EnsemW2S: Enhancing Weak-to-Strong Generalization with Large Language Model Ensembles

Anonymous ACL submission

Abstract

With Large Language Models (LLMs) rapidly approaching and potentially surpassing human-level performance, it has become imperative to develop approaches capable of effectively supervising and enhancing these powerful models using smaller, human-level models exposed to only human-level data. We address this critical weak-to-strong (W2S) generalization challenge by proposing a novel method aimed at improving weak experts, by training on the same limited human-level data, enabling them to generalize to complex, super-human-level tasks. Our approach, called **EnsemW2S**, employs a token-level ensemble strategy that iteratively combines multiple weak experts, systematically addressing the shortcomings identified in preceding iterations. By continuously refining these weak models, we significantly enhance their collective ability to supervise stronger student models. We extensively evaluate the generalization performance of both the ensemble of weak experts and the subsequent strong student model across in-distribution (ID) and out-of-distribution (OOD) datasets. For OOD, we specifically introduce question difficulty as an additional dimension for defining distributional shifts. Our empirical results demonstrate notable improvements, achieving 4%, and 3.2% improvements on ID datasets and, upto 6% and 2.28% on OOD datasets for experts and student models respectively, underscoring the effectiveness of our proposed method in advancing W2S generalization.

1 Introduction

Rapid advancements in Large Language Models (LLMs) have largely been driven by extensive internet-scale data and improvements in computational hardware. However, these vast datasets have largely reached their saturation point, offering diminishing returns in terms of new knowledge, and hardware-driven performance gains are also diminishing. Despite these limitations, the need for LLMs to effectively generalize to out-of-distribution, challenging, and unseen data remains crucial. This capability becomes even more essential as models approach and potentially exceed superhuman performance (Burns et al., 2023), necessitating super-human level supervision from human-level data and models. In this paper, we focus on enhancing the generalization performance of human-level experts using limited human-level labeled data and smaller-scale models. We subsequently leverage these improved models to supervise and enhance larger-scale models on superhuman-level data. Our framework is inspired from the "superalignment" or "w2s generalization" problem by Burns et al. (2023), but extends it by also focusing on generalization to superhuman-level data using only human-level labeled data.

W2S generalization has been previously studied (Sun et al., 2024; Burns et al., 2023; Ji et al., 2024; Charikar et al., 2024; Lang et al., 2024); however, several key challenges remain unresolved: **(C1) Generalization to Out-of-distribution (OOD) and super-human level data.** One key aspect of W2S generalization is the ability to extrapolate from human-level performance to super-human tasks using only easily accessible human-level data. While "easy" examples are abundant and cheap to label, "hard" or super-human examples often lie beyond the reach of annotators, either due to cognitive limits or prohibitive costs. In fact, for many tasks, groundtruth labels at the superhuman level may be *fundamentally unavailable*—raising a critical obstacle to benchmarking generalization at that level. Therefore, we believe that easy-to-hard (E2H) generalization is central to the W2S paradigm. To address this, we explicitly partition data into "easy" and "hard" categories and develop methods that promote generalization to harder questions even when trained solely on easy examples. **(C2) Limitation of Single Weak Supervisors.** Previous research predominantly utilizes a single weak

supervisor, often leading to biased, inconsistent, or incomplete supervisory signals. In practice, multiple complementary weak experts (e.g., humans specializing in different domains) can be readily accessed. Our work leverages an ensemble approach that aggregates these diverse weak supervisors, providing a richer, more balanced supervisory signal and significantly enhancing overall model generalization. **(C3) Lack of Iterative Weak Model Enhancement.** Existing approaches typically treat weak models as static entities, neglecting the potential for iterative improvements through limited labeled data. Although methods such as voting (Anonymous, 2025) and debate (Du et al., 2023) have attempted model combination during inference, they require a stronger supervisor (reward model or judge) to select optimal tokens or answers. In contrast, our approach iteratively refines weak models without relying on stronger supervisory signals or additional data, thereby fostering a self-reinforcing cycle of continuous improvement. This dynamic enhancement not only promotes continuous improvement of weak models for improved w2s generalization but also holds potential for answering previously unsolved and challenging questions.

To address the above challenges, we propose **EnsemW2S**, a novel token-level ensemble algorithm for autoregressive LLMs. Inspired by the spirit of AdaBoost—an ensemble method for binary classifiers known for improving generalization while resisting overfitting—EnsemW2S trains and combines complementary weak models at the token level. Specifically, it adjusts token probabilities in a manner analogous to controlled decoding methods (Mudgal et al., 2023a), but *extends them to a multi-model setting*. The approach iteratively improves newly introduced weak models based on identified shortcomings of prior models without requiring any additional labeled data.

We evaluate generalization rigorously on unseen data, distinguishing between in-distribution (ID) and OOD scenarios defined either by different datasets but the same task or question difficulty levels. We refer to this as **easy-to-hard (E2H) generalization**, a proxy for the unattainable goal of generalizing from human-level to superhuman performance—where groundtruth labels are inherently unavailable. By leveraging a small held-out labeled set for calibration, EnsemW2S enables effective supervision that improves both weak and strong models. As a result, the final strong model attains significantly better generalization and can even match or outperform models trained on full groundtruth datasets.

The **main contributions** of this paper are the following:

(1) EnsemW2S: A novel *token-level ensemble method* to improve generalization to difficult data and further improve w2s generalization. *(a)* EnsemW2S *iteratively improves weak models* without using additional data and aggregates them via a probabilistic voting mechanism. This self-reinforcing cycle of improvement significantly enhances generalization. *(b)* We also provide *theoretical justification* for exponential decay in training error bounds as more weak models are incorporated (assuming our new weak learning condition holds). We then discuss how training data margin distribution improves, leading to better generalization while preventing overfitting. *(c)* We achieve *improved generalization to unseen data by **upto 4% for ID and 4.34%, 6% for OOD*** (easy-hard) with test data from the same dataset and a different dataset, respectively.

(2) Better weak models further improve W2S Generalization by providing more effective supervision. We achieve *performance improvement of up to 3.2% on ID and 2.42%, 2.28% on OOD (easy-hard) with test data from the same dataset and a different dataset*, respectively.

