Sparse Logit Sampling: Accelerating Knowledge Distillation in LLMs

Anshumann* and Mohd Abbas Zaidi* and Akhil Kedia* and Jinwoo Ahn
Taehwak Kwon and Kangwook Lee and Haejun Lee and Joohyung Lee
Samsung Research, Seoul
{anshu.mann, abbas.zaidi, akhil.kedia, jinwoo.ahn, taehwak.kwon}@samsung.com

Abstract

Knowledge distillation can be a cost-effective technique to distill knowledge in Large Language Models, if the teacher output logits can be pre-computed and cached. However, successfully applying this to pre-training remains largely unexplored. In this work, we prove that naive approaches for sparse knowledge distillation such as caching Top-K probabilities, while intuitive, provide biased estimates of teacher probability distribution to the student, resulting in suboptimal performance and calibration. We propose an importance-sampling-based method 'Random Sampling Knowledge Distillation', which provides unbiased estimates, preserves the gradient in expectation, and requires storing significantly sparser logits. Our method enables faster training of student models with marginal overhead ($< 10\%$) compared to cross-entropy based training, while maintaining competitive performance compared to full distillation, across a range of model sizes from 300M to 3B.

1 Introduction

Distilling the knowledge from a larger teacher into a smaller student (Hinton et al., 2015) has been successfully used to train more efficient and stronger models across a range of applications (Fukuda et al., 2017; Jiao et al., 2020; Ahn et al., 2019; Tian et al., 2020; Sanh et al., 2019; Bergmann et al., 2020; Zhao et al., 2022; Xu et al., 2024b). As Large Language Models (LLMs) reach increasing adoption, Knowledge Distillation has also been applied to improve smaller LLMs (Sreenivas et al., 2024; Muralidharan et al., 2024; Gu et al., 2024; Wang et al., 2021; Gu et al., 2023; Palo et al., 2024; Boizard et al., 2024; Jiang et al., 2023).

Two common categories of Knowledge Distillation are distribution matching, where the teacher's final logits or output distribution are learned, and

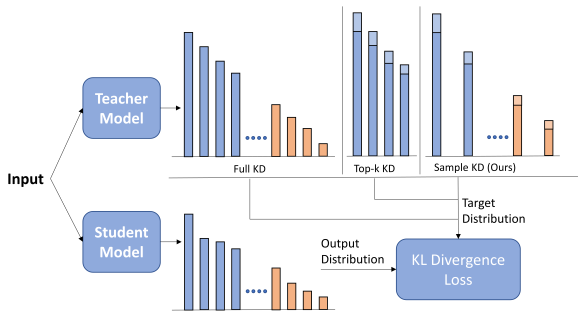


Figure 1: Sparse Knowledge Distillation Pipeline

representation matching, where intermediate-layer representations are distilled (Wen et al., 2023). In this work, we focus on the former, in a *offline logits* setting, where the logits from the teacher are pre-computed and cached, prior to training the student.

Particularly for LLMs, this setting has several advantages – The larger, more expensive teacher only needs to run once, and the saved representations can then be used to train a family of smaller models of various sizes. Teacher inference can be done on cheaper compute resources without fast multi-node interlinks, and the student can be trained on smaller clusters. Cluster size is further reduced by eliminating the memory footprint of the teacher. Lastly, this makes smaller-scale design experiments and ablations feasible by eliminating the constant large overhead of running the teacher model repeatedly for each experiment or training.

While this is often done for post-training (Shum et al., 2024) or for dataset generation/filtering (Gu et al., 2024; Wen et al., 2023; Gunasekar et al., 2023), extending this to pre-training is challenging. In contrast to vanilla pre-training, knowledge distillation requires the information-dense soft targets (teacher probabilities) to be stored. Due to the large vocabulary size of modern LLMs, naively storing all of these probabilities is infeasible (e.g., requiring 128 PetaBytes of storage for 1T tokens for Llama3 (Grattafiori et al., 2024)). Instead, sparse

*These authors contributed equally to this work

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 18085–18108

knowledge distillation approaches store an efficient Top-K subset of logits from the teacher's distribution (Raman et al., 2023; Peng et al., 2024). However, these methods still require a large number of logits (6400) to be stored, or even observe a *fall* in model performance (Peng et al., 2024).

In this work, via theoretical proofs, cross-validated by empirical analysis, we show that the performance drop in Top-K methods stem from two primary causes - 1) Top-K provides a biased estimator of the teacher's probability distribution, and 2) It fails to expose the tail of teacher's distribution to the student model. These result in the student learning a scaled-up and mis-calibrated distribution of the teacher probability.

We rectify both of these issues by instead utilizing importance sampling (Elvira and Martino, 2021) to randomly sample from the teacher's distribution. We show that our proposed Knowledge Distillation approach – 1) Provides an unbiased estimate of the teacher's probability distribution, 2) Preserves the gradient in expectation compared to full distillation, and 3) Eliminates the overhead of running the teacher inference, while maintaining model performance to full distillation, using extremely limited storage.

2 Top-K Knowledge Distillation

For storing KD logits, previous studies (Raman et al., 2023; Peng et al., 2024; Shum et al., 2024) have proposed to replace the full teacher distribution $\mathbf{t}$ in knowledge distillation with a sub-sampled version $\mathbf{t^s}$. The most intuitive way is to use only the top K probability values from the teacher ("Top-K KD"), specifically $t_i^s = t_i, i \in K$, and $t_i^s = 0$ otherwise, where t_i are the probabilities of the token i in $\mathbf{t}$. Note that $\sum t_i^s \neq 1$.

Theoretically, selecting the top K tokens results in the least error from the teacher distribution for a single token (Appendix A.3). This may be combined with "Top-p" which dynamically adjusts K to only keep a fixed probability mass p.

2.1 How Does Top-K perform compared to FullKD?

To study Top-K KD, we pre-train multiple LLaMA style 300M student models, while varying the number of probabilities used K. We train on web data using a well pre-trained 3B teacher (full hyper-parameters in Table 17), using forward KL-Divergence loss. As a baseline, we use a model trained with only Cross Entropy loss ("CE"), and as a ceiling, a model trained using the entire teacher distribution ("FullKD") to compare student performance on language modeling tasks.

As seen in the table Table 1, Top-K training lags behind the FullKD performance on the language modeling task. If a small number of Top-K tokens (< 25) are used, the student loss is worse than just than using CE loss without any distillation – Only after 300 tokens does the model performance start reaching close to FullKD. Using Top-p allows for the use of fewer tokens, but performance is still only 47% of FullKD.

We also measure the Expected Calibration Error (Guo et al., 2017) ("ECE") of these models, as prior works (Shum et al., 2024) have shown that calibration is strongly correlated with performance. Even though our teacher model is almost perfectly calibrated, we find that models trained with Top-K are strongly mis-calibrated, with calibration worsening as number of tokens (K) is being reduced. Models trained using CE and FullKD are almost perfectly calibrated, as has also been previously observed (Zhu et al., 2023; Shum et al., 2024; Hebbalaguppe et al., 2024).

Unique Tokens	LM Loss ↓	% CE to FullKD ↑	ECE %↓
CE	2.81	0%	1.2
3	3.04	−395%	10.6
5	2.96	−253%	7.7
12	2.87	−99%	4.7
25	2.82	−21%	3.2
50	2.80	5%	2.2
*50	2.78	47%	1.7
57	2.79	32%	2.0
100	2.77	55%	1.1
300	2.76	77%	1.5
FullKD	**2.75**	**100%**	**0.7**

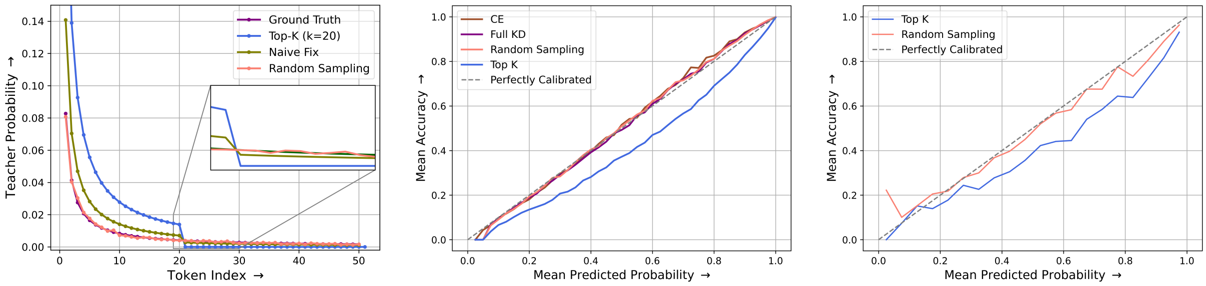
Table 1: Vanilla Top-K KD. The row *50 uses Top-p 0.98 with $K = 100$. '% CE to FullKD' refers to the % gap covered between CE and FullKD models.

2.2 Top-K KD Analysis

In this section, we demonstrate fundamental problems with Top-K methods.

2.2.1 Up-scaled Teacher Probabilities

Synthetic Toy Distribution: When only the Top-K values are kept from the teacher distribution, the probabilities of the top tokens are inevitably scaled-



(a) Visualizing Target Probabilities (b) Calibration on Synthetic Classes (c) Calibration on CIFAR-100

Figure 2: Comparing different sparse KD methods on synthetic examples (refer to Appendix B).

up compared to the original, as the probabilities must be normalized to sum to 1. We illustrate this in Figure 2a (see Appendix K for pseudo-code), where we simulate a synthetic distribution following a Zipf distribution (Kingsley, 1935). Similar bias was also observed in previous research (Zadeh and Schmid, 2021).

Gradients from KL-Divergence: When using KL-Divergence loss with Top-K KD, the non Top-K tokens are pushed to probability 0 due to restriction of the target distribution to Top-K probabilities. This happens even if one does not normalize the Top-K teacher probabilities. The backward gradients result in the student effectively learning an up-scaled version of the teacher probabilities as targets, with the remaining probability divided among the Top-K tokens. Specifically, if p_i and t_i are the student and teacher probabilities for the i^{th} token, the gradients for the i_{th} logit x_i in FullKD are:

$$\frac{\partial L}{\partial x_i} = p_i - t_i \tag{1}$$

But for Top-K KD, as we prove in Appendix A.4, the gradients are:

$$\frac{\partial L}{\partial x_i} = (\sum_{j \in K} t_j).p_i - t_i \tag{2}$$

The student will hence be over-confident in the Top-K tokens, and under-confident for the remaining tokens (Appendix A.4). This over-confidence for the top tokens is indeed what we observe with top-K pre-training for LLMs (Figure 3a), causing the calibration error in Table 1, which worsens as K is decreased. Other works (Busbridge et al., 2025) have also observed this top-K bias and miscalibration, while finding the full teacher distribution to be unbiased.

Synthetic Classification Task: This calibration error can even be observed in a very simple synthetic classification task (similar to Zhang et al., 2023), where we train a toy 3-layer MLP for classifying random points with Gaussian noise around class means in 128-dimensional space (see Appendix K for pseudo-code). As seen in Figure 2b, Top-K KD leads to over-confident models, whereas CE and FullKD are almost perfectly calibrated. The same effect is observed when training a toy ResNet (He et al., 2016) model on CIFAR-100 (Krizhevsky et al., 2009) dataset, as shown in Figure 2c.

