]\right| \leq \tau_g,$.
- m_t and m_s denote the hidden layer sizes of the student and teacher respectively.

- For $a, b, c \in \mathbb{R}$ we will say $a = b \pm c$ if $a \in [b - c, b + c]$. for $u, v \in \mathbb{R}^d$, we define $u \odot v = (u_1 v_1, \ldots, u_d v_d)$.
- $\theta(t) := \{\mathbf{W}^{(t)}, \mathbf{a}^{(t)}, \mathbf{b}^{(t)}\}$ denotes the parameters of the model that the algorithm recovers at timestep t.
- $\mathrm{Maj}(\mathbf{x}) : \{\pm 1\}^d \to \{-1, 0, 1\}$ returns $\mathrm{sign}(\sum_{i=1}^d x_i)$, where $\mathrm{sign}(0) := 0$ and $\mathrm{sign}(t) := t/|t|$ for $t \neq 0$.

We will also need the following lemma that controls the gradient error as a function of the batch size.

Claim D.1 (Gradient Concentration Panigrahi et al. (2024a)). *Let f be a two-layer network initialized using the symmetric initialization in Definition 3.2 with m being its hidden dimension. Fix $\delta, \tau_g > 0$. For all $i \in [m], j \in [d]$, for a randomly sampled batch of size B_1, $\{(\mathbf{x}_k, y_k)\}_{k=1}^{B_1}$, with probability at least $1 - \delta$,*

$$\left| \mathbb{E}_{\mathbf{x}, y \sim U(\{\pm 1\}^d)} \left[\nabla_{w_{ij}} f(\mathbf{x}) \right] - \mathbb{E}_{\{(\mathbf{x}_k, y_k)\}_{k=1}^{B_1}} \left[\nabla_{w_{ij}} f(\mathbf{x}) \right] \right| \leq \tau_g,$$

provided $B_1 \geq \Omega\left(\tau_g^{-2} \log(md/\delta)\right)$.

D.1 Teacher analysis after first stage of training

After the first stage of teacher training, the weights satisfy the property that they have larger magnitudes for in-support indices compared to out-of-support indices. This property is crucial for the second stage of training to achieve a good solution. This behavior is formally captured by the following lemma.

Lemma D.2 (Lemma B.2 (Single step gradient descent, from Panigrahi et al. (2024a))). *Let ζ_k denote the k^{th} Fourier coefficient of the majority function. Fix $\tau_g, \delta > 0$. Set $T_1 = 1$. Suppose the batch size $B_1 \geq \Omega(\tau_g^{-2} \log(m_t d/\delta))$. For learning rate $\eta_1 = \frac{m_t}{k|\zeta_{k-1}|}$ and $\lambda_1 = 1/2$, the following conditions hold true for all neurons $i \in [m]$ at the end of the first stage of training with probability at least $1 - \delta$.*

1. $\left| w_{\ell j}^{(1)} - \frac{\mathrm{sign}(a_\ell^{(0)} \zeta_{k-1}) \, \mathrm{sign}(\chi_{[k] \setminus \{j\}}(w_\ell^{(0)}))}{2k} \right| \leq \frac{\tau_g}{|\zeta_{k-1}|}$, *for all* $j \in [k]$.
2. $\left| w_{\ell j}^{(1)} - \frac{\zeta_{k+1}}{|\zeta_{k-1}|} \frac{\mathrm{sign}(a_\ell^{(0)}) \, \mathrm{sign}(\chi_{[k] \cup \{j\}}(w_\ell^{(0)}))}{2k} \right| \leq \frac{\tau_g}{|k \zeta_{k-1}|}$, *for all* $j > k$.

While we do not reproduce the proof of Lemma D.2, we point out that the proof essentially follows by demonstrating that the gradients $\nabla_{w_{\ell j}} [\ell_{f_t}(\mathbf{x}, y)]$ initialization satisfy properties similar to the ones stated above for $w_{\ell j}^{(1)}$, and setting $\lambda_1 = 1/2\eta_1$ ensures that the weights after one step are proportional to these gradients $w_{\ell j}^{(1)} = -\eta_1 \nabla_{w_{\ell j}} [\ell_{f_t}(\mathbf{x}, y)]$.

Teacher batch size: A consequence of Lemma D.2 is that, since we need the gap to be witnessed by the empirical gradients, the teacher batch size will be lower bounded by $B_1 \geq (d^2 k \, \zeta_{k-1})^2 \log(m_t d/\delta) \geq \Omega(d^{k-1})$. From this, we see that *even a moderate-sized* teacher requires $\Omega(d^{k-1})$ samples, according to training algorithm. In fact this holds more generally (as shown in Panigrahi et al. (2024a)).

D.2 Teacher analysis after second stage of training

Under the conclusions of Lemma D.2, the second stage of training produces a function with small loss relative to the unknown parity function, so long as the hidden layer is sufficiently large (i.e. $m_t \geq 2^k k \log(k/d)$). This result is formalized in the following theorem, which will be used to analyze the second stage of training for both the student and the teacher.