2 Framework

W2S Generalization with E2H Generalization As a proxy for human and superhuman-level data, we split data into easy and hard, using the former to generalize to the latter. This E2H framework is a more pragmatic setting to study the (im)possibility of w2s generalization. in this framework, weak models (or human-level models) trained on simpler tasks guide a strong model (or superhuman-level model) toward solving more complex challenges, mirroring real-world conditions of limited human supervision. Figure 2 outlines EnsemW2S, where **weak expert models** (h_θ) proficient in 'easy tasks' $(\boldsymbol{x}^e, \boldsymbol{y}^e)$ supervise a **strong model** (f_ϕ) to solve (unsolvable) hard problems ($\boldsymbol{x}^h$). The strong student model under EnsemW2S is trained using unlabeled hard data alongside pseudo-labels, $(\boldsymbol{x}^h, h_\theta(\boldsymbol{x}^h))$, where $h_\theta(\boldsymbol{x}^h) = \text{func}(\{h_\theta^t\}_{t=1}^T, \boldsymbol{x}^h)$ denotes the ensemble-generated labels from weak experts. Ultimately, our goal is to improve the **Performance Gap Recovered (PGR)** which compares EnsemW2S against two baselines: a lower bound—

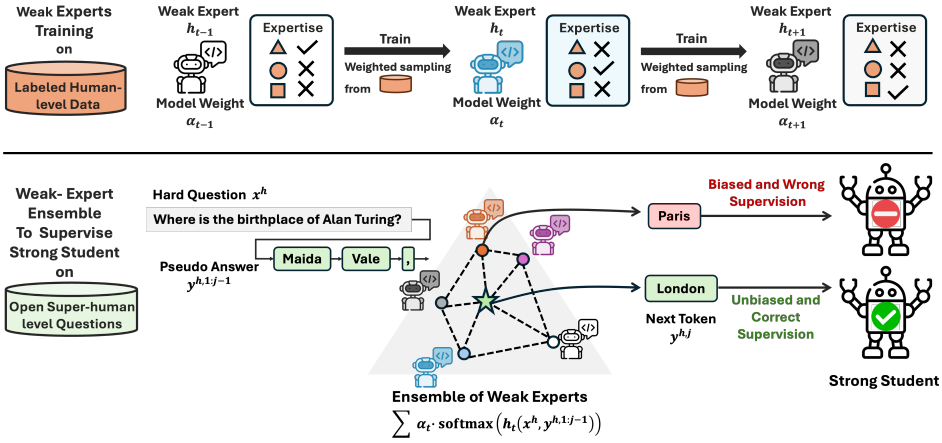

Figure 1: This figure illustrates the full pipeline of EnsemW2S. The top part shows how weak experts are iteratively improved by training on the same human-level labeled data. The bottom part depicts how these experts are combined at the token level during inference to supervise the strong student model. Training data is sampled using token-level weights, which define a distribution over training tokens. Based on the token-level errors of each trained weak model, we compute model weights (α_t), which are then used during inference to compute a weighted combination of token-level probabilities. This aggregated distribution is used to generate answer tokens for previously unseen, superhuman-level questions, providing supervision to the strong model—following the setup in Burns et al. (2023). EnsemW2S reflects a realistic scenario where weak experts perform well on easy questions and are leveraged to supervise a stronger model on harder (and potentially unsolved) tasks.

weak experts trained on easy data and evaluated on hard data—and an upper bound—the performance of a strong model trained directly on hard data. The upper bound or oracle, is the **strong model** (u_ϕ), which matches the student model in size and caliber but is trained using ground-truth labeled hard data $(\boldsymbol{x}^h, \boldsymbol{y}^h)$, typically unattainable in real-world scenarios. This oracle sets the upper performance limit for W2S models.

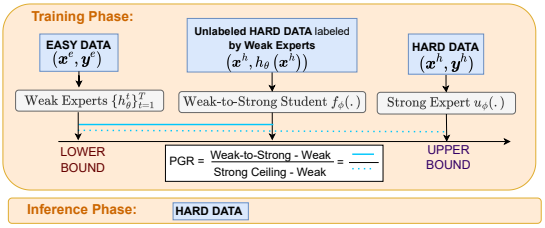

Figure 2: Overview. EnsemW2S leverages hard examples with pseudo-labels generated by weak experts to train a strong student model. To evaluate its effectiveness, we measure Performance Gap Recovered (PGR) (Burns et al., 2023)

3 EnsemW2S: Improving W2S via Improved Expert Ensembles

In this section, we introduce **EnsemW2S**, a token-level ensemble method for autoregressive generative models such as LLMs. Figure 1 outlines the method. As classification is a subset of generation, we first discuss ensemble methods for binary classification LLMs, followed by the challenges of applying such methods to generation tasks, and finally present our proposed solution.

3.1 AdaBoost for Binary Classification LLMs

AdaBoost is a well-known ensemble learning algorithm designed for binary classification. We apply AdaBoost to a binary classification LLM to study W2S generalization. Historically, AdaBoost demonstrates strong generalization on unseen data by increasing the margins of training examples, which correlates with lower generalization error, even after achieving zero training error-making it ideal for this study. AdaBoost iteratively reweights training examples, and trains weak experts one at a time focusing on hard-to-classify samples (check Line 5 of Algorithm 2). The essential requirement is that each weak learner must perform slightly better than random guessing, thereby satisfying the weak learning condition. The outputs of these weak experts are aggregated through a weighted majority vote to produce pseudo-labels: $\mathbb{1}\left(\sum_{t=1}^{T} \alpha_t h_\theta^t(\boldsymbol{x}^h) > 0\right) \in 0, 1$, where α_t are hyperparameters weighing each weak expert based on accuracy. For a detailed mathematical summary, refer to Appendix D. We then use this ensemble to generate answers for challenging ("hard") questions, $\boldsymbol{x}^h$.

3.2 Challenges of Ensemble for Generation Task

Popular ensemble methods like AdaBoost (Freund and Schapire, 1997) leverage the "wisdom of the crowd" to build stronger classifiers by combining weaker ones, typically through classification scores. Inspired by AdaBoost's ability to increase margin and improve generalization, we explore whether its principles can be extended to generation tasks. However, applying AdaBoost to autoregressive LLMs is infeasible due to several key

challenges: (1) **Single-Class vs Multi-Class:** Unlike binary classification, each token prediction in LLMs is a multi-class problem over a large vocabulary (typically >4000 classes). Aggregating such high-dimensional probability distributions introduces significant computational overhead and potential numerical instability. (2) **Autoregressive Generation:** Unlike classification, where scores are combined over a fixed output space, generation tasks involve sequential prediction, where early token errors can propagate and affect future outputs. This makes ensemble aggregation over generations inherently more complex. (3) **Variable-Length Outputs:** LLMs generate variable-length outputs, making it non-trivial to combine their responses consistently.

3.3 EnsemW2S: LLM Token-Level Ensemble Method

To address the above challenges, we propose a multi-class, generation based, AdaBoost inspired, LLM ensemble algorithm where the number of classes corresponds to the vocabulary size. To develop this algorithm, we treat each token as an independent sample and include error for each token. Algorithm 1 provides the steps for EnsemW2S algorithm for Multiple-Choice Q/A tasks. In the sections below we provide detail of each step along with theoretical proofs. Appx Fig. 5 gives details on algorithmic and data flow. Note, AdaBoost is a special case of EnsemW2S for binary classification task (see Algorithm 2 in Appx D). Refer Figure 1 for outline and intuition behind our method.