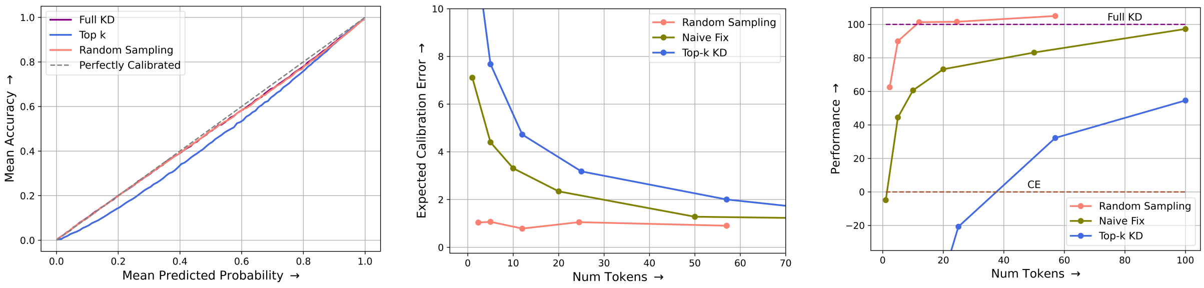
Hence, we cannot apply KL-Divergence loss on the Top-K target distribution without explicitly handling the remaining probability.

2.2.2 Missing Tail Information

However, only handling the problem of up-scaled teacher probabilities is not sufficient to fully recover the performance (Sections 3.1 and 3.2). In contrast to FullKD training, which utilizes the full distribution, Top-K KD discards the tail information which has been shown to be crucial for model performance (Shumailov et al., 2024). For rare ground truth tokens which fall in the tail of the teacher distribution, Top-K KD throws away the ground truth, providing a poor training signal compared to CE training. The tail, even though it contains a small probability mass, contains useful information and needs to somehow be learned.

3 Partial Empirical Solutions

In this section, we first discuss several empirical solutions to the problems discussed above. We apply these fixes to Top-K KD, and provide the corresponding results in Table 2.



(a) Calibration of Pre-trained models (b) Expected Calibration Error (c) LM Loss Performance

Figure 3: Comparing different sparse KD methods on Language Modeling Pre-Training

3.1 Label Smoothing

A straightforward solution is to distribute the residual probability over all the classes equally. Here, residual probability refers to $1 - p$ where p is the sum of the probabilities of the top-K tokens from the teacher's probability distribution. While this fixes the calibration error, smoothing leads to significant degradation in the performance compared to Top-K KD (Table 2). This is expected since real-world token probabilities are not uniformly distributed and are instead hyperbolic (Zipf Kingsley, 1935). While some studies show benefits of using smoothing (Menon et al., 2021), other works (Sultan, 2023; Shum et al., 2024) also find that label smoothing under-performs in KD.

Method	Top-k Loss ↓	New LM Loss ↓	% CE to FullKD ↑	ECE % ↓	0-shot Score ↑
CE	2.81	-	0	1.2	40.4
Smoothing	2.80	2.85	-73	0.4	41.2
Ghost Token	2.80	2.77	59	0.4	42.9
Naive Fix: Remaining Probability to Ground Truth					
Top-k 1	3.37	2.81	-5	7.1	41.3
Top-k 5	2.96	2.78	44	4.4	42.4
Top-k 10	2.88	2.77	61	3.3	42.4
Top-k 20	2.83	2.76	73	2.3	42.9
Top-k 50	2.80	2.76	83	1.3	42.8
Top-k 100	2.77	2.75	97	1.2	43.0
FullKD	**2.74**	-	**100**	0.7	42.1

Table 2: Naive Fixes for Top-K KD. Smoothing (Label Smoothing) and Ghost Token use 50 tokens.

3.2 Ghost Token

Another method to handle residual probability would be to create a "ghost token" which takes up the accumulated probabilities of non Top-K tokens for both the teacher and the student. We compute loss on the K top tokens, between predicted probabilities p_i and target $t_i^s = t_i$, and on the "ghost token" with probability $p_{\text{ghost}} = 1 - \sum_{i \in K} p_i$ and target $t^s_{\text{ghost}} = 1 - \sum_{i \in K} t_i$.

With the ghost token, the Top-K tokens receive the same gradient as FullKD, while the remaining tokens receive gradients proportional to the student confidence (Appendix A.5). This significantly improves both the LM loss and calibration (Table 2). However, the performance is still worse compared to FullKD – indicating that explicit supervision in the tail is essential to bridge the performance gap.

3.3 Naive Fix

A trivial candidate for the residual probability of the teacher is the ground truth itself. We label this method as "Naive Fix", where the probability of the target token is adjusted to ensure that the target probabilities sum up to 1. One can expect that this will result in probabilities more aligned to the real target (Figure 2a). This method significantly improves both performance and calibration error Table 2, however, it still requires 100 tokens to achieve performance comparable to FullKD.

The gradients for the logits are linked to the target teacher probability (Appendix A.1 - Equation (4)). The methods above are either biased estimators of the teacher probability distribution, and/or lack adequate supervision in the tail.

4 Proposed Method: Random Sampling KD

We propose a theoretically motivated method "Random Sampling KD", which overcomes all the drawbacks of the previous approaches. Given a teacher probability distribution $\mathbf{t_{full}}$ for each token i in the vocab V, unlike Top-K which truncates the teacher distribution, our method randomly samples tokens from teacher distribution.

Motivation For a given probability distribution $t(x)$, importance sampling (Elvira and Martino, 2021) allows us to obtain unbiased estimates of a function $f(x)$, by sampling from a different proposal distribution $q(x)$, and reweighing the samples using the likelihood ratio $t(x)/q(x)$.

$$E[f(x)] = \int f(x)t(x)dx = \int f(x)\frac{t(x)}{q(x)}q(x)dx$$

If the proposal $q(x) = 0$ at any x where $t(x) \neq 0$ (e.g., Top-K), then the estimate is no longer unbiased. While any non-zero proposal distribution $q(x)$ can be used to obtain an unbiased estimate, under certain constraints, the proposal distribution with the lowest variance $q^*(x)$ is of the form $q^*(x) \propto t(x)|f(x)|$ (Salakhutdinov, 2014). Motivated by these findings, we explore $q(x) = t(x)^\tau$ as a proposal distribution, where τ is the sampling temperature.

Sampling Distribution We sample tokens from $\mathbf{t_{full}}$, using the proposal distribution $\mathbf{q} = \mathbf{t}^\tau_{\mathbf{full}}$, for a fixed number of rounds N. Each occurrence of a token i is assigned a likelihood ratio $\frac{t_i}{q_i}$. Empirically, we find that for $0.8 < \tau < 1.2$, performance does not vary significantly (Table 12). We hence use $\tau = 1$, simply sampling N token ids from 1 to V (with replacement) with probability $\mathbf{t_{full}}$.

Obtaining Sampled Probabilities For each token, the likelihood ratio of each sample is added, and then normalized to obtain the sub-sampled target probability distribution $\mathbf{t^s}$. For $\tau = 1$, the likelihood ratio is simply 1, and t^s_i is then $\frac{c_i}{N}$, where c_i is the count of occurrences of each token i in N samples. This will be very sparse, with maximum N non-zero probabilities, and significantly less than N in practice (Appendix C).

Loss Calculation We use forward KL divergence between non-zero $\mathbf{t^s}$ and student predictions $\mathbf{p}$, $\sum t^s_i log\frac{t^s_i}{p_i}$. For $\tau = 1$, this may also be viewed as the sum of cross entropy loss between each sampled token and the student predictions.

This sub-sampled teacher distribution $\mathbf{t^s}$ can be stored/cached on disk and re-used across multiple experiments. The above gives us our final method, 'Random Sampling KD'.

5 Analysis of Random Sampling KD

5.1 Calibration

The toy distribution (Figure 2a) demonstrates that our method correctly estimates teacher distribution by providing an unbiased probability estimates, It achieves perfect calibration mirroring FullKD in the synthetic classification tasks (Figure 2b), in toy classification on CIFAR-100 (Figure 2c) and in LLM pre-training (Figure 3a).

As compared to the other KD methods discussed above, models trained with Random Sampling KD are much better calibrated, and using fewer tokens does not hurt the calibration (Figure 3b).

5.2 Gradient Similarity

In Appendix A.7, we prove that random sampling preserves the expected gradients at the logits when compared to FullKD. To further verify this empirically, we measure the gradients of the parameters of a 300M model trained with FullKD for one batch.

Method	Δ Angle $\downarrow$	Norm Ratio
Top-K 12	58°	2.4
Top-K 50	48°	1.8
Top-K 300	30°	1.3
Random Sampling 12	**4°**	**1.0**

Table 3: Comparing sparse KD gradients with FullKD

We find that the gradients from using Random Sampling are extremely similar to the gradients obtained from FullKD – with an angular difference of 4° and the same norm (cosine similarity of 0.998, and relative error of 0.07). Top-K methods on the other hand, have large angles and significantly different gradient norms even at 300 tokens, compared to just 12 unique tokens for Random Sampling.

5.3 Variance and Bias of Sampling Methods

While sampled distributions using Top-K have the least error for a single token, they inherently provide a biased estimate of the teacher distribution (Appendix A.3). This leads to the dissimilar gradients observed in Section 5.2. While our method is always unbiased, it is also crucial for the estimator to exhibit low variance (error). Lower variance will result in better approximation of the teacher distribution and hence better gradient approximation.

For example, using $\tau = 0$ in our proposal (sampling uniformly across the vocabulary) causes training to diverge, as the estimate is too noisy (Table 12). Similarly, using fewer tokens (with $\tau = 1$) will have higher error – but 12 tokens seems to be sufficient (Table 6), and hence we use 12 unique tokens in the rest of our experiments.

5.4 Speed/Throughput Comparison

In this section, we compare the speed in tokens/sec and TFlops for 300M / 3B student models with 3B / 8B teachers on 8 H100 GPUs. Our (RS-KD) caching implementation is 1.7 to 2.6 times faster than FullKD, and only slightly slower ($\approx$ 10%) than CE. This overhead stems from computing the loss over the entire vocabulary for distillation compared to a single ground truth token for CE.

Method	Tokens/sec ↑ 300M	Tokens/sec ↑ 3B	TFlops ↑ 300M	TFlops ↑ 3B
CE	2.9x	1.77x	330	544
Random Sampling	2.6x	1.73x	295	530
Full KD	1.0x	1.00x	100	304

Table 4: Speed/Throughput Comparison.

5.5 Storage Comparison

For CE training, storing raw UTF-8 text for 100B tokens requires $\approx$ 0.5TB storage for English (more for other languages). Storing tokenized data consumes 0.3TB, assuming 3 bytes per token. For FullKD storing the entire output distribution would requiring infeasible 10PB of storage, assuming 1byte for probability.

For sparse KD (KD) methods such as ours or Top-K, need to additionally store the Vocabulary Ids of the saved tokens. As detailed in Appendix D, we use 17 bits for Vocabulary IDs, and 7 bits for probabilities, totaling 24 bits (3 bytes) per unique token. As we require only 12 tokens (Table 6), we need only additional 3.6TB of space, 25x less than Top-300.