Lemma D.3 (Theorem 4, Barak et al. (2022), version from Panigrahi et al. (2024a)). *Fix $\epsilon, \delta > 0$. Suppose $m \geq \Omega(2^k k \log(k/\delta))$, $d \geq \Omega(k^4 \log(kd/\epsilon))$. Furthermore, suppose $B_1 \geq \Omega(d^k k^2 \log(kd/\epsilon))$ such that the weights satisfy the conditions in Lemma D.2 with $\tau_g = O(d^{-k} k^{-1} d^{-2})$ after the first phase, and let $\theta(t)$ denote the model at timestep t. Then after $T_2 = \Omega(md^2 k^3/\epsilon^2)$ steps of training*

with batch size $B_2 = 1$ and learning rate $\eta_2 = 4k^{1.5}/(dm(T_2 - 1))$, we have, with expectation over the randomness of the initialization and the sampling of the batches:

$$\min_{t \in [T_2]} \mathbb{E}[L_{\theta(t)}(x, y)] \leq \epsilon.$$

Thus, the minimal sample complexity to reach a loss of ϵ is given by:

$$T_1 \times B_1 + T_2 \times B_2 = \Theta(d^{O(k)} \epsilon^{-2} \log(dk/\epsilon\delta)).$$

Remark: The higher sample complexity of this algorithm is not primarily due to the teacher's hidden dimension (since it can be set to $O(2^k k \log(k/\delta))$), but rather due to the need for accurately estimating the gradients to an error of $O(d^{-k} k^{-1} d^{-2})$.

E Student Training

In this section, we analyze our student training algorithm (Algorithm 2). The student algorithm differs from the teacher training algorithm only in the first phase, where we use the distillation loss $\ell_{DL}(\mathbf{x}, f, g) := -f(\mathbf{x}) \cdot g(\mathbf{x})$. Here, $f = f_s^{(1)} \in \mathbb{R}^{m_s}$ is the first layer of the student network, and $g = A f_t^{(1)}(\mathbf{x}) \in \mathbb{R}^{m_s}$ is the first layer of the teacher network projected to the student's hidden layer dimension.

Algorithm 2 2-stage training for teacher

Require: Timestep T_2, Learning rates η_1, η_2, batch sizes B_1, B_2, weight decay λ_1.

Inner Layer Training:
for $t = 1$ **do**
  Sample B_1-samples $\{(\mathbf{x}^{(j)}, y^{(j)})\}_{j=1}^{B_1}$.
  Update the inner layer weights $\{\mathbf{w}_1^{(t)}, \ldots, \mathbf{w}_{m_s}^{(t)}\}$ as:

$$\mathbf{w}_i^{(t)} \leftarrow \mathbf{w}_i^{(t-1)} - \eta_1 \mathbb{E}_{(\mathbf{x},y) \in \{(\mathbf{x}^{(j)}, y^{(j)})\}_{j=1}^{B_1}} \left[\nabla_{\mathbf{w}_i} \left(\ell_{DL}(\mathbf{x}, f_s^{(1)}, A f_t^{(1)}) + \lambda_1 \|\mathbf{w}_i^{(t-1)}\|^2 \right) \right]$$

end for
Outer Layer Training:
for $t \in [0, T_2]$ **do**
  Sample B_2-samples $\{(\mathbf{x}^{(j)}, y^{(j)})\}_{j=1}^{B_2}$.
  Update the outer layer weights:

$$\mathbf{a}^{(t)} \leftarrow \mathbf{a}^{(t-1)} - \eta_2 \mathbb{E}_{(\mathbf{x},y) \in \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{B_2}} \left[\nabla_{\mathbf{a}} \ell(\mathbf{x}, y) \right]$$

end for

E.1 First stage analysis of the student

Most of our effort will focus on showing that $\mathbf{W}_s^{(1)}$, the first layer of the student network after the first stage of training, satisfies a property similar to the conclusion of Lemma D.2.

By choosing $\lambda_1 = 1/(2\eta_1)$ in Algorithm 2, we obtain

$$\mathbf{w}_i^{(1)} = -\eta_1 \mathbb{E}_{(\mathbf{x},y) \in \{(\mathbf{x}^{(j)}, y^{(j)})\}_{j=1}^{B_1}} \left[\nabla_{\mathbf{w}_\ell} \ell_{DL}(\mathbf{x}, f_s^{(1)}, A f_t^{(1)}) \right].$$

Thus, it suffices to show that the gradient update for the student has larger magnitudes for in-support coordinates than for out-of-support coordinates. This is captured in Lemma E.3 and corollary E.6, which will be the focus of this section. We first recall some variants of lemmas from Panigrahi et al. (2024a) which we will need.

E.1.1 Preliminary Setup

The following lemma shows that this may be expressed as a function of the fourier coefficients of $(A f_t^{(1)}(\mathbf{x}))_i$ and $(A f_t^{(1)}(\mathbf{x}))_i$ $\mathrm{Maj}(\mathbf{w}_i \odot \mathbf{x})$.