Token-Level Weighting. Our method generates weights for each token within a sentence sample. We define the initial token-sample weights vector $D_1(i,j) \leftarrow \frac{1}{n}$ for all $i \in [m], j \in [k_i]$, where $n = \sum_{i=1}^{m} k_i$, k_i is the number of tokens in the answer part of each sample i, m is the total number of training data samples and j is the jth token in a particular chosen i^{th} sample. We update these weights, $D_t(i,j)$, for each iteration t of EnsemW2S. Note here that **unlike AdaBoost**, EnsemW2S assigns weights to all tokens in a sample, adjusting for varying lengths, whereas AdaBoost maintains only one weight for each sample.

Token-Level Data Sampling. We sample $S' = \{(\boldsymbol{x}_i'^e, \boldsymbol{y}_i'^e)\}_{i=1}^{m}$ from S using token-sample weights $D_t(i,j)$. By sampling with respect to probability masses $D_t(i,j)$ with repetition, we obtain a set of $n = \sum_{i=1}^{m} k_i$ tokens to train on. However, treating these n sampled tokens as independent training samples is very inefficient. Instead, we "assemble" the sampled tokens back into the sentences they belong to and implement label masking to only train on the sampled tokens in each sentence. Following **this novel treatment proposed**, we can train on sampled tokens with minimal overheads.

Training and Generating New Weak Experts. For each iteration, t, of EnsemW2S algorithm we train a new weak expert model h_θ^t on the sampled data, S'. In this way, EnsemW2S *iteratively improves weak models* and recruits additional weak models throughout the training process—**the first approach** to dynamically refine weak models while training the strong model. This self-reinforcing cycle of improvement significantly enhances generalization.

Incorporating Prior Term. Vanilla Adaboost for binary classification uses $\alpha_t = \log(\frac{1-\epsilon_t}{\epsilon_t})$ to calculate votes for each classifier. To keep α_t positive they introduce a weak learning condition, $\epsilon_t < 0.5$, which means that weighted error for each classifier should be better than random. However, using the same weak-learning condition and voting mechanism is not feasible for multi-classification tasks. Thus, Hastie et al. (2009) uses an additional prior $\log(c-1)$ term, where c is the number of classes, in the calculation of the AdaBoost parameter α_t. This term serves two purposes: (1) It enables the generation of weak models with accuracy above $\frac{1}{c}$ %, where $\frac{1}{c}$ % is random selection accuracy. This is crucial for smaller models and challenging tasks that cannot achieve 50% accuracy. (2) It ensures that α remains positive. Now, for generative LLMs with large vocabularies, $\log(c-1)$ renders α_t nearly identical, making this approach infeasible. Therefore, we introduce a different method for calculating votes, $\alpha_t \leftarrow \log(\frac{1-\epsilon_t}{\epsilon_t}) + \log(\frac{1}{1-\epsilon_{pre}} - 1)$, where ϵ_{pre} is the pre-trained model error of the chosen LLM. We provide a theoretical proof (Theorem 1 in Section 3.3) regarding how the training error bound of the ensemble weak learners reduces exponentially with the addition of new weak learners while using new α_t method. This term is sensible because ϵ_{pre} represents the pretrained model error and makes our new weak learning condition to be $\epsilon_t < \epsilon_{pre}$, effectively replacing the binary and multi-class AdaBoost's conditions.

Weighted Error Calculation. Following the new weak learner's condition, the strict condition for each weak expert training is now that the weighted model error (calculated by com-

Algorithm 1 Main Algorithm: EnsemW2S

Require: "Easy" Q/A dataset $S^e = \{(x_i^e, y_i^e)\}_{i=1}^m$, pre-trained weak expert model h_θ^0, iterations T, "Hard" unlabeled (questions only) dataset $S^h = \{x_o^h\}_{o=1}^O$

Ensure: Weak-to-Strong Student Model $f_\phi(\cdot)$

Initialize Token-Sample Weights: $D_1(i,j) \leftarrow \frac{1}{n}, \quad \forall i \in [m], j \in [k_i]$, where k_i is token length of $\boldsymbol{y}_i^e = (\boldsymbol{y}_i^{e,1}, \boldsymbol{y}_i^{e,2} ... \boldsymbol{y}_i^{e,k_i})$ and $n = \sum_{i=1}^m k_i$

Compute Pre-Training Error of h_θ^0: $\epsilon_{\text{pre}} \leftarrow \sum_{i=1}^m \sum_{j=1}^{k_i} \mathbb{1}\{h_\theta^0(x_i^e, y_i^{e,j-1}) \neq y_i^{e,j}\} D_1(i,j)$

for $t \leftarrow 1$ to T **do**

  Sample $S' = \{(x_i^e, y_i^{e'})\}_{i=1}^m$ from S^e using $D_t(i,j)$

  Train new weak expert model h_θ^t on S'

  Compute error: $\epsilon_t \leftarrow \sum_{i=1}^m \sum_{j=1}^{k_i} \mathbb{1}\{h_\theta^t(x_i^e, y_i^{e,j-1}) \neq y_i^{e,j}\} D_t(i,j)$

  if $\epsilon_t \geq \epsilon_{\text{pre}}$ **then**

    Break

  end if

  Compute model weight: $\alpha_t \leftarrow \log \frac{1-\epsilon_t}{\epsilon_t} + \log\left(\frac{1}{1-\epsilon_{\text{pre}}} - 1\right)$