Method	Logits per Train Token	Bytes per Logit	Total Memory (TB)
Full KD	100 000	1	10 000.0
Top-K 300	300	3	90.0
Ours	12	3	3.6
Vanilla CE	1	3	0.3

Table 5: Storage Requirements for 100B train tokens

6 Results

Evaluation Tasks We evaluate our method across multiple metrics – LM loss on the pre-training dataset, Expected Calibration Error, the acceptance rate on speculative decoding of the teacher model, 0-shot NLU scores (settings detailed in Appendix E.1, full scores in Table 22) before and after Instruction Following training, and 0-shot NLG scores (settings detailed in Appendix E.3).

6.1 Small-Scale Results

We train LLaMA-style 300M student models using a 3B teacher (hyper-parameters in Table 17) for 10B tokens, 1.5x more than Chinchilla-optimal (Hoffmann et al., 2022) number of tokens. Our proposed method achieves very similar performance and calibration compared to FullKD, while using only 12 tokens (Table 6).

We also measure Speculative Decoding acceptance rate, as Top-1 agreement rate with the teacher has been shown to correlate with performance (Stanton et al., 2021). We find that our method again performs comparable to FullKD.

Somewhat surprisingly, as the number of unique tokens is increased, random sampling achieves marginally better performance compared to FullKD. Prior work has found that perturbing teacher logits results in better KD (Zhang et al., 2023), and we conjecture this sampling may achieve something similar.

Unique Tokens	LM Loss ↓	ECE % ↓	Speculative Accept % ↑	0-shot Score ↑
CE	2.81	0.4	59.95	40.4
2.4	2.77	1.0	61.47	42.1
5.0	2.75	1.1	61.83	42.6
12.1	2.75	0.8	61.85	43.0
24.5	2.75	1.1	61.93	**43.1**
57.0	**2.74**	0.9	61.97	42.9
FullKD	2.75	**0.7**	**62.02**	42.1

Table 6: Random Sampling KD (3B $\rightarrow$ 300M)

Effect of Longer Training On extending training of the student model for 100B tokens (16x Chinchilla-optimal), our model again achieves performance comparable to FullKD, both in speculative decoding and in 0-shot NLU scores (Table 7).

Method	LM Loss ↓	ECE % ↓	Speculative Accept % ↑	0-shot Score ↑
CE	2.48	0.7	64.6	45.0
Ours	2.48	**0.3**	65.7	**46.2**
FullKD	2.48	0.4	**65.8**	**46.2**

Table 7: Random Sampling KD 100B toks (3B$\rightarrow$300M)

6.2 Large-Scale Results

In order to replicate our findings with open-source LLMs on public datasets, we train student models using the LLaMA-3-8B model on the Fineweb-edu (Penedo et al., 2024) dataset.

First, we train a 3B LLaMA-style student using 100B tokens (Table 8). The loss gap between Top-K KD and FullKD is much higher in this regime. On the contrary, the student trained using "Random Sampling KD" (12 unique tokens) achieves similar loss, calibration and speculative decoding acceptance rate with significantly better downstream and instruction following performance. The improvements observed in our small-scale experiments persist for larger models with much longer training.

Method	LM Loss ↓	ECE % ↓	Speculative Accept % ↑	0-shot Score ↑	IF SFT Score ↑
CE	2.37	0.3	71.1	55.6	54.5
Top-K 12	2.50	4.7	73.0	56.6	57.7
Top-K 50	2.40	1.8	73.1	57.1	58.3
Ours (12)	2.35	**0.2**	73.2	57.5	**59.4**
Ours (12)+	**2.32**	1.7	**73.5**	**57.9**	59.1
FullKD	2.34	**0.2**	73.4	57.5	58.4

Table 8: Comparing sparse KD methods, 8B→ 3B 100B toks. The row 'Ours (12)+' is described in Section 6.3.

Evaluation with LLM-as-a-judge on Generative Tasks We also evaluate the 3B models using Llama 3.1 405B Instruct (Grattafiori et al., 2024) as a judge on five instruction following tasks. The student model trained with "Random Sampling KD" outperforms all other methods across all the evaluated tasks as seen in Table 9.

Dataset	CE	Top-K 12	Top-K 50	Ours 12	FullKD
Dolly	64.2	59.0	65.4	**71.3**	66.1
SelfInst	64.6	60.9	63.4	**73.1**	66.1
Vicuna	49.1	48.9	53.1	**58.2**	56.9
S-NI	62.4	63.4	62.6	**63.8**	60.7
UnNI	60.4	58.0	58.3	**61.4**	61.0
Avg	60.2	58.0	60.6	**65.6**	62.2

Table 9: Evaluations of 3B models on downstream generative tasks, with LLM-as-judge (8B → 3B)

Effect of Student Size We also vary the student sizes, training 100M, 300M, 1B and 3B all trained using LLaMA-3-8B as teacher, for 30x model-size tokens. The average performance on 0-shot downstream evaluations using "Random Sampling KD" over CE consistently improves as the student model size increases (Figure 4). While similar increasing trends have been previously observed for Top-K pre-training in Peng et al. (2024), they report a *fall* in performance for smaller student models. We conjecture that this may be attributed to the issues with Top-K KD we highlight in this work.

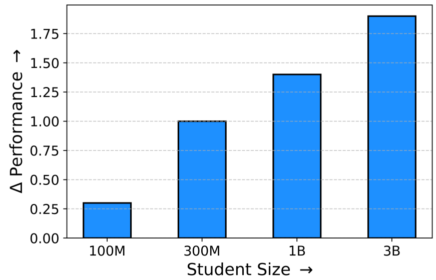


Figure 4: Downstream Improvements vs Student Size

6.3 Orthogonal Improvements to KD

Some orthogonal methods have also been proposed in the literature to improve the performance of FullKD. In this section, we show that these approaches can also be applied to "Random Sampling KD". Adding a combination of KLD and CE losses is often used during training (Gunter et al., 2024; Peng et al., 2024; Zhang et al., 2024) where the final loss is defined as $L = \alpha \cdot L_{CE} + (1-\alpha) \cdot L_{KLD}$ where α is the CE weight. Some prior works (Zhong et al., 2024; Zhao et al., 2022; Jiang et al., 2023; Palo et al., 2024) use different training modes for different tokens based on teacher's confidence/score in the target, where higher a score indicates that a token is easy to learn.

Setup We apply a similar adaptive method to our "Random Sampling KD" by categorizing tokens in a batch as "Easy" and "Hard" based on their target confidence percentile. Hard tokens use a higher learning rate (by a factor of "LR Ratio") compared to the easy tokens during training, while the average LR is kept constant. We train 300M models using a 3B teacher, and simultaneously vary the CE weight and the LR ratio together and report the '% CE to FullKD' metric.

Results As seen in Table 10, these methods enable "Random Sampling KD" to surpass FullKD. The best model is achieved with 0.1 CE weight and 2.0 LR Ratio. We further apply this approach to train a 3B student with 8B teacher for 100B tokens. This model (the row Ours (12)+ in Table 8), further improves on "Random Sampling KD" in LM loss, speculative acceptance rate, and 0-shot NLU scores.

Caveats However, this model does not improve as much after Instruction Tuning. We conjecture that up-weighing the "Hard" examples in the LR tends to effectively up-weigh the tail of the distribution. This was evidenced by the relatively higher calibration error of this model – we find that this model is *under-confident* in its predictions. While this improves the pre-training scores, it negatively impacts downstream fine-tuning of this model.

LR Ratio	CE Weight α			
	0.3	**0.2**	**0.1**	**0.0**
1.0	101	111	95	98
1.5	124	121	120	111
2.0	116	124	**125**	112

Table 10: '% CE to FullKD' with Orthogonal Improvements to Random Sampling KD (8B $\rightarrow$ 300M)

6.4 Comparison with Prior Works

In Table 11, we compare our sampling approach with those from prior works. For Raman et al. (2023), we use Top-50, and for Peng et al. (2024), Top-100 with $p = 0.98$. We also recreate these works including other sampling-orthogonal changes. Raman et al. (2023) uses a different LR for harder tokens, and adds the CE Loss to training. For Peng et al. (2024), we implement the temperature before softmax, and the WSD scheduler for the relative weight of CE and KD. Our method significantly outperforms these prior works.

Method	LM Loss ↓	%CE to FullKD ↑	ECE % ↓	Spec. Accept % ↑
CE	2.81	0%	1.2	60.0
Peng et al. (2024)*	2.78	−47%	1.7	61.9
Peng et al. (2024)	2.85	−78%	1.4	61.5
Raman et al. (2023)*	2.80	5%	2.2	61.9
Raman et al. (2023)	2.77	57%	1.9	61.0
Ours	**2.74**	**100%**	0.9	**62.0**
FullKD	2.75	**100%**	**0.7**	**62.0**

Table 11: Comparison with Prior Works. Rows marked with * only use the sampling method. (3B $\rightarrow$ 300M)

7 Ablations

7.1 Proposal Distributions

Choosing the optimal sampling temperature t can reduce the variance of the probability estimates, by allowing a trade-off between sampling more varied tokens, vs. obtaining more accurate estimates for higher-probability tokens. While this optimal temperature would depend on the exact shape of the distribution (and hence the teacher model), numerical simulations show that $t \in [0.8, 1.2]$ results in the lowest variance. The post-training performance of these was also comparable (Table 12).

While a better proposal distribution may be obtained following Optimal Experimental Design (Fedorov, 2013), our sampling method performs comparable to FullKD, hence for simplicity we choose proposal with $t = 1.0$ in this work.

Sample Temp	Unique Tokens	LM Loss ↓	ECE % ↓	0-shot Score ↑	Speculative Accept % ↑
0.0	57	∞	-	-	-
0.8	57	**2.74**	**0.7**	42.4	61.9
1.0	54	2.75	0.8	**43.0**	**61.8**
1.2	57	**2.74**	0.8	42.2	61.9

Table 12: Proposal Temperature Ablation (3B$\rightarrow$300M)

7.2 Effect of Adapting Teacher

Sreenivas et al. (2024) found that if the student is being trained on a data distribution different from the teacher's pre-training data, the teacher should first be adapted (finetuned) on this data by training for a short while. We also observe the same – when training a 300M student on Fineweb-edu data with the LLaMA-3-8B model as teacher, using the original teacher model directly yields only a small improvement over CE (Table 13). After teacher adaptation for 50B tokens, this increases significantly.

Method	LM Loss ↓	0-shot Score ↑
CE	2.99	40.1
KD w/o adapt	2.98	40.2
KD w adapt	**2.96**	**41.1**

Table 13: Adapting Teacher Model on Pre-training Dataset (8B $\rightarrow$ 300M)

7.3 Effect of Different Student Architecture

Our method is independent of the model architecture, and is equally applicable to other models such as Qwen (Team, 2024). Using the above LLama-3-8B as teacher, we train a 0.5B Qwen-style model (same architecture as Qwen2.5-0.5B) using our Random Sampling Method and with vanilla CE for 10B training tokens. Our method improves over CE as shown in Table 14.