Lemma E.1 (Teacher Correlation Gap implies Student Gradient Gap). *Let* $h_i(\mathbf{x}) := (Af_t^{(1)}(\mathbf{x}))_i$. *For all* $i \in [m_s]$, *suppose*

$$\left|\mathop{\mathbf{E}}_{\mathbf{x}}\left[h_i(\mathbf{x})\, x_j\right] + \mathop{\mathbf{E}}_{\mathbf{x}}\left[h_i(\mathbf{x})\ \mathrm{Maj}(\mathbf{w}_i \odot \mathbf{x}) x_j\right]\right| = \begin{cases} \geq \frac{1}{m_t k}, & \text{for } j \in [k], \\ < \frac{1}{m_t k d}, & \text{for } j > k. \end{cases}$$

Then,

$$\mathop{\mathbf{E}}_{\mathbf{x}}\left[\nabla_{w_{ij}} \ell_{DL}(\mathbf{x}, f_s^{(1)}, Af_t^{(1)})\right] = \begin{cases} \Omega\left(\frac{1}{m_t k}\right), & \text{for } j \in [k], \\ o\left(\frac{1}{m_t k d}\right), & \text{otherwise.} \end{cases}$$

Proof. At initialization, the gradient of the weight vector of neuron i at coordinate j is given by,

$$\begin{aligned} \mathop{\mathbf{E}}_{\mathbf{x}}[\nabla_{w_{ij}} \ell_{DL}(\mathbf{x}, f_s^{(1)}, Af_t^{(1)})] &= -\mathop{\mathbf{E}}_{\mathbf{x}}[\nabla_{w_{ij}} (f_s^{(1)} \cdot Af_t^{(1)})] \\ &= -\mathop{\mathbf{E}}_{x}[\mathbf{1}(\mathbf{w}_i \cdot \mathbf{x} + b_i \geq 0)(Af_t^{(1)})_i x_j] \end{aligned}$$

Since $|b_i| < 1$ and $\mathbf{w}_i, \mathbf{x} \in \{\pm 1\}^d$ at initialization, $\mathbf{1}(\mathbf{w}_i \cdot \mathbf{x} + b_i \geq 0) = \frac{1}{2} + \frac{\mathrm{Maj}(\mathbf{w}_i \odot \mathbf{x})}{2}$. Substituting this above,

$$\mathop{\mathbf{E}}_{\mathbf{x}}[\nabla_{w_{ij}} \ell_{DL}(\mathbf{x}, f_s^{(1)}(\mathbf{x}), Af_t^{(1)}(\mathbf{x}))] = -\frac{1}{2}\left(\mathop{\mathbf{E}}_{\mathbf{x}}[(Af_t^{(1)})_i(\mathbf{x})\, x_j] + \mathop{\mathbf{E}}_{\mathbf{x}}[(Af_t^{(1)})_i(\mathbf{x})\ \mathrm{Maj}(\mathbf{w}_i \odot \mathbf{x})\, x_j]\right)$$

□

Define $\phi_b(a) := \sigma(a + b) - \sigma(-a + b)$. Then, we see that for $\mathbf{W}^{(0)}$ initialized according to the scheme in Section 3.1, the following holds:

Lemma E.2 (Bounds on coefficients). *For a teacher network in the setting of Lemma D.3; with probability* $1 - \delta$ *over the randomness of initialization of* b_ℓ, *the following hold as long as* $m_t \geq 10 \log(1/\delta)$*:*

1. *For* $j \in [k]$, *there are at least* $m_t/8$ *values of* $\ell \in [m_t/2]$ *satisfying* $|\mathbf{E}_{\mathbf{x}}[\phi_{b_\ell}(\mathbf{w}_\ell \cdot \mathbf{x})\, x_j]| \geq \Omega(1/k)$.
2. *For all* $j > k$ *and* $\ell \in [m_t/2]$, $|\mathbf{E}_{\mathbf{x}}[\phi_{b_\ell}(\mathbf{w}_\ell \cdot \mathbf{x})]| \leq O(1/kd)$.

Proof. This result follows from the calculations in the "estimates of in-support correlations" and "estimations of out-of-support correlations" sections of Lemma B.5 in Panigrahi et al. (2024a) (pages 25–26).

Item 1 follows from the analysis in the "estimates of in-support correlations" section, which shows that with probability at least $1/2$ over the randomness of b_ℓ,

$$|\mathbb{E}_{\mathbf{x}}[\phi_{b_\ell}(\mathbf{w}_\ell \cdot \mathbf{x}) x_j]| \geq \frac{1}{4k} - O(\tau_g d |\zeta_{k-1}|^{-1}).$$

Applying Hoeffding's inequality to this event, we conclude that if $m_t \geq \Omega(\log(1/\delta))$, then with probability $1 - \delta$, at least $m_t/8$ neurons satisfy

$$|\mathbb{E}_{\mathbf{x}}[\phi_{b_\ell}(\mathbf{w}_\ell \cdot \mathbf{x}) x_j]| \geq \frac{1}{16k} - O(\tau_g d |\zeta_{k-1}|^{-1}).$$

Item 2 follows directly from the "estimations of out-of-support correlations" section.