  Update sample weights: $D_{t+1}(i,j) \leftarrow \frac{D_t(i,j)}{Z_t} e^{\alpha_t \mathbb{1}\{h_\theta^t(x_i^e, y_i^{e,j-1}) \neq y_i^{e,j}\}} \forall i \in [m], j \in [k_i]$, where $Z_t = \sum_{i=1}^m \sum_{j=i}^{k_i} D_t(i,j) \cdot e^{\alpha_t \mathbb{1}(h_\theta^t(\boldsymbol{x}_i^e, \boldsymbol{y}_i^{e,j-1}) \neq \boldsymbol{y}_i^{e,j})}$

end for

for $o \leftarrow 1$ to O **do**

  for $j \leftarrow 1$ to k_o **do**

    Autoregressively generate the jth token of the "pseudo-answer" $\hat{y}_o^{h,j} \sim \Delta_{\text{vocab}}\big(\sum_{t=1}^T \alpha_t \cdot \text{softmax}(h_\theta^t(x_o^h, \hat{y}_o^{h,1:j-1}))\big)$, where Δ^{vocab} denotes the simplex on the vocabulary

  end for

end for

Train weak-to-strong model $f_\phi(\cdot)$ on $\{(x_o^h, \hat{y}_o^h)\}_{o=1}^O$

paring each token of each sample) must be less than the pre-training error, i.e., $\epsilon_t < \epsilon_{pre}$. The weighted model error ϵ_t is defined as, $\epsilon_t = \sum_{i=1}^m \sum_{j=1}^{k_i} \mathbb{1}\{h_\theta^t(\boldsymbol{x}_i^e, \boldsymbol{y}_i^{e,j-1}) \neq \boldsymbol{y}_i^{e,j}\} D_t(i,j) < \epsilon_{pre}$. Here, $\boldsymbol{y}_i^{e,j-1}$ is the $(j-1)^{\text{th}}$ ground-truth token in the answer part. The model $h_\theta^t(\boldsymbol{x}_i^e, \boldsymbol{y}_i^{e,j-1})$ predicts the next token and compares it with the ground-truth token $\boldsymbol{y}_i^j$ (during training when ground-truth is still available).

Weight Update Equation. Our sample-weight update equation for each token is $D_{t+1}(i,j) \leftarrow \frac{1}{Z_t} D_t(i,j) e^{\alpha_t \mathbb{1}\{h_\theta^t(\boldsymbol{x}_i^e, \boldsymbol{y}_i^{e,j-1}) \neq \boldsymbol{y}_i^{e,j}\}}$ where Z_t is a normalization factor ensuring that the updated weights satisfy $\sum_{i=1}^m \sum_{j=1}^{k_i} D_{t+1}(i,j) = 1$. The main idea is to adjust the sample weights to emphasize misclassified examples, thereby guiding the sampling process for next weak learner.

Combining experts to Generate Pseudo Answers for Hard Questions. To combine the outputs of different experts trained during the various EnsemW2S rounds, we scale the probability distribution for each token generated by the model h_θ^t in round t by its corresponding weight α_t. Specifically, we multiply α_t by the probability distribution vector of each token. We then aggregate these weighted distributions across all rounds, normalizing the resulting vector to form a new probability distribution for each token. Using this aggregated distribution, we sample the final predicted token. The process is autoregressive, where the jth token of the "pseudo-answer" is generated as

$$\widehat{\boldsymbol{y}}_o^{h,j} \sim \Delta^{\text{vocab}}\left(\sum_{t=1}^T \alpha_t \cdot \text{softmax}\left(h_\theta^t\left([\boldsymbol{x}_o^h, \widehat{\boldsymbol{y}}_o^{h,1:j-1}]\right)\right)\right) \tag{1}$$

where Δ^{vocab} represents the simplex over the vocabulary.

By combining the outputs of multiple experts, each trained in different EnsemW2S rounds, the ensemble approach leverages diverse perspectives from the weak models. Each expert contributes their learned strengths, and through weighted aggregation, we diminish the influence of models that are less confident or less effective on certain tokens. This helps reduce variance in the generation process, ensuring that errors from individual weak models are mitigated. The result is a more robust pseudo-labeling system that is better aligned with the true distribution of the hard data, often yielding a performance improvement over any single weak model.

Pseudo Answer Generation on Multiple-Choice Datasets. On multiple-choice Q/A datasets, instead of using generated tokens $\widehat{\boldsymbol{y}}^h$ as pseudo-answers, we can select one of the choices in the MCQ dataset using negative log-likelihood (NLL). Basically, we calculate the NLL between the choices and $\widehat{\boldsymbol{y}}^h$ and select the choice with the

lowest NLL. For datasets without multiple choices, we can directly use $\widehat{\boldsymbol{y}}^h$.

How does EnsemW2S improve generalization? First, Theorem 1 (3.3) shows that the training error decreases as more weak learners are added. Next, the "Margin theory for generalization" demonstrates that the margin distribution over the training data improves with each additional weak learner. Consequently, adding new models in the EnsemW2S ensemble reduces training error *without* overfitting, thereby improving generalization.

Theorem 1 (Exponential decay of training error with additional weak learners) *Under the condition $\epsilon_t < \epsilon_{prev}$, our token-level ensembling method, EnsemW2S, reduces the training error exponentially:*

$$\text{Error}_{\text{train}} = \frac{1}{n}\sum_{i=1}^{m}\sum_{j=1}^{k_i}\sum_{t=1}^{T}\alpha_t \mathbb{1}\left(h_\theta^t(\boldsymbol{x}_i^e, \boldsymbol{y}_i^{e,j-1}) \neq \boldsymbol{y}_i^{e,j}\right) < \prod_{t=1}^{T} Z_t, \tag{2}$$

where $Z_t < 1$, T is the number of EnsemW2S rounds, $n = \sum_{i=1}^{m} k_i$, and k_i is the token length of the ith easy example $S^e = \{(\boldsymbol{x}_i^e, \boldsymbol{y}_i^e)\}_{i=1}^{m}$. We assume tokens are i.i.d. for this proof. See Appendix F for full details.

Margin Theory for Generalization. Drawing on results from AdaBoost (binary classification) and SAMME (multiclass boosting), we observe that the margin over training data increases with the number of weak learners, making the ensemble's predictions more confident. Prior work shows increased margins improve generalization. Since we assume i.i.d. tokens, these conclusions can be directly apply. See Appendix G for details.

Train W2S Model. The strong student model, $f_\phi(\cdot)$, is trained using pseudo answers generated for the hard data $\{(\boldsymbol{x}_o^h, \widehat{\boldsymbol{y}}_o^h)\}_{o=1}^{O}$. Unlike Burns et al. (2023), we also include easy human-level data in the training, as it is a sensible choice. For completeness and proper comparison, we additionally run experiments that exclude the easy data.

Ablation Studies. Logit combination instead of probabilities showed no improvement (see Appendix Figure 9). We conducted ablation studies where, instead of treating each token as independent, we used a sliding window of length L while calculating weights and aggregating errors (see Appendix Figure 10 and 11). Different window lengths did not cause significant changes in values, so we ultimately chose a window of $L = 1$. We also explored treating each sample as independent instead of each token as independent in the sample-answer part, finding better results with the latter. This is reasonable since the error calculated using independent-sample weights is less accurate.

Evaluation Metric. We used two metrics to evaluate this Q/A dataset. One is **(1) Token-wise comparison**, where we compare each predicted token and average the total error, and **(2) Option-wise comparison**, where we compare the negative log-likelihood (NLL) of the correct answer completion with the NLLs of the incorrect answer completions. Accuracy represents the number of entries where the correct answer completion has the lowest NLL among all choices. We use option accuracy to reported results in the main paper.