Method	LM Loss ↓	Speculative Accept % ↑
CE	2.99	58.9%
Ours	**2.95**	**60.0%**

Table 14: Pre-training Qwen-style models (3B $\rightarrow$ 0.5B)

7.4 Choice of Loss/Divergence Function

We also experiment with alternative loss/divergence functions, by training 300M students with 8B Llama-3 teacher for 10B tokens. Some prior works (Kim et al., 2021; Wu et al., 2024b; Gu et al., 2023; Ko et al., 2024) find alternative objectives such as Reverse KL Divergence, Mean Squared Error as superior, while other works (Sultan, 2023; Wen et al., 2023; Muralidharan et al., 2024; Peng et al., 2024) have observed the opposite. In Table 15, we observe that vanilla forward KLD outperforms other objectives.

Metric	CE	L1	MSE	KLD		
				R	F+R	F
LM Loss ↓	2.81	∞	5.38	3.37	2.78	**2.75**

Table 15: Loss Ablation. F and R in KLD refer to forward and reverse KLD respectively.

8 Related Work

Knowledge Distillation (Hinton et al., 2015) has often been used to improve smaller LLMs (Jiao et al., 2020; Sanh et al., 2019; Sreenivas et al., 2024; Muralidharan et al., 2024; Wang et al., 2021; Gu et al., 2023; Boizard et al., 2024). Many works focus on using teacher models for dataset generation/filtering (Kim and Rush, 2016; Zhang et al., 2023; Wen et al., 2023; Gunasekar et al., 2023; Jiang et al., 2023; Gu et al., 2024; Palo et al., 2024). These methods are somewhat complementary to our method – our work is agnostic to the source of the pre-training data corpus, and focuses on distilling the teacher model's logits on this data.

Similar to our work, Shum et al. (2024) stores the Top-5 teacher probabilities from an LLM for training smaller students. They also observe that distillation with Top-K tokens leads to over-confident students – which they solve by employing temperature scaling. By sampling from the teacher distribution, our method offers a principled approach of achieving a calibrated student (Figure 3b). While they observe mis-calibration of their teacher as well, pretrained LLMs are well-calibrated, but alignment may degrade this calibration (Zhu et al., 2023; Hebbalaguppe et al., 2024). We find both our 3B as well as Llama 8B teachers well calibrated, as they are not instruction-tuned models.

Closest to our work are Raman et al. (2023), Peng et al. (2024) and Kamath et al. (2025). Raman et al. (2023) also observe that distillation improves student model performance – but they store Top-5% of the teacher logits, which is prohibitively large for modern LLMs (6400 for the Llama3 model) – we successfully bring this down to 12 logits in this work.

Peng et al. (2024) explores caching teacher logits in Knowledge Distillation in pre-training of LLMs utilizing Top-K with Top-p. They also conclude that forward KLD outperforms other objectives, adding CE loss improves distillation, and increasing performance improvement on scaling the model size and pre-training corpus. However, they observe a *fall* in performance on smaller students – vanilla Top-K may reduce model performance if K is not large enough as we show in Table 1. Our method remedies this issue, matching FullKD with significantly sparser tokens.

Contemporaneous work Gemma3 (Kamath et al., 2025) also used Knowledge Distillation for pretraining. Their method seems to be the same as our approach, sampling teacher logits weighed by original teacher probabilities, using cross-entropy loss on the sampled tokens. They successfully apply this method for training model up-to 27B params for 14T tokens, showing that our method can scale to very large models and tokens.

9 Conclusion

In this work, we identified key issues of bias and tail supervision with sparse teacher logits for Knowledge Distillation. We theoretically proved and empirically verified these claims in both synthetic and real-world scenarios, and proposed an importance-sampling based method to rectify them. By preserving gradients and logits distribution in expectation, we enable significantly sparser logits than prior methods. Our method maintains model performance while utilizing only 0.01% of pre-computed teacher logits, across a range of model sizes, training tokens, and evaluation metrics.

Limitations

Due to limited compute resources, we were only able to experiment upto 3B scale models trained for 100B tokens. Training longer with larger models should be explored, but our experiments indicate the benefits of our model only increase with model scale. Representation matching, which distills intermediate activations from the teacher, may improve distillation further. However, caching teacher representations due to limited compute resources was a primary requirement for this work, which rendered representation matching infeasible. Lastly, more sophisticated sampling schemes can also be explored, but we did not attempt that as our methods already achieved the desired outcome of matching full KD with low storage requirements.

References

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. 2023. On-Policy Distillation of Language Models: Learning from Self-Generated Mistakes.

Sungsoo Ahn, Shell Xu Hu, Andreas C. Damianou, Neil D. Lawrence, and Zhenwen Dai. 2019. Variational information distillation for knowledge transfer. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 9163–9171. Computer Vision Foundation / IEEE.

allenai. 2025. allenai/olmo-2-hard-coded · Datasets at Hugging Face.

Paul Bergmann, Michael Fauser, David Sattlegger, and Carsten Steger. 2020. Uninformed students: Student-teacher anomaly detection with discriminative latent embeddings. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 4182–4191. IEEE.

Yonatan Bisk, Rowan Zellers, Ronan LeBras, Jianfeng Gao, and Yejin Choi. 2020. PIQA: reasoning about physical commonsense in natural language. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7432–7439. AAAI Press.

Nicolas Boizard, Kevin El Haddad, Céline Hudelot, and Pierre Colombo. 2024. Towards Cross-Tokenizer Distillation: The Universal Logit Distillation Loss for LLMs.

Faeze Brahman, Sachin Kumar, Vidhisha Balachandran, Pradeep Dasigi, Valentina Pyatkin, Abhilasha Ravichander, Sarah Wiegreffe, Nouha Dziri, Khyathi Chandu, Jack Hessel, Yulia Tsvetkov, Noah A. Smith, Yejin Choi, and Hanna Hajishirzi. 2025. The Art of Saying No: Contextual Noncompliance in Language Models. *Advances in Neural Information Processing Systems*, 37:49706–49748.

Dan Busbridge, Amitis Shidani, Floris Weers, Jason Ramapuram, Etai Littwin, and Russ Webb. 2025. Distillation scaling laws. *ArXiv*, abs/2502.08606.

Haw-Shiuan Chang, Nanyun Peng, Mohit Bansal, Anil Ramakrishna, and Tagyoung Chung. 2024. REAL Sampling: Boosting Factuality and Diversity of Open-Ended Generation via Asymptotic Entropy.

Yevgen Chebotar and Austin Waters. 2016. Distilling knowledge from ensembles of neural networks for speech recognition. In *Interspeech 2016, 17th Annual Conference of the International Speech Communication Association, San Francisco, CA, USA, September 8-12, 2016*, pages 3439–3443. ISCA.

Zhihao Chi, Tu Zheng, Hengjia Li, Zheng Yang, Boxi Wu, Binbin Lin, and Deng Cai. 2023. NormKD: Normalized Logits for Knowledge Distillation.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90%* ChatGPT Quality.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have Solved Question Answering? Try ARC, the AI2 Reasoning Challenge.

Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. 2023. Free Dolly: Introducing the World's First Truly Open Instruction-Tuned LLM.

Jiequan Cui, Zhuotao Tian, Zhisheng Zhong, Xiaojuan Qi, Bei Yu, and Hanwang Zhang. 2023. Decoupled Kullback-Leibler Divergence Loss.

Víctor Elvira and Luca Martino. 2021. Advances in Importance Sampling.

V. V. Fedorov. 2013. *Theory Of Optimal Experiments*. Elsevier.

Takashi Fukuda, Masayuki Suzuki, Gakuto Kurata, Samuel Thomas, Jia Cui, and Bhuvana Ramabhadran. 2017. Efficient knowledge distillation from an ensemble of teachers. In *Interspeech 2017, 18th Annual Conference of the International Speech Communication Association, Stockholm, Sweden, August 20-24, 2017*, pages 3697–3701. ISCA.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, and 5 others. 2024. A framework for few-shot language model evaluation.

Jianping Gou, Baosheng Yu, Stephen J. Maybank, and Dacheng Tao. 2021. Knowledge Distillation: A Survey. *International Journal of Computer Vision*, 129(6):1789–1819.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and 542 others. 2024. The llama 3 herd of models.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. 2023. MiniLLM: Knowledge Distillation of Large Language Models. In *The Twelfth International Conference on Learning Representations*.

Yuxian Gu, Hao Zhou, Fandong Meng, Jie Zhou, and Minlie Huang. 2024. MiniPLM: Knowledge Distillation for Pre-Training Language Models.

Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, Adil Salim, Shital Shah, Harkirat Singh Behl, Xin Wang, Sébastien Bubeck, Ronen Eldan, Adam Tauman Kalai, Yin Tat Lee, and Yuanzhi Li. 2023. Textbooks Are All You Need.

Tom Gunter, Zirui Wang, Chong Wang, Ruoming Pang, Andy Narayanan, Aonan Zhang, Bowen Zhang, Chen Chen, Chung-Cheng Chiu, David Qiu, Deepak Gopinath, Dian Ang Yap, Dong Yin, Feng Nan, Floris Weers, Guoli Yin, Haoshuo Huang, Jianyu Wang, Jiarui Lu, and 136 others. 2024. Apple Intelligence Foundation Language Models.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. 2017. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1321–1330. PMLR.

Seungju Han, Kavel Rao, Allyson Ettinger, Liwei Jiang, Bill Yuchen Lin, Nathan Lambert, Yejin Choi, and Nouha Dziri. 2024. WildGuard: Open One-Stop Moderation Tools for Safety Risks, Jailbreaks, and Refusals of LLMs.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778. IEEE Computer Society.

Ramya Hebbalaguppe, Mayank Baranwal, Jatin Prakash, Neelabh Madan, Kartik Anand, and Chetan Arora. 2024. Understanding Calibration Transfer in Knowledge Distillation. *OpenReview Preprint*.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the Knowledge in a Neural Network.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katherine Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, and 3 others. 2022. An empirical analysis of compute-optimal large language model training. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Or Honovich, Thomas Scialom, Omer Levy, and Timo Schick. 2023. Unnatural instructions: Tuning language models with (almost) no human labor. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14409–14428, Toronto, Canada. Association for Computational Linguistics.

Tao Huang, Shan You, Fei Wang, Chen Qian, and Chang Xu. 2022. Knowledge distillation from A stronger teacher. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Brian Kenji Iwana, Ryohei Kuroki, and Seiichi Uchida. 2019. Explaining convolutional neural networks using softmax gradient layer-wise relevance propagation. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 4176–4185. IEEE.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. On using very large target vocabulary for neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1–10, Beijing, China. Association for Computational Linguistics.

Haozhe Ji, Pei Ke, Zhipeng Hu, Rongsheng Zhang, and Minlie Huang. 2023. Tailoring language generation models under total variation distance. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Liwei Jiang, Kavel Rao, Seungju Han, Allyson Ettinger, Faeze Brahman, Sachin Kumar, Niloofar Mireshghallah, Ximing Lu, Maarten Sap, Yejin Choi, and Nouha Dziri. 2024. WildTeaming at Scale: From In-the-Wild Jailbreaks to (Adversarially) Safer Language Models.

Yuxin Jiang, Chunkit Chan, Mingyang Chen, and Wei Wang. 2023. Lion: Adversarial distillation of proprietary large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3134–3154, Singapore. Association for Computational Linguistics.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. TinyBERT: Distilling BERT for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4163–4174, Online. Association for Computational Linguistics.