The error term $2d\tau_g |\xi_{k-1}|^{-1}$ for the trained *teacher network* is controlled by setting τ_g appropriately. Note that this is not something that affects the student sample complexity. □

E.1.2 Teacher correlation Gap and Gradient Correlation Gap

Most of our effort will now focus on establishing the hypothesis of Lemma E.1. In this section, we demonstrate a gap between the in-support and out-of-support indices $j \in [d]$ for any fixed i in the expression, $\mathbf{E}_{\mathbf{x}}[(Af_t^{(1)})_i(\mathbf{x})\ x_j] + \mathbf{E}_{\mathbf{x}}[(Af_t^{(1)})_i(\mathbf{x})\ \mathrm{Maj}(\mathbf{w}_i \odot \mathbf{x})\ x_j]$.

In Lemma E.3 we show this gap for the first term, $\mathbf{E}_{\mathbf{x}}[(Af_t^{(1)})_i(\mathbf{x})\ x_j]$. In Corollary E.6 we show this gap for the second term $\mathbf{E}_{\mathbf{x}}[(Af_t^{(1)})_i(\mathbf{x})\ \mathrm{Maj}(\mathbf{w}_i \odot \mathbf{x})\ x_j]$. In what follows, we will suppress the dependence on i, as the argument is identical for each output coordinate i. We reuse this variable instead to range over the teacher hidden dimension.

Lemma E.3 (Correlation Gap for One Projected Teacher Dimension)**.** *Let $f(\mathbf{x}) := \sum_{i=1}^{m_t} a_i^{(0)} \sigma(\mathbf{w}_i \cdot \mathbf{x} + b_i)$ and $f^r(\mathbf{x}) = \sum_{i=1}^{m_t} a_i^r \sigma(\mathbf{w}_i \cdot \mathbf{x} + b_i)$, where a_i^r are independently drawn u.a.r. from $U(\{\pm 1/m_t\})$ for $i \in [m_t/2]$ and $a_{i+m_t/2}^r = -a_i^r$, and $a_i^{(0)}$ and $\mathbf{W}^{(0)}$ are initialized according to the initialization scheme in Section 3.2.1. Let $m_t \geq \Omega(k^4 \log(d)^2/\delta^2)$, then*

$$\min_{j\in[k]} \left|\mathop{\mathbf{E}}_{\mathbf{x}}[f^r(\mathbf{x})x_j]\right| > \frac{1}{m_t k} \text{ and } \max_{j>k} \left|\mathop{\mathbf{E}}_{\mathbf{x}}[f^r(\mathbf{x})x_j]\right| < \frac{1}{m_t k d}$$

Proof. To make it easier for us to estimate $\mathbf{E}_{\mathbf{x}}[f^r(\mathbf{x})x_j]$, we can rewrite $f^r(\mathbf{x})$:

$$\begin{aligned} f^r(\mathbf{x}) &= \sum_{i=1}^{m_t} a_i^r \sigma(\mathbf{w}_i \cdot \mathbf{x} + b_i) \\ &= \sum_{i=1}^{m_t/2} a_i^r (\sigma(\mathbf{w}_i \cdot \mathbf{x} + b_i) - \sigma(-\mathbf{w}_i \cdot \mathbf{x} + b_i)) \end{aligned}$$

The final equality follows by combining the terms for $\ell = i$ and $\ell = i + m_t/2$.

Define $\phi_b(a) := \sigma(a + b) - \sigma(-a + b)$. Using the linearity of expectation after multiplying by x_j, we obtain:

$$\mathbb{E}_{\mathbf{x}}[f^r(\mathbf{x})x_j] = \sum_{i=1}^{m_t/2} a_i^r \mathbb{E}_{\mathbf{x}}\left[\phi_{b_i}(\mathbf{w}_i \cdot \mathbf{x})x_j\right].$$

To derive bounds on $\mathbb{E}_{\mathbf{x}}[f^r(\mathbf{x})x_j]$, we interpret the right-hand side as a sum of Rademacher random variables scaled by coefficients. The remainder of the proof relies on anti-concentration and concentration inequalities for such sums. To apply these inequalities, we require bounds on the coefficients a_i^r, specifically $\mathbb{E}_{\mathbf{x}}[\phi_{b_i}(\mathbf{w}_i \cdot \mathbf{x})x_j]$. These bounds are provided by Lemma E.2 implicit in Panigrahi et al. (2024a)

Given Lemma E.2, we apply the Littlewood-Offord lemma (Lemma C.3) to show that the sum $\sum_{i=1}^{m_t/2} a_i^r\, \mathbf{E}_{\mathbf{x}}[\phi_{b_i}(\mathbf{w}_i \cdot \mathbf{x})x_j]$ cannot be too small with large probability.