Computational cost of EnsemW2S. The primary increase in computational cost compared to the baseline—where a single weak expert supervises the student—comes from multi-LLM decoding. However, its important to consider that unlike other multi-LLM works (Anonymous, 2025; Du et al., 2023), we do not use an additional reward or judge model for combining multi-LLM outputs, significantly reducing hardware demands. Moreover, we rely on the same labeled data rather than collecting new data to train separate models. A detailed computational analysis of each EnsemW2S component is provided in Appendix I.

4 Experimental Setup

ID and OOD Data Setup. To assess our method's improvement in generalization on both in-distribution (ID) and out-of-distribution (OOD) datasets, we use factual question-answering datasets such as ARC (Clark et al., 2018) and Quartz (Tafjord et al., 2019), as well as more complex mathematical datasets like math-mc (Tan et al., 2024) and MMLU (categories: high-school mathematics, elementary mathematics, college mathematics) (Hendrycks et al., 2021). All these datasets feature multiple-choice QA tasks, facilitating pseudo-label generation based on options.

For studying ID performance, we employ ARC, Quartz, and math-mc datasets, randomly splitting each dataset into two halves. One half is used for training weak experts. We treat the second half as unlabeled, generating pseudo-labels from the trained weak experts. Subsequently, we train the strong student model using both this pseudo-labeled data and the initially labeled half. **For assessing OOD performance**, we adopt two distinct strategies. First, we partition datasets into easy and hard subsets, training weak experts on labeled easy data, and the strong student on unlabeled hard data combined with labeled easy data

	In-Distribution Performance						
	Qwen2.5-3B		Qwen2.5-7B				
Dataset	Single Expert	EnsemW2S Expert	Student (Baseline)	Student (Ours)	Oracle	Student on Expert's data	Pre-trained Student
ARC	54.02	**57.04**	56.74	**58.70**	59.39	56.04	45.48
Quartz	81.25	**85.20**	84.95	**88.10**	88.78	83.47	60.00
Math-mc	61.74	**64.00**	62.90	**65.00**	65.87	62.90	24.73
	Out-of-Distribution Performance						
	Qwen2.5-3B		Qwen2.5-7B				
	Single Expert	EnsemW2S Expert	Student (Baseline)	Student (Ours)	Oracle	Student on Expert's data	Pre-trained Student
ARC	43.00	**45.76**	45.39	**46.67**	51.19	45.42	34.00
Quartz	80.99	**82.02**	83.29	**85.71**	87.5	82.48	56.12
Math-mc	33.00	**37.00**	55.86	**59.74**	60.91	49.00	26.72
MMLU-Elementary	71.43	**73.28**	73.91	**76.19**	75.66	73.57	44.71
MMLU-High-Skl	50.00	**53.00**	52.59	**53.70**	54.00	52.18	23.70
MMLU-College	33.00	**37.00**	46.00	**47.00**	47.50	44.00	33.00

Table 1: The first two columns show performance of a single weak expert and the EnsemW2S based ensemble. Its followed by strong students trained on the single and combined experts. The table also includes an oracle upper bound and two baselines: one trained on weak labels and one showing pretrained performance.

(similar to the ID setup). We evaluate performance on a small, labeled subset sharing the hard data distribution. Difficulty levels in math-mc are directly available through question-difficulty ratings, whereas for ARC and Quartz, we generate difficulty ratings via a cross-validation approach described in Appendix H. Difficulty rating plots for ARC and Quartz are provided in Figures 7 and 8, respectively. The second strategy involves training weak and strong models on one dataset (math-mc) and evaluating performance on a different dataset (MMLU's mathematics categories) to study how cross-data OOD generalization changes.

Expert and Student Models. We employ Qwen2.5-1.5B and Qwen2.5-3B models as weak experts, and Qwen2.5-7B as the strong student to study both ID and OOD generalization. Additionally, we explore scaling laws through multiple combinations of weak and strong model pairs using the Quartz dataset. For this scaling analysis, we utilize the Pythia model series, ranging from 70M to 2.8B parameters since pythia even though relatively old model series provided us with a wide range of models sizes. To **determine the optimal number of AdaBoost rounds**—i.e., how many models to combine—we use a validation set matching the training distribution. We limit our experiments to a maximum of 5 AdaBoost rounds/models.

Baseline. We compare EnsemW2S to a single weak expert and a strong student trained on outputs of single weak expert. Additional baselines include training the strong student on weak experts' labeled data and strong student's pre-training performance.

5 Results

5.1 Enhancing ID and OOD Generalization of Weak Experts using EnsemW2S

As depicted in Figure 4, we analyze the in-distribution (ID) and out-of-distribution (OOD) generalization performance of our EnsemW2S-based combination of weak experts for two models, Qwen2.5-1.5B and Qwen2.5-3B. We compare the performance of a single weak model (darker color) against an ensemble of weak models trained on the same labeled dataset, with lighter hues indicating performance improvements. For **ID generalization**, we observe absolute improvements between of **0.4-3.02%, 2.3-4%, and 2.24-2.26%** on the ARC, Quartz, and Math-MC datasets, respectively. For **OOD generalization** based on an **easy-to-hard data split**, the ensemble achieves improvements between **1.02-2.84%, 1.03-1.66%, and 3.7-4.34%** on the same datasets. We also evaluate **cross-dataset OOD performance** by training on the easy split of Math-MC and testing across three difficulty levels in MMLU's mathematics categories. In this setting, we observe gains between of **2.07-2.7%, 3-6%, and 3-4%** for elementary, high school, and college-level math questions, respectively. These results suggest that generalization performance is more sensitive to question difficulty than to dataset variation.

As expected, performance of ID data is higher than OOD data. However, the consistent gains achieved by our ensemble method—despite using the same labeled data as the single-model baseline—highlight its robustness and effectiveness. A few simple baselines are presented in Appendix J.1.