Gemma Team Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ram'e, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean-Bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, Gael Liu, and 191 others. 2025. Gemma 3 technical report. *ArXiv*, abs/2503.19786.

Taehyeon Kim, Jaehoon Oh, Nakyil Kim, Sangwook Cho, and Se-Young Yun. 2021. Comparing kullback-leibler divergence and mean squared error loss in knowledge distillation. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, pages 2628–2635. ijcai.org.

Yoon Kim and Alexander M. Rush. 2016. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327, Austin, Texas. Association for Computational Linguistics.

Zipf George Kingsley. 1935. *The psycho-biology of language: an introduction to dynamic philology*. Houghton Mifflin.

Jongwoo Ko, Sungnyun Kim, Tianyi Chen, and Se-Young Yun. 2024. DistiLLM: Towards Streamlined Distillation for Large Language Models.

Andreas Köpf, Yannic Kilcher, Dimitri von Rütte, Sotiris Anagnostidis, Zhi Rui Tam, Keith Stevens, Abdullah Barhoum, Duc Nguyen, Oliver Stanley, Richárd Nagyfi, Shahul ES, Sameer Suri, David Glushkov, Arnav Dantuluri, Andrew Maguire, Christoph Schuhmann, Huu Nguyen, and Alexander Mattick. 2023. Openassistant conversations - democratizing large language model alignment. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Alex Krizhevsky, Geoffrey Hinton, and 1 others. 2009. Learning multiple layers of features from tiny images. *" "*.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, and 4 others. 2024a. Tulu 3: Pushing Frontiers in Open Language Model Post-Training.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, and 4 others. 2024b. Tulu 3: Pushing Frontiers in Open Language Model Post-Training.

Jia Li, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Huang, Kashif Rasul, Longhui Yu, Albert Q Jiang, Ziju Shen, and others. 2024a. Numinamath: The largest public dataset in ai4maths with 860k pairs of competition math problems and solutions. *Hugging Face repository*, 13:9.

Peng Li, Yeye He, Dror Yashar, Weiwei Cui, Song Ge, Haidong Zhang, Danielle Rifinski Fainman, Dongmei Zhang, and Surajit Chaudhuri. 2024b. Table-GPT: Table Fine-tuned GPT for Diverse Table Tasks. *Proc. ACM Manag. Data*, 2(3).

Chengyuan Liu, Fubang Zhao, Kun Kuang, Yangyang Kang, Zhuoren Jiang, Changlong Sun, and Fei Wu. 2024. Evolving knowledge distillation with large language models and active learning. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 6717–6731, Torino, Italia. ELRA and ICCL.

Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. 2023. WizardCoder: Empowering Code Large Language Models with Evol-Instruct.

Aditya Krishna Menon, Ankit Singh Rawat, Sashank J. Reddi, Seungyeon Kim, and Sanjiv Kumar. 2021. A statistical perspective on distillation. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 7632–7642. PMLR.

Roy Miles and Krystian Mikolajczyk. 2024. Understanding the role of the projector in knowledge distillation. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational*

Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada, pages 4233–4241. AAAI Press.

Saurav Muralidharan, Sharath Turuvekere Sreenivas, Raviraj Joshi, Marcin Chochowski, Mostofa Patwary, Mohammad Shoeybi, Bryan Catanzaro, Jan Kautz, and Pavlo Molchanov. 2024. Compact Language Models via Pruning and Knowledge Distillation.

Team OLMo, Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, Nathan Lambert, Dustin Schwenk, Oyvind Tafjord, Taira Anderson, David Atkinson, Faeze Brahman, Christopher Clark, Pradeep Dasigi, Nouha Dziri, and 21 others. 2025. 2 OLMo 2 Furious.

Flavio Di Palo, Prateek Singhi, and Bilal H Fadlallah. 2024. Performance-Guided LLM Knowledge Distillation for Efficient Text Classification at Scale. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 3675–3687, Miami, Florida, USA. Association for Computational Linguistics.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1525–1534, Berlin, Germany. Association for Computational Linguistics.

Sangun Park. 2012. Generalized Kullback-Leibler information and its extensions to censored and discrete cases. *Journal of the Korean Data and Information Science Society*, 23(6):1223–1229.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. 2024. The fineweb datasets: Decanting the web for the finest text data at scale.

Hao Peng, Xin Lv, Yushi Bai, Zijun Yao, Jiajie Zhang, Lei Hou, and Juanzi Li. 2024. Pre-training Distillation for Large Language Models: A Design Space Exploration.

Dennis Prangle and Cecilia Viscardi. 2019. Distilling Importance Sampling for Likelihood Free Inference.

Nazneen Rajani, Lewis Tunstall, Edward Beeching, Nathan Lambert, Alexander M. Rush, and Thomas Wolf. 2023. No Robots.

Mrigank Raman, Pranav Mani, Davis Liang, and Zachary Lipton. 2023. For Distillation, Tokens Are Not All You Need. In *NeurIPS 2023 Workshop on Instruction Tuning and Instruction Following*.

Ruslan Salakhutdinov. 2014. Deep learning. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, page 1973. ACM.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter.

Arash Shahriari. 2017. Unified backpropagation for multi-objective deep learning. *ArXiv preprint*, abs/1710.07438.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-lm: Training multi-billion parameter language models using model parallelism.

KaShun Shum, Minrui Xu, Jianshu Zhang, Zixin Chen, Shizhe Diao, Hanze Dong, Jipeng Zhang, and Muhammad Omer Raza. 2024. FIRST: Teach A Reliable Large Language Model Through Efficient Trustworthy Distillation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 12646–12659, Miami, Florida, USA. Association for Computational Linguistics.

Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Nicolas Papernot, Ross Anderson, and Yarin Gal. 2024. Ai models collapse when trained on recursively generated data. *Nature*, 631(8022):755–759.

Shivalika Singh, Freddie Vargus, Daniel Dsouza, Börje F. Karlsson, Abinaya Mahendiran, Wei-Yin Ko, Herumb Shandilya, Jay Patel, Deividas Mataciunas, Laura OMahony, Mike Zhang, Ramith Hettiarachchi, Joseph Wilson, Marina Machado, Luisa Souza Moura, Dominik Krzemiński, Hakimeh Fadaei, Irem Ergün, Ifeoma Okoh, and 14 others. 2024. Aya Dataset: An Open-Access Collection for Multilingual Instruction Tuning.

Sharath Turuvekere Sreenivas, Saurav Muralidharan, Raviraj Joshi, Marcin Chochowski, Mostofa Patwary, Mohammad Shoeybi, Bryan Catanzaro, Jan Kautz, and Pavlo Molchanov. 2024. LLM Pruning and Distillation in Practice: The Minitron Approach.

Samuel Stanton, Pavel Izmailov, Polina Kirichenko, Alexander A. Alemi, and Andrew Gordon Wilson. 2021. Does knowledge distillation really work? In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 6906–6919.

Md Sultan. 2023. Knowledge Distillation \approx Label Smoothing: Fact or Fallacy? In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 4469–4477, Singapore. Association for Computational Linguistics.

Shangquan Sun, Wenqi Ren, Jingzhi Li, Rui Wang, and Xiaochun Cao. 2024. Logit Standardization in

Knowledge Distillation. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15731–15740.

Akio Suzukawa, Hideyuki Imai, and Yoshiharu Sato. 2001. Kullback-Leibler Information Consistent Estimation for Censored Data. *Annals of the Institute of Statistical Mathematics*, 53(2):262–276.

Qwen Team. 2024. Qwen2.5: A party of foundation models.

Yonglong Tian, Dilip Krishnan, and Phillip Isola. 2020. Contrastive representation distillation. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Takuma Udagawa, Aashka Trivedi, Michele Merler, and Bishwaranjan Bhattacharjee. 2023. A comparative analysis of task-agnostic distillation methods for compressing transformer language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pages 20–31, Singapore. Association for Computational Linguistics.

David Wadden, Kejian Shi, Jacob Morrison, Aakanksha Naik, Shruti Singh, Nitzan Barzilay, Kyle Lo, Tom Hope, Luca Soldaini, Shannon Zejiang Shen, Doug Downey, Hannaneh Hajishirzi, and Arman Cohan. 2024. SciRIFF: A Resource to Enhance Language Model Instruction-Following over Scientific Literature.

Abdul Waheed, Karima Kadaoui, and Muhammad Abdul-Mageed. 2024. To Distill or Not to Distill? On the Robustness of Robust Knowledge Distillation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12603–12621, Bangkok, Thailand. Association for Computational Linguistics.

Wenhui Wang, Hangbo Bao, Shaohan Huang, Li Dong, and Furu Wei. 2021. MiniLMv2: Multi-head self-attention relation distillation for compressing pretrained transformers. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 2140–2151, Online. Association for Computational Linguistics.

Xinpeng Wang, Leonie Weissweiler, Hinrich Schütze, and Barbara Plank. 2023a. How to distill your BERT: An empirical study on the impact of weight initialisation and distillation objectives. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1843–1852, Toronto, Canada. Association for Computational Linguistics.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023b. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13484–13508, Toronto, Canada. Association for Computational Linguistics.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Atharva Naik, Arjun Ashok, Arut Selvan Dhanasekaran, Anjana Arunkumar, David Stap, Eshaan Pathak, Giannis Karamanolakis, Haizhi Lai, Ishan Purohit, Ishani Mondal, Jacob Anderson, Kirby Kuznia, Krima Doshi, Kuntal Kumar Pal, and 16 others. 2022. Super-NaturalInstructions: Generalization via declarative instructions on 1600+ NLP tasks. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5085–5109, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022. Finetuned language models are zero-shot learners. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Yuqiao Wen, Zichao Li, Wenyu Du, and Lili Mou. 2023. f-divergence minimization for sequence-level knowledge distillation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10817–10834, Toronto, Canada. Association for Computational Linguistics.

Cindy Wu, Ekdeep Singh Lubana, Bruno Kacper Mlodozeniec, Robert Kirk, and David Krueger. 2024a. What Mechanisms Does Knowledge Distillation Distill? In *Proceedings of UniReps: The First Workshop on Unifying Representations in Neural Models*, pages 60–75. PMLR.

Siyue Wu, Hongzhan Chen, Xiaojun Quan, Qifan Wang, and Rui Wang. 2023. AD-KD: Attribution-driven knowledge distillation for language model compression. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8449–8465, Toronto, Canada. Association for Computational Linguistics.

Taiqiang Wu, Chaofan Tao, Jiahao Wang, Runming Yang, Zhe Zhao, and Ngai Wong. 2024b. Rethinking Kullback-Leibler Divergence in Knowledge Distillation for Large Language Models.

Sang Michael Xie and Stefano Ermon. 2019. Reparameterizable subset sampling via continuous relaxations. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pages 3919–3925. ijcai.org.

Wenda Xu, Rujun Han, Zifeng Wang, Long T. Le, Dhruv Madeka, Lei Li, William Yang Wang, Rishabh Agarwal, Chen-Yu Lee, and Tomas Pfister. 2024a. Speculative Knowledge Distillation: Bridging the Teacher-Student Gap Through Interleaved Sampling.