$$\Pr_{a_i^r}\left[\exists j \in [k].\ \left|\sum_{i=1}^{m_t/2} a_i^r \mathop{\mathbf{E}}_{\mathbf{x}}[\phi_{b_i}(\mathbf{w}_i \cdot \mathbf{x})x_j]\right| \leq \Omega\left(\frac{1}{m_t k}\right)\right] \leq O\left(\frac{k}{\sqrt{m_t}}\right), \tag{1}$$

where the final k on the right hand side is a consequence of a union bound over $j \in [k]$. Similarly, an consequence of Hoeffding's inequality (Lemma C.4) and Item 2 in Lemma E.2 is,

$$\Pr_{a_i^r}\left[\exists j > k.\ \left|\sum_{i=1}^{m_t/2} a_i^r \mathop{\mathbf{E}}_{\mathbf{x}}[\phi_{b_i}(\mathbf{w}_i \cdot \mathbf{x})x_j]\right| > O\left(\frac{1}{m_t k d}\right)\right] \leq d\exp(-2m_t) \tag{2}$$

Hence, if $m_t \geq \Omega(k^4 \log(d)^2/\delta^2)$ we see that with probability at least $1 - \delta$,

$$\min_{j\in[k]} \left|\sum_{i=1}^{m_t/2} a_i^r \mathop{\mathbf{E}}_{\mathbf{x}}[\phi_{b_i}(\mathbf{w}_i \cdot \mathbf{x})x_j]\right| > \Omega\left(\frac{1}{m_t k}\right)$$

and

$$\max_{j>k} \left| \sum_{i=1}^{m_t/2} a_i^r \mathop{\mathbf{E}}_{\mathbf{x}}[\phi_{b_i}(\mathbf{w}_i \cdot \mathbf{x}) x_j] \right| < O\left(\frac{1}{m_t k d}\right)$$

Whenever $d > k$, this difference is $\Omega(1/m_t k^2)$. □

We now focus on getting similar bounds on $\mathbb{E}_{\mathbf{x}}[f^r(\mathbf{x}) \cdot \mathrm{Maj}(\mathbf{w} \odot \mathbf{x}) x_j]$. To bound the first term, we must first bound all the Fourier coefficients (Definition C.1) of $f^r(\mathbf{x})$. This is necessary because the degree-1 Fourier coefficients of a product of Boolean functions (in this case $f^r(\mathbf{x})$ and $\mathrm{Maj}(\mathbf{w} \odot \mathbf{x})$) depend on their *entire* Fourier expansions (see Lemma E.4 below).

Lemma E.4 (Fourier Coefficients of Inner Product)**.** *Let $f, g : \{-1, 1\}^n \to \mathbb{R}$ be two Boolean functions with Fourier expansions:*

$$f(x) = \sum_{S \subseteq [n]} \hat{f}(S) \chi_S(x) \quad \textit{and} \quad g(x) = \sum_{T \subseteq [n]} \hat{g}(T) \chi_T(x),$$

where $\chi_S(x) = \prod_{i \in S} x_i$ are the parity (Walsh) basis functions, and $\hat{f}(S)$, $\hat{g}(T)$ are the Fourier coefficients of f and g, respectively.

Then, the Fourier coefficients of the inner product $h(x) = f(x) \cdot g(x)$ are given by:

$$\hat{h}(S) = \sum_{T \subseteq [n]} \hat{f}(T) \hat{g}(S \triangle T),$$

where $S \triangle T$ denotes the symmetric difference of the sets S and T.

These bounds on the expansion of $(A f_t^{(1)})$ follow from the following modified versions of Lemma B.5 and Corollary B.6 from Panigrahi et al. (2024a), which we state below.

Lemma E.5 (Correlation within-support variables)**.** *Under the event that the conditions in Lemma D.2 are satisfied by each neuron, which occurs with probability at least $1 - \delta$ w.r.t. the randomness of initialization as long as $m_t \geq \Omega(m_s^2 \, k^4 \log(d)^2 / \delta^2)$, the output of the model after the first phase satisfies the following conditions:*

1. $\mathbb{E}_{\mathbf{x},y}\left[(A f_t^{(1)})_i(\mathbf{x}) x_j\right] \geq \Omega(\frac{1}{m_t k})$ *for all $j \in S$.*
2. $\mathbb{E}_{\mathbf{x},y}\left[(A f_t^{(1)})_i(\mathbf{x}) x_j\right] \leq O\left(\frac{1}{m_t k d}\right)$ *for all $j \notin S$.*
3. $\mathbb{E}_{\mathbf{x},y}\left[(A f_t^{(1)})_i(\mathbf{x}) \chi_S(\mathbf{x})\right] \leq O\left(\tau_g d |\zeta_{k-1}|^{-1}\right)$ *for all S with even $|S|$.*
4. $\left\|(A f_t^{(1)})_i(\mathbf{x})\right\|_2^2 = \mathbb{E}_{\mathbf{x},y}\left[(A f_t^{(1)})_i(\mathbf{x})\right]^2 \leq O\left(\frac{d}{k}\right)$.