5.2 Stronger Expert further Improve Student and its W2S generalization performance.

As shown in Table 1, we study how an ensemble of weak models improves the generalization performance of the strong model. We focus on Qwen2.5-7B for this analysis and evaluate both ID and OOD generalization, following the same setup as before. For **ID generalization**, we observe **absolute performance improvement of 2%, 3.2%, and 2.1%** on the ARC, Quartz, and Math-MC datasets, re-

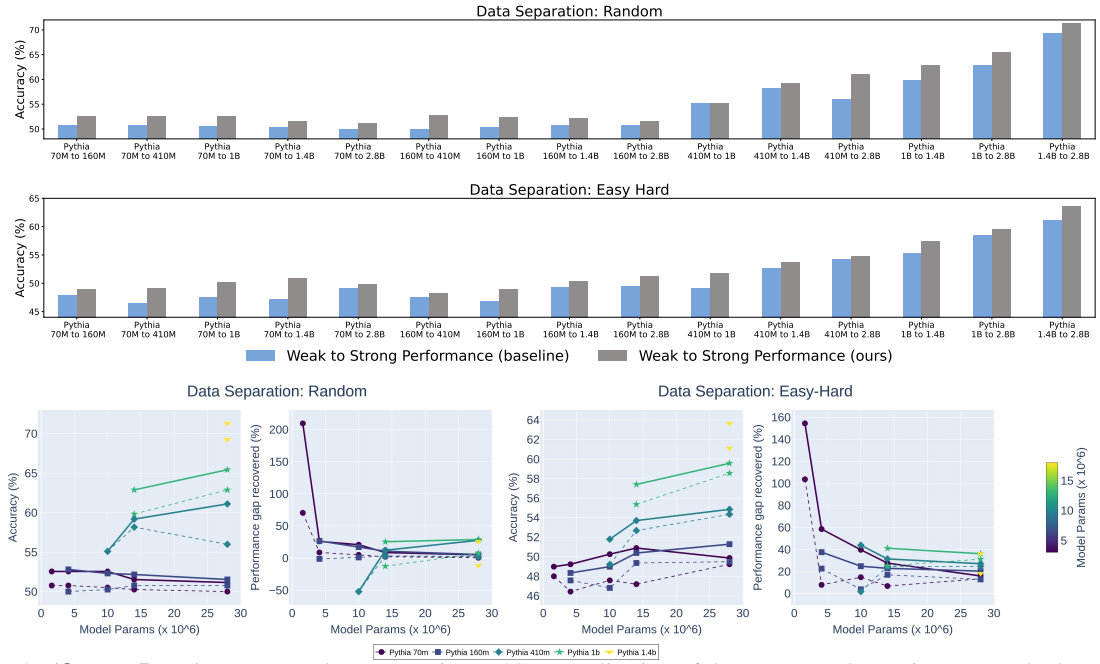

Figure 3: **(Quartz Data.)** Top: Bar plots comparing W2S generalization of the strong student using our method (gray) vs. baseline (blue) across weak–strong model pairs for SFT on Q/A data—ID (top bars) and OOD (bottom bars). Bottom: Line plots showing accuracy and PGR. Left: ID (random split); Right: OOD (easy–hard split) for E2H generalization.

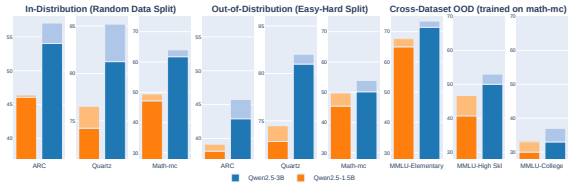

Figure 4: Performance comparison of weak models (single vs. ensemble). Blue bars: Qwen2.5-3B; orange: Qwen2.5-1.5B. Dark shades show single-model performance; light shades show ensemble gains. The bars are grouped into: (1) ID evaluation (first three subplots), where train and test distributions are similar; (2) OOD via easy-hard splits (middle three), where models are trained on the easy and tested on the hard; and (3) cross-dataset OOD generalization (last three), where models trained on the easy of Math-MC are evaluated on three difficulty levels of MMLU.

spectively. For **OOD generalization based on an easy-to-hard** split, we observe gains of **1.28%, 2.42%, and 1.17%** on the same datasets. We also evaluate **cross-dataset OOD performance** by training the ensemble on the easy subset of Math-MC and testing on varying difficulty levels within MMLU's mathematics categories. Here, we observe improvements of **2.28%, 1.11%, and 1%** for high-school, elementary, and college-level math questions, respectively.

A few notable conclusions: the EnsemW2S-based ensemble of weak experts achieves performance comparable to the W2S-trained strong student, demonstrating the effectiveness of ensembling. It even outperforms a strong model trained on weak experts' labeled data, despite the total combined model size being comparable in most cases. Moreover, the strong model trained with weak supervision from the ensemble surpasses the baseline which use a single expert model.

5.3 Improvement in PGR and W2S generalization for all range of model sizes.

Following Burns et al. (2023), we study scaling laws for W2S generalization in Fig.3 by creating multiple weak–strong model pairs from the same model family. We use the Pythia model series, which offers a wide range of model sizes. To enable a direct comparison with Burns et al. (2023), we train the strong model using only hard data. We also replicate the binary classification setup on SciQ using AdaBoost and the code from Burns et al. (2023) (Appendix J.3). Across experiments, we observe consistent improvements—indicating stronger W2S generalizations—except with models too small to learn—while the binary setup even shows complete generalization, underscoring the strength of ensemble-based weak supervision.

6 Conclusion

This paper presents a method to iteratively train weak experts by exposing them only to human-level labeled data, enabling them to better generalize to superhuman-level data. These weak supervisors are then combined at the token level to generate stronger supervision, improving W2S generalization for the strong student model. Our method shows significant improvement on both in-distribution and out-of-distribution datasets.

7 Limitation and Future Work

This work only explores the SFT phase. While SFT is an important part of the LLM learning pipeline, our future work will focus on developing weak supervision in the reward modeling phase. Another interesting future direction would be to improve the combination of tokens in the decoding phase by replacing the classical AdaBoost algorithm with more adaptive ensemble learning methods. We hope this work sparks discussion on combining multiple LLMs to improve w2s generalization.

References

Anonymous. 2025. Collab: Controlled decoding using mixture of agents for LLM alignment.

Hritik Bansal, Arian Hosseini, Rishabh Agarwal, Vinh Q Tran, and Mehran Kazemi. 2024. Smaller, weaker, yet better: Training llm reasoners via compute-optimal sampling. *arXiv preprint arXiv:2408.16737*.

Collin Burns, Pavel Izmailov, Jan Hendrik Kirchner, Bowen Baker, Leo Gao, Leopold Aschenbrenner, Yining Chen, Adrien Ecoffet, Manas Joglekar, Jan Leike, and 1 others. 2023. Weak-to-strong generalization: Eliciting strong capabilities with weak supervision. *arXiv preprint arXiv:2312.09390*.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D Lee, Deming Chen, and Tri Dao. 2024. Medusa: Simple llm inference acceleration framework with multiple decoding heads. *arXiv preprint arXiv:2401.10774*.

Jonathan D Chang, Kiante Brantley, Rajkumar Ramamurthy, Dipendra Misra, and Wen Sun. 2023. Learning to generate better than your llm. *arXiv preprint arXiv:2306.11816*.

Moses Charikar, Chirag Pabbaraju, and Kirankumar Shiragur. 2024. Quantifying the gain in weak-to-strong generalization. *arXiv preprint arXiv:2405.15116*.