Xiaohan Xu, Ming Li, Chongyang Tao, Tao Shen, Reynold Cheng, Jinyang Li, Can Xu, Dacheng Tao, and Tianyi Zhou. 2024b. A Survey on Knowledge Distillation of Large Language Models.

Shekoufeh Gorgi Zadeh and Matthias Schmid. 2021. Bias in Cross-Entropy-Based Training of Deep Survival Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(9):3126–3137.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.

Rongzhi Zhang, Jiaming Shen, Tianqi Liu, Jialu Liu, Michael Bendersky, Marc Najork, and Chao Zhang. 2023. Do Not Blindly Imitate the Teacher: Using Perturbed Loss for Knowledge Distillation.

Songming Zhang, Xue Zhang, Zengkui Sun, Yufeng Chen, and Jinan Xu. 2024. Dual-Space Knowledge Distillation for Large Language Models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 18164–18181, Miami, Florida, USA. Association for Computational Linguistics.

Youcai Zhang, Zhonghao Lan, Yuchen Dai, Fangao Zeng, Yan Bai, Jie Chang, and Yichen Wei. 2020. Prime-Aware Adaptive Distillation.

Borui Zhao, Quan Cui, Renjie Song, Yiyu Qiu, and Jiajun Liang. 2022. Decoupled knowledge distillation. In *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, pages 11953–11962.

Wenting Zhao, Xiang Ren, Jack Hessel, Claire Cardie, Yejin Choi, and Yuntian Deng. 2024. WildChat: 1M ChatGPT Interaction Logs in the Wild. In *The Twelfth International Conference on Learning Representations*.

Qihuang Zhong, Liang Ding, Li Shen, Juhua Liu, Bo Du, and Dacheng Tao. 2024. Revisiting Knowledge Distillation for Autoregressive Language Models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 10900–10913, Bangkok, Thailand. Association for Computational Linguistics.

Helong Zhou, Liangchen Song, Jiajie Chen, Ye Zhou, Guoli Wang, Junsong Yuan, and Qian Zhang. 2021. Rethinking soft labels for knowledge distillation: A bias-variance tradeoff perspective. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Qinhong Zhou, Zonghan Yang, Peng Li, and Yang Liu. 2023a. Bridging the gap between decision and logits in decision-based knowledge distillation for pre-trained language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13234–13248, Toronto, Canada. Association for Computational Linguistics.

Yongchao Zhou, Kaifeng Lyu, Ankit Singh Rawat, Aditya Krishna Menon, Afshin Rostamizadeh, Sanjiv Kumar, Jean-François Kagy, and Rishabh Agarwal. 2023b. DistillSpec: Improving Speculative Decoding via Knowledge Distillation.

Chiwei Zhu, Benfeng Xu, Quan Wang, Yongdong Zhang, and Zhendong Mao. 2023. On the calibration of large language models and alignment. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 9778–9795, Singapore. Association for Computational Linguistics.

A Proofs

A.1 Backward Gradient through Softmax-KL Divergence Loss

The output probability $\mathbf{p}$ is defined in terms of the model's logits $\mathbf{x}$

$$\mathbf{p} = \text{Softmax}(\mathbf{x})$$
$$p_i = \frac{e^{x_i}}{\sum_{j=1}^{|V|} e^{x_j}}$$

The gradient through Softmax (Iwana et al., 2019) is:

$$\frac{\partial p_i}{\partial x_j} = p_i.(1\{i = j\} - p_j)$$

Given a target probability distribution $\mathbf{t}$, the KL divergence loss is defined as:

$$L = \sum_{i=1}^{|V|} t_i \log \frac{t_i}{p_i} \tag{3}$$

For Softmax-KL Divergence Loss, the gradient flowing to the j_{th} logit x_j can be calculated as follows:

$$\begin{aligned}\frac{\partial L}{\partial x_j} &= -\sum_{i=1}^{|V|} t_i \frac{1}{p_i}\frac{\partial p_i}{\partial x_j} \\ &= \sum_{i=1}^{|V|} t_i.(p_j - 1\{i = j\}) \\ &= (\sum_{i=1}^{|V|} t_i).p_j - t_j\end{aligned}$$

If the full teacher distribution is provided $\sum_{i=1}^{|V|} t_i = 1$. However, in the most generalized form, the gradient through Softmax-KL divergence loss can be written as:

$$\frac{\partial L}{\partial x_j} = (\sum_{i=1}^{|V|} t_i).p_j - t_j \tag{4}$$

A.2 Cross Entropy Loss

The cross entropy loss L defined as follows:

$$\begin{aligned} L_{CE} &= -\sum_{i=1}^{|V|} t_i \log p_i \\ &= L_{KLD} - \sum_{i=1}^{|V|} t_i \log t_i \end{aligned}$$

Compared to the KLD loss, the additional term $(\sum_{i=1}^{|V|} t_i log t_i)$ is independent of the student model. Hence, the gradient for CE loss remains the same as that computed for KL Divergence loss in Equation (3). For cross entropy (and similarly for FullKD with KLD loss), $\sum_{i=1}^{|V|} t_i = 1$. Hence, the gradient can be further simplified to:

$$\frac{\partial L}{\partial x_j} = p_j - t_j$$

In this case, the theoretical optima lies at the point where the predicted probabilities $\mathbf{p}$ become same as target probabilities $\mathbf{t}$ across the vocabulary, resulting in 0 gradient and minimum loss.

A.3 Vanilla Top-K has the Least L1 Error, but is a Biased Estimate

For a given distribution $\mathbf{t}$, if only K probabilities from $\mathbf{t}$ must be kept, and they are then normalized to sum to 1, we show that selecting the Top K probabilities results in the least L_1 error.

Let $\mathbf{K}$ be the set of tokens selected. Let $a = \sum_{j \in K} t_j$. This can be viewed as constructing a new distribution $\mathbf{v}$, where normalizing the probabilities

$$\begin{aligned} v_i &= \frac{t_i}{a}, i \in K, \\ v_i &= 0, i \notin K \end{aligned}$$

Then the L_1 error between $\mathbf{t}$ and $\mathbf{v}$ is

$$\begin{aligned} L_1 &= \sum_i |t_i - v_i| \\ &= \sum_{i \in K} |t_i - t_i/a| + \sum_{i \notin K} |t_i - 0| \\ &= (1/a - 1) * \sum_{i \in K} t_i + (1 - \sum_{i \in K} t_i) \\ &= (1/a - 1) * a + (1 - a) \\ &= 2 * (1 - a) \end{aligned}$$

Hence L_1 will be minimized when a is the largest, which will happen when the K largest probabilities are selected.

However, note that this gives us a biased estimate, as $E[v_i] = 0 \neq E[t_i], i \notin K$.

A.4 Vanilla Top-K KD provides scaled teacher as target

We can restrict the target probability to a subset of tokens in our vocabulary. If we select $\mathbf{K}$ as the set of tokens with top-k probabilities, then the loss is defined as follows:

$$L = \sum_{i \in K} t_i \log \frac{t_i}{p_i}$$

This can be viewed as zeroing out the non-top-k target probabilities in the original KLD loss. In this case, the gradient flowing to the logits are (Equation (4)):

$$\frac{\partial L}{\partial x_j} = (\sum_{i \in K} t_i).p_j - t_j \tag{5}$$

If $j \notin K$, the gradient is $(\sum_{i \in K} t_i).p_j$. As opposed to the previous case, model's optima lies at the point where non-top-k probabilities are 0 and hence the student is **under-confident** in the non-top-k probabilities. Similarly, the top-k predicted probabilities $\mathbf{p}$ are a scaled up version of the target probabilities $\mathbf{t}$ across the top-k tokens, $p_i = \frac{t_i}{(\sum_{j \in K} t_j)}$, hence making the student **over-confident** in top-k probability predictions. At this optima, the gradient is 0 (but the loss is negative).

A.5 Ghost Token Backward

One possible solution to the above discussed problem is to add a ghost token which accounts for the remainder of the probability. This ghost token ensures that the sum of probability outside the top-k region is exactly the same for the teacher and student. Ideally, it would ensure that the top-k tokens receive the exact teacher probability as the target. The modified loss function is written below-

$$L = \Bigg(\sum_{i \in K} t_i \log \frac{t_i}{p_i} + (1 - \sum_{i \in K} t_i) log \Big(\frac{1 - \sum_{i \in K} t_i}{1 - \sum_{i \in K} p_i} \Big) \Bigg)$$

Let us consider the second term in the loss and find its gradient

$$L_{ghost} = (1 - \sum_{i \in K} t_i) log \Big(\frac{1 - \sum_{i \in K} t_i}{1 - \sum_{i \in K} p_i} \Big)$$

$$\frac{\partial L_{ghost}}{\partial x_j} = \Big(\frac{1 - \sum_{i \in K} t_i}{1 - \sum_{i \in K} p_i} \Big) . \sum_{i=1}^{k} \frac{\partial p_i}{\partial x_j}$$

$$= \Big(\frac{1 - \sum_{i \in K} t_i}{1 - \sum_{i \in K} p_i} \Big) . \sum_{i=1}^{k} p_i . (1\{i = j\} - p_j)$$

The gradient becomes:

$$\frac{\partial L_{\text{ghost}}}{\partial x_j} = \begin{cases} \left(1 - \sum_{i \in K} t_i\right) p_j & j \in K, \\ -\left(\frac{1 - \sum_{i \in K} t_i}{1 - \sum_{i \in K} p_i}\right) p_j \sum_{i \in K} p_i & \text{else.} \end{cases}$$

Next we can add the gradient from top-k KD loss Equation (5) and ghost token loss to obtain the final gradient

$$\frac{\partial L}{\partial x_j} = \begin{cases} (p_j - t_j) & j \in K, \\ \left(\frac{\sum_{i \in K} (t_i - p_i)}{1 - \sum_{i \in K} p_i}\right) p_j & \text{else.} \end{cases}$$

For the non top-k tokens, the gradients can be rewritten as

$$\frac{\partial L_{\text{ghost}}}{\partial x_j} = p_j - \Big(\frac{1 - \sum_{i \in K} t_i}{1 - \sum_{i \in K} p_i} \Big) p_j \notin K$$

By adding the ghost token, the top-k tokens get the same gradient as KLD loss with FullKD, while the remaining tokens receive gradient in proportion of their predicted probability p_i. The target probability for non top-k tokens is $\left(\frac{1 - \sum_{i \in K} t_i}{1 - \sum_{i \in K} p_i}\right) p_j$. In this case, if the predicted probability distribution is exactly the same as that of teacher probability only for top-k tokens, the gradient becomes 0 and loss becomes minimum.

A.6 Random Sampling KD provides Unbiased Estimates

Our method Random Sampling KD uses importance sampling. By definition, importance sampling estimator is an unbiased estimator (Elvira and Martino, 2021). We provide a short intuition of this below for temperature $t = 1$.

We sample token ids N times with replacement from proposal distribution $q_i = p_i$.

Each occurrence is assigned a likelihood ratio $\frac{p_i}{q_i} = 1$, and then normalized by dividing by N.

The expected counts of token i will then be $\frac{q_i * N}{N} = q_i = p_i$. Hence this sampling is unbiased.