Proof. The proofs of the first two items follow from Lemma E.3. The proofs of the second two items are exactly the same as the proofs of Items 3 and 4 of Lemma B.5 in Panigrahi et al. (2024a) □

Lemma E.5 now allows us to recover the following variant of Corollary B.6 from Panigrahi et al. (2024a), effectively obtaining a similar gap for $\mathbf{E}_{\mathbf{x},y}[f^r(\mathbf{x}) \cdot \mathrm{Maj}(\mathbf{w} \odot \mathbf{x}) x_j]$.

Corollary E.6 (Fourier expansion of $f^r(\mathbf{x})$)**.** *Let $f^r(\mathbf{x})$ be defined as in Lemma E.2 and suppose the conditions in Lemma E.2 are satisfied by each neuron, which occurs with probability at least $1 - \delta$ with respect to the randomness of initialization and sampling, the output of the model after the first phase can be given as:*

$$f^r(\mathbf{x}) = \sum_{j=1}^{k} c_j x_j + \sum_{j=k+1}^{d} c_j x_j + \sum_{\substack{S \subseteq [d] \\ |S|\%2=1, |S| \geq 3}} c_S \chi_S(x) + \sum_{\substack{S \subseteq [d] \\ |S|\%2=0}} c_S \chi_S(x),$$

where

$$\begin{aligned}
&|c_j| \geq \Omega((km_t)^{-1}), && \text{for all } 1 \leq j \leq k,\\
&|c_j| \leq O((kdm_t)^{-1}), && \text{for all } j > k,\\
&|c_S| \leq O(\tau_g d|\zeta_{k-1}|^{-1}), && \text{for all } S \subseteq [d] \text{ with } |S|\%2 = 0,\\
&|c_S| \leq O(d/k), && \text{for all } S \subseteq [d] \text{ with } |S|\%2 = 1.
\end{aligned}$$

As such, given a fixed $\mathbf{w}$*, the following correlations hold true for all* i*:*

$$\mathbb{E}_{x,y}\left[f^r(\mathbf{x}) \cdot \mathit{Maj}(\mathbf{w} \odot \mathbf{x})x_i\right] = O(\tau_g d^{5/3}|\zeta_{k-1}|^{-1}).$$

If the batch size B_1 *is set such that* $B_1 \geq \Omega(k^2 d^{10/3}\zeta_{k-1}^{-4})$ *and* $\tau_g \leq O(k^{-1}d^{-5/3}|\zeta_{k-1}|m_t^{-1})$*, then the following holds for all* i*:*

$$\begin{aligned}
&\mathbb{E}_{x,y}\left[f^r(\mathbf{x}) \cdot \mathit{Maj}(\mathbf{w} \odot \mathbf{x})x_j\right] \geq \Omega((m_t k)^{-1}), && \text{if } j \in [k],\\
&\mathbb{E}_{x,y}\left[f^r(\mathbf{x}) \cdot \mathit{Maj}(\mathbf{w} \odot \mathbf{x})x_j\right] \leq o((m_t kd)^{-1}), && \text{if } j \notin [k].
\end{aligned}$$

Proof. Observe that for a fixed $\mathbf{w}$,

$$\begin{aligned}
\mathbb{E}_{\mathbf{x},y}\left[f^r(\mathbf{x}) \cdot \mathrm{Maj}(\mathbf{w} \odot \mathbf{x})x_i\right] = \mathop{\mathbf{E}}_{\mathbf{x},y}\left[\sum_{j=1}^{d} c_j x_j \,\mathrm{Maj}(\mathbf{w} \odot \mathbf{x})x_i\right] + \sum_{S\subset[d],|S|\%2=1,|S|\geq 3} \mathop{\mathbf{E}}_{\mathbf{x},y}\left[c_S \,\mathrm{Maj}(\mathbf{w} \odot \mathbf{x})\chi_S(\mathbf{x})x_i\right]\\
+ \mathop{\mathbf{E}}_{\mathbf{x},y}\left[\sum_{S\subset[d],|S|\%2=0} c_s \,\mathrm{Maj}(\mathbf{w} \odot \mathbf{x})\chi_S(\mathbf{x})x_i\right].
\end{aligned}$$