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. *arXiv preprint arXiv:1803.05457*.

Mucong Ding, Chenghao Deng, Jocelyn Choo, Zichu Wu, Aakriti Agrawal, Avi Schwarzschild, Tianyi Zhou, Tom Goldstein, John Langford, Anima Anandkumar, and Furong Huang. 2024. Easy2hard-bench: Standardized difficulty labels for profiling llm performance and generalization. *Preprint*, arXiv:2409.18433.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B Tenenbaum, and Igor Mordatch. 2023. Improving factuality and reasoning in language models through multiagent debate. *arXiv preprint arXiv:2305.14325*.

Yoav Freund and Robert E Schapire. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139.

Trevor Hastie, Saharon Rosset, Ji Zhu, and Hui Zou. 2009. Multi-class adaboost. *Statistics and its Interface*, 2(3):349–360.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Jiaming Ji, Boyuan Chen, Hantao Lou, Donghai Hong, Borong Zhang, Xuehai Pan, Juntao Dai, and Yaodong Yang. 2024. Aligner: Achieving efficient alignment through weak-to-strong correction. *arXiv preprint arXiv:2402.02416*.

Lifeng Jin, Baolin Peng, Linfeng Song, Haitao Mi, Ye Tian, and Dong Yu. 2024. Collaborative decoding of critical tokens for boosting factuality of large language models. *arXiv preprint arXiv:2402.17982*.

Hunter Lang, David Sontag, and Aravindan Vijayaraghavan. 2024. Theoretical analysis of weak-to-strong generalization. *arXiv preprint arXiv:2405.16043*.

Yuejiang Liu and Alexandre Alahi. 2024. Co-supervised learning: Improving weak-to-strong generalization with hierarchical mixture of experts. *arXiv preprint arXiv:2402.15505*.

Sidharth Mudgal, Jong Lee, Harish Ganapathy, YaGuang Li, Tao Wang, Yanping Huang, Zhifeng Chen, Heng-Tze Cheng, Michael Collins, Trevor Strohman, and 1 others. 2023a. Controlled decoding from language models. *arXiv preprint arXiv:2310.17022*.

Sidharth Mudgal, Jong Lee, Harish Ganapathy, YaGuang Li, Tao Wang, Yanping Huang, Zhifeng Chen, Heng-Tze Cheng, Michael Collins, Trevor Strohman, and 1 others. 2023b. Controlled decoding from language models. *arXiv preprint arXiv:2310.17022*.

Corby Rosset, Ching-An Cheng, Arindam Mitra, Michael Santacroce, Ahmed Awadallah, and Tengyang Xie. 2024. Direct nash optimization: Teaching language models to self-improve with general preferences. *arXiv preprint arXiv:2404.03715*.

Jitao Sang, Yuhang Wang, Jing Zhang, Yanxu Zhu, Chao Kong, Junhong Ye, Shuyu Wei, and Jinlin Xiao. 2024. Improving weak-to-strong generalization with scalable oversight and ensemble learning. *arXiv preprint arXiv:2402.00667*.

Shannon Zejiang Shen, Hunter Lang, Bailin Wang, Yoon Kim, and David Sontag. 2024. Learning to decode collaboratively with multiple language models. *arXiv preprint arXiv:2403.03870*.

Zhiqing Sun, Longhui Yu, Yikang Shen, Weiyang Liu, Yiming Yang, Sean Welleck, and Chuang Gan. 2024. Easy-to-hard generalization: Scalable alignment beyond human supervision. *arXiv preprint arXiv:2403.09472*.

Oyvind Tafjord, Matt Gardner, Kevin Lin, and Peter Clark. 2019. Quartz: An open-domain dataset of qualitative relationship questions. *arXiv preprint arXiv:1909.03553*.

Wenting Tan, Dongxiao Chen, Jieting Xue, Zihao Wang, and Taijie Chen. 2024. Teaching-inspired integrated prompting framework: A novel approach for enhancing reasoning in large language models. *arXiv preprint arXiv:2410.08068*.

Pat Verga, Sebastian Hofstatter, Sophia Althammer, Yixuan Su, Aleksandra Piktus, Arkady Arkhangorodsky, Minjie Xu, Naomi White, and Patrick Lewis. 2024. Replacing judges with juries: Evaluating llm generations with a panel of diverse models. *arXiv preprint arXiv:2404.18796*.

Edwin Zhang, Vincent Zhu, Naomi Saphra, Anat Kleiman, Benjamin L Edelman, Milind Tambe, Sham M Kakade, and Eran Malach. 2024. Transcendence: Generative models can outperform the experts that train them. *arXiv preprint arXiv:2406.11741*.

A Limitation and Future Work

This work only explores the SFT phase. While SFT is an important part of the LLM learning pipeline, our future work will focus on developing weak supervision in the reward modeling phase. Another interesting future direction would be to improve the combination of tokens in the decoding phase by replacing the classical AdaBoost algorithm with more adaptive ensemble learning methods. We hope this work sparks discussion on combining multiple LLMs to improve w2s generalization.

Computational Overhead: For fully generative tasks, multiple forward passes are required in an autoregressive manner. At each step, the final voted token is input to all LLMs to predict the next token. This increases generation time, which can be mitigated using efficient decoding algorithms like speculative decoding. Addressing this also forms part of our future work. *Smaller Models:* Another limitation is of all w2s work is they attempt to mimic the weak and strong setting as an analogy to the realistic problem and cannot test on a real human with super-human model.

B Broader Impact

The proposed framework for weak-to-strong (w2s) generalization using ensembles of weak language models (LLMs) has significant implications across various domains. By demonstrating that multiple weak supervisors can effectively train more powerful models, our research addresses the critical challenge of superalignment, potentially transforming how advanced AI systems are developed and supervised. This approach could democratize access to powerful AI technologies by reducing reliance on scarce, high-quality labeled data and enabling more inclusive participation in AI development. Furthermore, our method encourages the creation of robust AI systems capable of tackling complex problems, which can drive advancements in fields such as healthcare, education, and scientific research. However, careful consideration must be given to ethical implications, ensuring that the deployment of these advanced models aligns with societal values and mitigates risks associated with misuse or unintended consequences.

C Related Work

Weak-to-Strong. (Burns et al., 2023) first introduced w2s generalization for super-alignment, aiming to elicit strong model capabilities using only single weak model supervision. (Charikar et al., 2024) provides a theoretical framework for the same with insights on how much w2s improvement can occur, though their work is limited to a few layers of neural networks. (Lang et al., 2024) establishes expansion bounds for w2s generalization under finite data distributions, but focuses solely on binary classification. (Zhang et al., 2024) shows that transcendence—surpassing the capability of the training data source—is possible via low-temperature sampling, offering insights relevant to w2s.