A.7 Unbiased Sampling preserves gradients in expectation

For any partial knowledge distillation scheme which sub-samples the full distribution, the expected gradients at the logits will be preserved in expectation if sampling is unbiased.

Proof: The gradient g_j for the logit x_j through the softmax-KL divergence loss is (replacing $\sum_{i=1}^{|V|} t_i = 1$ in Equation (4)))

$$g_j = p_j - t_j \tag{6}$$

Taking expectations on both sides

$$E[g_j] = E[p_j] - E[t_j]$$

Similarly, for a sub-sampling method which reduced $\mathbf{t} \to \mathbf{t}^{\mathbf{s}}$, expected gradient is as follows

$$E[g_j^s] = E[p_j] - E[t_j^s]$$

The gradients at the logits are preserved in expectation if $E[t_j] = E[t_j^s]$ and the sub-sampling process is unbiased.

B Synthetic Examples

Visualizing Target Probabilities We generate a Zipf distribution where the probability of i^{th} token is proportional $\frac{1}{i}$. Next we select tokens and assign them probabilities based on different sparse knowledge distillation methods. We plot these probabilities with the ground truth FullKD probabilities to visualize the alignment of sparse KD target distributions with FullKD.

Calibration on Synthetic Classes As discussed in the main paper and the psuedocode (Appendix K), we generate synthetic data by generating random points around randomly chosen class means with Gaussian error distribution. We use a simple 3-layer MLP as our model. We train the model using different sparse KD techniques and FullKD and plot the mean accuracy after binning the probabilities.

Calibration on CIFAR-100 We follow the exact same methodology as the synthetic classification while using CIFAR-100 task and a weaker/smaller version of ResNet-18 model.

C Number of Sampling Rounds for Given Number of Effective Tokens

For a fair comparison between Top-K KD and random sampling methods, the number of sampling rounds N were chosen such that the number of unique tokens sampled match K. This will be specific to the dataset and the teacher model. For example, $N = 50$, we find $K = 12$. The relationship between the two for pre-training data is shown in Figure 5 (log-log scale), and is almost perfectly linear, showing an approximate power-law relationship.

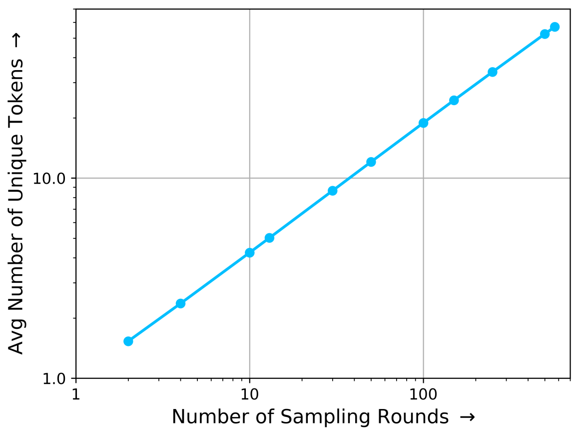


Figure 5: Number of unique tokens sampled vs sampling rounds

D Implementation Concerns

D.1 Quantization for Teacher Probabilities

For our vocab size $V = 100000$, our token ids require $log_2(V) = 17$ bits. We store the byte-aligned data, which leaves us with $24 - 17 = 7$ bits for teacher probabilities. As probabilities are in range $0..1$, for Top-K method, we use the 7 bits to split the 0 to 1 range into 2^7 equal intervals. This resulted in slightly lower performance compared to storing the probabilities in fp16. Instead switching to ratio encoding with sorted Top-K probabilities resulted in significantly reduced quantization error to almost 0, and results matched that of using unquantized probabilities.

In the case of our proposed random sampling, we use 50 sampling rounds, so token probabilities can only be of the form $x/50$, where x is some integer. As this is less than 2^7, we can store all of these exactly in 7 bits by only storing the numerator. If sampling rounds are increased beyond 128, ratio encoding with sorted probabilities can be used instead.

D.2 Efficiency Concerns

Naively implementing the sampling and the loss calculation incurred significant memory usage, due to the large vocabulary size. Manual backward and forward for the softmax KLD needed to implemented (via plain Pytorch, custom kernels were not created). Writing and reading the logits needed to be streamlines via shared memory ring buffers and async writer processes, so as to not block the GPU.

D.3 Aligning Teacher and Student Sequences

In our pre-training, we pack shuffled training documents to maximum sequence length, but we do not mask attention across document boundaries due to efficiency reasons. In our initial implementation, different shuffling seed was used between the teacher (during inference)and student (during training) – This resulted in the prefix-context of tokens seen by the teacher and student not being aligned after the first document boundary. This had a surprisingly large effect on student model performance, particularly if smaller sequence lengths were used during teacher inference. We conjecture that with longer sequence lengths, far-away tokens from other documents will have less of an impact on the distribution of the logits. After fully aligning the teacher and student sequences, this effect was eliminated, and the offline run was within random error of the online run.

E Downstream Evaluation Details

E.1 Natural Language Understanding

We evaluate the downstream natural language understanding performance of our trained models us-

Shuffle Seeds	Seq Len	LM Loss	% CE to online
Different	1024	2.760	79
Different	4096	2.753	90
Same	4096	**2.749**	**96**

Table 16: Effect of aligning teacher and student sequences, with different/same shuffle seeds and sequence length of the teacher during inference. The last column shows the performance of the offline (cached) implementation relative to an online implementation, where the entire teacher model is run.

ing the following benchmarks: HellaSwag (Zellers et al., 2019), Arc-Easy (Clark et al., 2018), LAMBADA (Paperno et al., 2016), and PiQA (Bisk et al., 2020). We conduct zero-shot evaluation of all benchmarks using LM-Eval-Harness (Gao et al., 2024). In the main paper, we report the average scores obtained across these tasks, and full scores are provided in Table 22.

E.2 Supervised Finetuning for Instruction Following

We used the Olmo2 (OLMo et al., 2025) version of the Tulu (Lambert et al., 2024a) Instruction Following dataset for SFT training after Language Modeling pre-training.

E.3 Instruction Following Evaluation

Similar to Gu et al. (2023), we evaluate the ability of fine-tuned models to follow instructions on five datasets:

- **DollyEval** (Conover et al., 2023): 15k human-written instruction-response pairs. Following Gu et al. (2023), we use the 500-sample test set for evaluation.
- **SelfInst** (Wang et al., 2023b): A user-oriented instruction following dataset containing 252 samples.
- **VicunaEval** (Chiang et al., 2023): 80 diverse and challenging question-answer pairs.
- **S-NI**: The test set of Supernatural Instruction (Wang et al., 2022). We sample 1694 pairs whose ground-truth response length is longer than 11.
- **UnNI**: A 10k subset of Unnatural Instruction (Honovich et al., 2023). Similar to S-NI, we only use pairs where the ground-truth length is longer than 11.

We adopt the LLM-as-a-Judge approach, where we use Llama 3.1 405B Instruct (Grattafiori et al., 2024) to score the quality of model responses. For each instruction, we generate the response five times using different seeds and temperature = 1. We prompt the judge model to rate both the ground-truth response and the model-generated response on a scale of 1-10, and use the average ratio of the total score of the ground-truth and model-generated responses as the final score.

F Hyper-parameters

The hyper-paramerters for our experiments are described in Tables 17, 19 and 20 and Appendix F

Parameters	Values
Optimizer	Adam
β_1, β_2	$0.9, 0.95$
Effective Batch Size	1024
Drop-out (p)	0.0
Sequence Length	1024
Train Iters	$10,000$
Learning rate	$4 * 10^{-4}$
Schedule	Cosine / Constant
LR Decay Iterations	100%
Warmup steps	4%
Min LR	$4 * 10^{-5}$
Gradient clipping	1.0

Table 17: Pre-Training Hyper-Parameters for 300M model. The pre-training dataset was web data, primarily Fineweb-Edu.

G Package versions

Versions of packages used are described in Table 21.

H Computational Resources

All experiments were carried out on nodes with 8 Nvidia H100 GPUs with 80Gb memory. Most experiments utilized one node or less, while the large scale ones used $2-4$ nodes.

I Use of AI Assistants

AI assistants were consulted while writing a small fraction of the code for this work, but their work was carefully checked, and the majority of the code was handwritten. AI assistants were not used in writing the text of this paper.

Parameters	Values
Optimizer	Adam
β_1, β_2	$0.9, 0.95$
Effective Batch Size	1024
Drop-out (p)	0.0
Sequence Length	4096
Train Iters	$10,000$
Learning rate	$3 * 10^{-4}$
Schedule	Cosine
LR Decay Iterations	100%
Warmup steps	4%
Min LR	$3 * 10^{-5}$
Gradient clipping	1.0

Table 18: Training Hyper-Parameters for 3B Llama model

Parameters	Values
Optimizer	Adam
β_1, β_2	$0.9, 0.95$
Effective Batch Size	256
Drop-out (p)	0.0
Sequence Length	4096
Train Iters	$1,234$
Learning rate	$2 * 10^{-5}$
Schedule	Cosine
LR Decay Iterations	100%
Warmup steps	3%
Min LR	$2 * 10^{-6}$
Gradient clipping	1.0

Table 19: SFT Hyper-Parameters for 3B Llama model

J Artifacts

We use LLaMA-3-8B (Grattafiori et al., 2024) as the teacher for some of experiments. We also used the Llama-3.1-405b as a judge for evaluation. Both of these uses are permitted under the license of these models. The datasets used here are also permitted for research use, and were only used for research. The pre-training dataset Fineweb-Edu (Penedo et al., 2024) is primarily composed of English educational-style web data, and so is the SFT data Tulu (Lambert et al., 2024a).