Since $\mathrm{Maj}(\mathbf{w}\odot\mathbf{x})$ is an odd function (for a fixed $\mathbf{w}$), $\mathbf{E}_{\mathbf{x},y}[\mathrm{Maj}(\mathbf{w}\odot\mathbf{x})\chi_S(\mathbf{x})x_i] = 0$ for $|S|\%2 = 1$. This allows us to remove the term. A similar argument holds for the first term, giving us

$$\mathbb{E}_{\mathbf{x},y}\left[f^r(\mathbf{x}) \cdot \mathrm{Maj}(\mathbf{w} \odot \mathbf{x})x_i\right] = c_i \mathop{\mathbf{E}}_{\mathbf{x},y}\left[\mathrm{Maj}(\mathbf{w} \odot \mathbf{x})\right] + \mathop{\mathbf{E}}_{\mathbf{x},y}\left[\sum_{S\subset[d],|S|\%2=0} c_s \,\mathrm{Maj}(\mathbf{w} \odot \mathbf{x})\chi_S(\mathbf{x})x_i\right].$$

The first term is 0 because $\mathbf{E}_{\mathbf{x},y}[\mathrm{Maj}(\mathbf{w} \odot \mathbf{x})] = 0$, since, $\mathrm{Maj}(\mathbf{w} \odot \mathbf{x}) = -\mathrm{Maj}(\mathbf{w} \odot (-\mathbf{x}))$. The second term may be bounded as follows,

$$\begin{aligned}
&\left|\mathop{\mathbf{E}}_{\mathbf{x},y}\sum_{S\subset[d],S\%2=0} c_S \,\mathrm{Maj}(\mathbf{w} \odot \mathbf{x})\chi_S(\mathbf{x})x_i\right|\\
&\leq O(\tau_g d|\zeta_{k-1}|^{-1}) \cdot \left(\sum_{S\subset[d],S\%2=0} |\mathop{\mathbf{E}}_{\mathbf{x},y} \mathrm{Maj}(\mathbf{w} \odot \mathbf{x})\chi_S(\mathbf{x})x_i|\right)\\
&\leq O(\tau_g d|\zeta_{k-1}|^{-1}) \cdot \left(\sum_{S\subset[d],S\%2=0} |\mathop{\mathbf{E}}_{\mathbf{x},y} \mathrm{Maj}(\mathbf{w} \odot \mathbf{x})\chi_S(\mathbf{x})|\right)\\
&\leq O(\tau_g d|\zeta_{k-1}|^{-1}) \cdot \left(\sum_{S\subset[d],S\%2=0} \Theta\left(\frac{|S|^{-1/3}}{\binom{d}{|S|}}\right)\right)\\
&\leq O(\tau_g d^{5/3}|\zeta_{k-1}|^{-1})
\end{aligned}$$

Where the bounds follow from standard bounds on the Fourier coefficients of the majority function. By ensuring that the batch size $B_1 \geq \tilde{\Omega}(\tau_g^{-2})$, we see that for $\tau_g \leq O(k^{-1}d^{-5/3}\zeta_{k-1}^{-2}m_t^{-1})$ we see that $\mathbf{E}_{\mathbf{x},y}[(Af_t^{(1)})(\mathbf{x})\,\mathrm{Maj}(\mathbf{w} \odot \mathbf{x})x_i] = o(\frac{1}{m_t dk})$. □

E.2 STUDENT SAMPLE COMPLEXITY

Lemma E.7 (Student version of Theorem 4, Barak et al. (2022)). *Let a teacher f_t with hidden dimension $m_t = \Theta(2^{O(k)} \log(d)^2/\delta^2) \geq m_s^2\, k^4 \log(d)^2/\delta^2$ be trained in the setting of Lemma D.3. Furthermore, for student training Algorithm 2 $B_1 \geq k^2 d^2 \log(kd/\delta)$, $\eta_1 = m_t$. Then after $T_2 = \Omega(m_s d^2 k^3/\epsilon^2)$ steps of training with batch size $B_2 = 1$ and learning rate $\eta_2 = 4k^{1.5}/(dm(T_2 - 1))$, we have, with expectation over the randomness of the initialization and the sampling of the batches:*

$$\min_{t \in [T_2]} \mathbb{E}[L_{\theta(t)}(x, y)] \leq \epsilon.$$

Thus, the minimal sample complexity to reach a loss of ϵ for the student is given by:

$$T_1 \times B_1 + T_2 \times B_2 = \Theta(2^{O(k)} d^2 \epsilon^{-2} \log(k/\delta\epsilon)).$$

Proof. For the first stage, the sample complexity of the student is determined by two key factors: (1) the lower bound on m_t required to ensure that, after random projection, $(Af_t)_\ell$ for each $\ell \in [m_s]$ satisfies the conclusion of Lemma E.3, and (2) the sample complexity required to obtain a sufficiently precise gradient estimate so that the gap can be observed.

To ensure that the overall event occurs with probability δ, applying a union bound over the m_s coordinates to the conclusions of Lemmata 3.4 and E.3 results in a lower bound on the teacher's hidden dimension, given by $m_t \geq \frac{k^4 \log(d)^2 m_s^2}{\delta^2}$. As long as $m_t = \Theta(2^{O(k)} \log(d)^2/\delta^2)$, the rest of the argument follows.