Parameters	300M Model	3B Model
Num Layers	24	28
Hidden Size	1024	3072
FFN Hidden Size	2816	8192
Num Attn Heads	8	24
Num Query Groups	8/4	8

Table 20: Student Model Architecture Details. The 100B experiments for 300M model used 4 query groups for efficiency. The pre-training dataset was FineWeb-Edu (Penedo et al., 2024)

Package	Version
megatron	0.7.0
deepspeed	0.15.3
flash_attn	2.4.2
safetensors	0.4.5
scikit-learn	1.5.2
scipy	1.14.0
sentencepiece	0.2.0
torch	2.5.0
transformer_engine	1.11.0
transformers	4.46.1

Table 21: Package Versions for Pre-training

K Pseudo-code

The pseudocode for topk sampling and random sampling approaches is provided below.

```python
import torch

## Create downsampled probabilities
def create_prob(values, indices, probs):
    downsampled_probs = torch.zeros_like(probs)
    downsampled_probs.scatter_(1, indices, values)
    return downsampled_probs

## Downsampling Functions
def downsample_topk(probs, k=50):  # Top-k
    topk_values, topk_indices = probs.topk(k)
    return create_prob(topk_values, topk_indices, probs)

def downsample_ours(probs, N=50):  # Sampling
    sampled_indices = torch.multinomial(probs, N, replacement=True)
    prob_value = 1.0 / N
    values = torch.full((probs.size(0), N), prob_value, device=probs.device)
    return create_prob(values, sampled_indices, probs)

## Knowledge distillation loss
def distillation_loss(student_logits, teacher_probs, downsample_fn):
    # Downsample teacher distribution
    downsampled_teacher_probs = downsample_fn(teacher_probs)

    # Compute KL divergence
    loss =  torch.nn.functional.kl_div(
         torch.nn.functional.log_softmax(student_logits, dim=-1),
        downsampled_teacher_probs,
    )
    return loss

## Training step
def train_step(inputs, labels, teacher_model, student_model, downsample_fn, alpha=0.5):
    # Forward pass through teacher and student
    with torch.no_grad():
        teacher_logits = teacher_model(inputs)
        teacher_probs =  torch.nn.functional.softmax(teacher_logits, dim=-1)

    student_logits = student_model(inputs)

    # Compute standard cross-entropy loss
    ce_loss =  torch.nn.functional.cross_entropy(student_logits, labels)

    # Compute distillation loss
    kd_loss = distillation_loss(student_logits, teacher_probs, downsample_fn)

    # Combine losses
    total_loss = alpha * kd_loss + (1 - alpha) * ce_loss

    return total_loss
```

The pseudocode for running different sampling strategies on a toy distribution.

```python
# Set random seed for reproducibility
np.random.seed(12345)

# Configuration parameters
VOCAB_SIZE = 100000
TOP_K = 20
NUM_SAMPLES = 22
NUM_SAMPLING_ROUNDS = 1000
Y_MAX = 50

# Create synthetic data distribution
def create_synthetic_data(vocab_size):
    idx = np.array(range(1, vocab_size + 1))
    data_dist = 1 / idx
    data_dist /= np.sum(data_dist)  # Normalize to sum to 1
    return idx, data_dist

# Generate data
idx, data_dist = create_synthetic_data(VOCAB_SIZE)

# Top-K method
def apply_top_k(data_dist, idx, top_k):
    top_k_probs = data_dist[:top_k]
    top_k_probs_redistributed = top_k_probs / np.sum(top_k_probs)
    # top_k_probs_redistributed = top_k_probs

    # Create top-k distribution with a small offset for visualization
    top_k_dist = np.zeros_like(data_dist)
    top_k_dist[:top_k] = top_k_probs_redistributed
    top_k_dist = list(top_k_dist[:top_k]) + [0] + list(top_k_dist[top_k:])
    return top_k_dist

data_dist_top_k = apply_top_k(data_dist, idx, TOP_K)

# Naive fix method
def apply_naive_fix(data_dist, idx, top_k):
    naive_fix_dist = np.zeros_like(data_dist)
    naive_fix_dist[:top_k] = data_dist[:top_k]
    naive_fix_dist += data_dist * (1 - np.sum(naive_fix_dist))
    return naive_fix_dist

data_dist_remaining_gt = apply_naive_fix(data_dist, idx, TOP_K)

# Random sampling method
def apply_random_sampling(data_dist, num_samples, num_rounds):
    random_sampling_dist = np.zeros_like(data_dist)
    num_samples_effective = 0

    for _ in range(num_rounds):
        current_dist = np.zeros_like(data_dist)
        samples = np.random.choice(len(data_dist), size=num_samples, p=data_dist)
        for i in samples:
            current_dist[i] += 1
        num_samples_effective += np.count_nonzero(current_dist)
        current_dist /= num_samples
        random_sampling_dist += current_dist

    num_samples_effective /= num_rounds
    random_sampling_dist /= np.sum(random_sampling_dist)
    return random_sampling_dist, num_samples_effective

data_dist_random_sampling, num_samples_effective = apply_random_sampling(data_dist, NUM_SAMPLES, NUM_SAMPLING_ROUNDS)


def plot_probability_distributions(LINE_WIDTH=2.0, MARKER_SIZE=3):
    plt.plot(idx[:Y_MAX], data_dist[:Y_MAX], label='Ground Truth', color='purple', linewidth=LINE_WIDTH, marker='o', markersize=MARKER_SIZE)

    # Plot Top-K distribution
    idx_topk = list(idx[:TOP_K]) + list(idx[TOP_K:])
    data_dist_top_k_truncated = list(data_dist_top_k[:TOP_K])  + list(data_dist_top_k[TOP_K:])
    plt.plot(idx_topk[:Y_MAX+1], data_dist_top_k_truncated[:Y_MAX+1],
             label='Top-K (k=20)', color='royalblue', linewidth=LINE_WIDTH, marker='o', markersize=MARKER_SIZE)

    plt.plot(idx[:Y_MAX], data_dist_remaining_gt[:Y_MAX],
             label='Naive Fix', color='darkgoldenrod', linewidth=LINE_WIDTH, marker='o', markersize=MARKER_SIZE)

    plt.plot(idx[:Y_MAX], data_dist_random_sampling[:Y_MAX],
             label='Random Sampling', color='salmon', linewidth=LINE_WIDTH, marker='o', markersize=MARKER_SIZE)

    # Add plot details
    plt.ylim(-0.002, 0.15)
    plt.legend(fontsize=12, framealpha=0.6)
    plt.xticks(fontsize=11)
    plt.yticks(fontsize=11)
    plt.grid()
    plt.xlabel(r'Token Index $\rightarrow$', fontsize=14)
    plt.ylabel(r'Teacher Probability $\rightarrow$', fontsize=14)
    plt.savefig("images/image.png", dpi=600, bbox_inches='tight')

plot_probability_distributions()
print(f"Effective number of samples: {num_samples_effective:.2f}")
```

The pseudocode for running different top-k strategies on a synthetic classification task.

```
torch.random.manual_seed(1234)
torch.set_default_dtype(torch.float64)
device='cuda'
num_classes = 1024
sigma = 1.5
num_dim = 128
num_hidden_teacher = 128
num_hidden_student = 96
class_centers = torch.rand((num_classes, num_dim), device=device)
class_sigma = torch.unsqueeze(torch.rand((num_classes, ), device=device), dim=-1) * sigma
class_indices = torch.tensor(range(num_classes), device=device)
num_calibration_batches = 100

def get_batch(batch_size=4096):
    idx = torch.randint(low=0, high=num_classes, size=(batch_size,), device=device)
    class_centers_batch = class_centers[idx]
    class_sigma_batch = class_sigma[idx]
    batch = class_centers_batch + torch.randn((batch_size, num_dim), device=device)*class_sigma_batch
    return batch, idx

def eval(model, method):
    all_probs = []
    all_acc = []
    with torch.no_grad():
        for i in tqdm(range(num_calibration_batches)):
            model.eval()
            batch, labels = get_batch()
            probs = model(batch)
            probs = torch.nn.functional.softmax(probs, dim=-1)
            all_probs.append(torch.max(probs, dim=-1)[0])
            all_acc.append(torch.argmax(probs, dim=-1).detach() == labels)
    all_probs = torch.vstack(all_probs)
    all_acc = torch.vstack(all_acc)
    print(f'Accuracy for {method}', all_acc.float().mean().item()*100)

def train(model, method, teacher=None, lr=2e-3, num_rounds=20000, **kwargs):
    optimizer = torch.optim.AdamW(params = model.parameters(), lr=lr, weight_decay=0.00)
    for step in tqdm(range(num_rounds)):
        optimizer.zero_grad()
        batch, labels = get_batch()
        logits = model(batch)
        if teacher:
            teacher.eval()
            logits_teacher = teacher(batch)
            probs_teacher = torch.nn.functional.softmax(logits_teacher, dim=-1).detach()
            loss = loss_kd(logits, probs_teacher, method, **kwargs)
        else:
            loss = torch.nn.functional.cross_entropy(logits, labels)
        loss.backward()
        optimizer.step()
    eval(model, method)
    return model


def loss_kd(logits, probs_teacher, method, topk=7, to_sample=50):
    if "topk" in method:
        topk_probs, topk_ids = probs_teacher.topk(topk, dim=-1)
        probs_teacher *= 0
        probs_teacher.scatter_reduce_(dim=-1, index=topk_ids, src=topk_probs, reduce='sum')
    elif "random_sampling" in method:
        probs_teacher_cumsum = probs_teacher.cumsum(dim=-1)
        rand_probs = torch.rand(size=(probs_teacher_cumsum.shape[0], to_sample), device=probs_teacher_cumsum.device)
        rand_probs = rand_probs.sort(dim=-1)[0]
        sample_token_ids = torch.searchsorted(probs_teacher_cumsum, rand_probs) # Inverse Transform Sampling
        probs_teacher *= 0
        probs_teacher.scatter_reduce_(dim=-1, index=sample_token_ids, src=torch.ones_like(probs_teacher), reduce='sum')
        probs_teacher.div_(probs_teacher.sum(dim=-1, keepdim=True))

    logits_exp = torch.exp(logits)
    logits_sum_exp = torch.sum(logits_exp, dim=-1)
    logits_log_sum_exp = torch.log(logits_sum_exp)
    loss = - probs_teacher * (logits - torch.unsqueeze(logits_log_sum_exp, dim=-1))
    loss = torch.sum(loss, dim=-1).mean()
    return loss

class ToyModel(torch.nn.Module):
    def __init__(self, num_hidden):
        super().__init__()
        self.layer1 = torch.nn.Linear(num_dim, num_hidden)
        self.layer2 = torch.nn.Linear(num_hidden, num_hidden)
        self.layer3 = torch.nn.Linear(num_hidden, num_classes)
    def forward(self, x):
        x = torch.nn.functional.gelu(self.layer1(x))
        x = torch.nn.functional.gelu(self.layer2(x))
        x = self.layer3(x)
        return x

teacher = train(ToyModel(num_hidden_teacher).to(device), 'teacher')
student = train(ToyModel(num_hidden_student).to(device), 'student')
student_kd = train(ToyModel(num_hidden_student).to(device), 'student_full_kd', teacher=teacher)
student_topk = train(ToyModel(num_hidden_student).to(device), 'student_topk', teacher=teacher, topk=7)
student_random = train(ToyModel(num_hidden_student).to(device), 'student_random_sampling', teacher=teacher, to_sample=50)
```

L NLU Tasks Full Scores

Experiment	ARC Easy	HellaSwag	LAMBADA OpenAI	LAMBADA Standard	PIQA	Avg.
3B Teacher → 300M Student						
Base						
CE	46.59	41.18	38.85	30.80	67.41	44.97
Ours (12)	50.76	41.84	40.25	30.70	67.46	46.20
FullKD	51.56	41.98	40.69	29.52	67.25	46.20
8B Teacher → 3B Student						
Base						
CE	64.90	56.35	45.64	38.31	72.58	55.56
Top12	65.07	57.04	47.76	39.86	73.50	56.65
Top50	65.66	57.80	47.88	40.87	73.50	57.14
Ours (12)	66.29	58.93	47.47	40.99	73.83	57.50
Ours (12)++	68.14	60.82	46.83	39.80	73.99	57.92
FullKD	66.08	58.76	48.01	40.71	73.88	57.49
SFT, Tulu						
CE	58.84	57.51	45.66	37.92	72.69	54.52
Top12	63.51	58.49	50.92	42.97	72.47	57.67
Top50	66.58	59.26	50.86	42.07	72.91	58.34
Ours (12)	68.43	60.14	52.14	42.67	73.83	59.44
Ours (12)++	66.96	60.91	50.71	42.23	74.48	59.06
FullKD	68.22	59.59	50.46	42.32	73.01	58.72

Table 22: Full performance results on various benchmarks for 300M and 3B experiments.