By applying the conclusions of Lemmata 3.4 and E.3 to Lemma E.1, we know that the expected gradient has in-support coordinates of $\Omega(1/m_t k)$ and out-of-support coordinates of $O(1/m_t kd)$. After one step of gradient descent with an appropriate regularization parameter ($\lambda_1 = 1/2\eta$), we have $\mathbf{w}_i = -\eta\, \mathbf{E}_{\mathbf{x},y}\left[\nabla_{\mathbf{w}_i} \ell_{DL}(\mathbf{x}, f_s^{(1)}, Af_t^{(1)})\right]$.

The gradient estimate needs to be accurate up to an error of $O((kdm_t)^{-1})$. Setting $|\tau_g| = O(2^{-\Omega(k)} \log(k/d)^{-1})$ for the *student* in Claim D.1 ensures that $B_1 = O(2^{O(k)} \log(kd/\delta))$ is sufficient. We then set $\eta = m_t$ to ensure that the gap between the student weights matches the gap in Panigrahi et al. (2024a) and is bounded below by $\Omega(1/k^2)$.

The sample complexity for the second stage is exactly the same as in Lemma D.3, and the exponential dependence arises from there.

□

Remark: We observe that *even when the gap between the teacher's width and the student's width is only polynomial*, the teacher requires $\Omega(d^{k-1})$ samples, while the student only needs $\tilde{O}(2^{O(k)}\text{poly}(d, k))$ samples, since $m_t = \tilde{O}(2^{O(k)}\text{poly}(d, k))$. This difference arises because of the difference in the magnitude of the gap between the in-support and out-of-support coordinates of the gradient in these two cases.

F PROBABILISTIC CONTEXT-FREE GRAMMARS

In this section we formally define a PCFG.

Definition F.1 (Probabilistic Context-Free Grammar (PCFG)). *A Probabilistic Context-Free Grammar (PCFG) is a 5-tuple (N, Σ, S, R, P) where:*

- *N is a finite set of non-terminal symbols*
- *Σ is a finite set of terminal symbols ($N \cap \Sigma = \emptyset$)*
- *$S \in N$ is the distinguished start symbol*
- *$R \subseteq N \times (N \cup \Sigma)^*$ is a finite set of production rules*

- $P : R \to [0, 1]$ *is a probability function satisfying:*

$$\forall A \in N, \sum_{(A \to \beta) \in R} P(A \to \beta) = 1$$

The probability of a derivation tree T is given by:

$$P(T) = \prod_{(A \to \beta) \in T} P(A \to \beta)$$

In `cfg3b`, the PCFG is constructed such that the degree for every non-terminal A is 2. In any generation rule, consecutive pairs of symbols in the generated strings are distinct. The $25\%, 50\%, 75\%$, and 95% percentile string lengths generated by the PCFG are $251, 278, 308, and 342$, respectively, we refer to the commonly cited Figure 5 below from Allen-Zhu & Li (2023).

```
22 → 21 20
22 → 20 19
     19 → 16 17 18
     19 → 17 18 16
     20 → 17 16 18
     20 → 16 17
     21 → 18 16
     21 → 16 18 17
          16 → 15 13
          16 → 13 15 14
          17 → 14 13 15
          17 → 15 13 14
          18 → 15 14 13
          18 → 14 13
               13 → 11 12
               13 → 12 11
               14 → 11 10 12
               14 → 10 11 12
               15 → 12 11 10
               15 → 11 12 10
                    10 → 7 9 8
                    10 → 9 8 7
                    11 → 8 7 9
                    11 → 7 8 9
                    12 → 8 9 7
                    12 → 9 7 8
                         7 → 3 1
                         7 → 1 2 3
                         8 → 3 2
                         8 → 3 1 2
                         9 → 3 2 1
                         9 → 2 1
```

Figure 5: **cfg3b** from Allen-Zhu & Li (2023). Vocabulary is $\{1, 2, 3\}$. Indentation reflects production hierarchy.

We also we show similar performance gains to those we observe in Section 4 for experiments with larger bandwidth. In particular, for experiments with a total of 6000 and 8000 iterations respectively, with three and four-stage curricula.

G Miscellaneous Figures

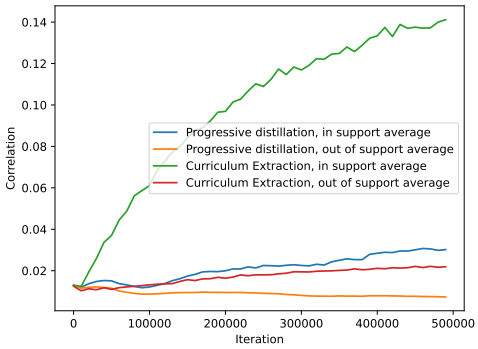

Figure 6: **MLP Projection vs Layer Correlation** We look at the magnitude of the correlations of the hidden layer weights of the depth-two MLP with the support of a 100-dimensional 6-sparse parity after the first phase of training. We observe that the curriculum extraction in-support coverage is significantly larger the out-of-support coverage, and with a significantly larger advantage than that for progressive distillation.

.