Several works have attempted to solve w2s generalization in LLMs. (Sang et al., 2024) explores ensemble learning for and scalable oversight for binary classification NLP tasks, but sees limited gains. (Ji et al., 2024) introduces a model that enhances the alignment of LLMs with human intentions by correcting the residual differences between aligned and unaligned answers by training on a query-answer correction dataset. It enhances w2s generalization by leveraging smaller models for supervisory signals. (Sun et al., 2024) proposes a scalable e2h generalization method, training reward models on simpler tasks and using them to evaluate harder ones. (Liu and Alahi, 2024) adapts the hierarchical mixture of experts, using multiple specialized weak supervisors instead of a single generalist for w2s. (Bansal et al., 2024) compares LLM training on weak (cheap) vs. strong (expensive) model-generated data and finds that larger weak-model datasets yield better w2s.

Ensemble Learning. Binary Classification Boosting (Freund and Schapire, 1997) and multi-classification boosting (Hastie et al., 2009) are common ensemble learning algorithms. In (Verga et al., 2024), they use a voting mechanism to combine multiple small LLMs instead of a single large LLM to evaluate another LLM and show it performs better than large LLMs.

Multi-LLM learning: There are numerous works involving the collaboration of multiple LLMs. (Chang et al., 2023) proposes Reinforcement Learning with Guided Feedback (RLGF), where a dynamic black-box guide like GPT-3 is used to fine-tune large language models. (Rosset et al., 2024) introduces Direct Nash

Optimization (DNO), a scalable algorithm that combines contrastive learning with general preference optimization. (Cai et al., 2024) presents MEDUSA, an innovative framework designed to accelerate inference in large language models by introducing multiple decoding heads, enabling simultaneous prediction of several tokens, and enhancing efficiency through reduced decoding steps and parallel processing capabilities. (Shen et al., 2024) proposes Co-LLM, a collaborative decoding framework that interleaves token-level generations from multiple models. This method optimizes the latent variable model for marginal likelihood, allowing a base model to decide when to generate tokens itself or utilize an assistant model, thereby improving performance across various specialized tasks without direct supervision. (Jin et al., 2024) introduces a novel collaborative decoding framework aimed at improving the factuality of large language models by employing a critical token classifier. This approach strategically uses both pre-trained and aligned models to selectively generate critical tokens, significantly enhancing the model's ability to maintain factual accuracy without compromising the diversity of the generated content.

Additionally, (Mudgal et al., 2023b) introduces Controlled Decoding (CD), a method for aligning language model outputs with desired outcomes using a separate prefix scorer module. This approach allows multi-objective RL without additional training and performs well on benchmarks, bridging the gap between token-level control and sequence-level best-of sampling strategies.

D Adaboost for combing weak Binary classification LLM.

AdaBoost is an ensemble learning algorithm that combines multiple weak classifiers, such as decision stumps, to create a strong classifier. It works iteratively by focusing on the samples that are hardest to classify, assigning them higher weights in each subsequent iteration. Weak classifiers are trained one at a time, and their contributions are weighted based on their accuracy. The final prediction is made by taking a weighted majority vote of all weak classifiers. AdaBoost is known for its ability to improve generalization by focusing on difficult cases and is often resistant to overfitting with simple weak learners. However, it can struggle with noisy data if overemphasis is placed on misclassified samples. Its also presented as Algorithm 2.

We use Adaboost to conduct a thought experiment to test the ensemble idea in w2s generalization for simple classification task. This is also the first task evaluated by Burns et al. (2023). We use vanilla AdaBoost (Algorithm 2) to generate answers to hard questions, $\boldsymbol{x}^h$, from each weak LLM teacher, $h_\theta^t(\boldsymbol{x}^h)$ for $t \in \{1, \ldots, T\}$, where T is max Adaboost round. AdaBoost iteratively reweights training examples, and trains weak teachers one at a time focusing on hard-to-classify samples, as described in Line 5 of Algorithm 2. The only requirement is they perform better than random, thus satisfying the weak learning condition. A weighted majority vote aggregates their outputs to produce a pseudo-label, $\mathbb{1}(\sum_{t=1}^{T} \alpha_t h_\theta^t(\boldsymbol{x}^h) > 0) \in \{0, 1\}$, where α_t are hyperparameters weighing each weak teacher based on accuracy. A detailed mathematical summary is provided below.

Let the training dataset consist of m samples:

$$\{(x_i, y_i) \mid i = 1, 2, \ldots, m\}, \quad x_i \in \mathbb{R}^d, \quad y_i \in \{-1, +1\}.$$

Each weak learner $h_t(x)$ outputs a prediction $h_t(x_i) \in \{-1, +1\}$. The goal is to sequentially train weak learners such that the combined model minimizes the classification error. A weight distribution $D_t(i)$ is maintained over the training samples at each iteration t, where:

$$D_t(i) \geq 0, \quad \sum_{i=1}^{m} D_t(i) = 1.$$

Initially, all samples are equally weighted: $D_1(i) = \frac{1}{m}, \quad \forall i$

Training the Weak Learners: For each iteration $t = 1, 2, \ldots, T$, train a weak learner $h_t(x)$ using the current weight distribution D_t. Compute the weighted error:

$$\epsilon_t = \sum_{i=1}^{m} D_t(i) \cdot \mathbb{I}(h_t(x_i) \neq y_i),$$

where $\mathbb{I}(\cdot)$ is the indicator function.

Weak Learner Weight Assign a weight α_t to the weak learner based on its performance:

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right)$$

Intuition behind it is that if ϵ_t is small, α_t is large, giving more importance to the weak learner. If $\epsilon_t = 0.5$, $\alpha_t = 0$, indicating no contribution to the ensemble. $\epsilon_t > 0.5$ is undesirable, as the weak learner performs worse than random guessing.

Update the weights of the training samples to focus on misclassified samples:

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t},$$

where Z_t is a normalization factor ensuring $\sum_{i=1}^{m} D_{t+1}(i) = 1$:

$$Z_t = \sum_{i=1}^{N} D_t(i) \exp(-\alpha_t y_i h_t(x_i)).$$

Misclassified samples ($y_i \neq h_t(x_i)$) receive higher weights, making them more influential in the next iteration. The **final strong classifier** $H(x)$ is a weighted majority vote of the weak learners:

$$H(x) = \text{sign} \left(\sum_{t=1}^{T} \alpha_t h_t(x) \right).$$

Generalization Abilities: AdaBoost improves generalization by maximizing the margins on the training set. The margin for a sample (x_i, y_i) is defined as:

$$\text{Margin}(x_i) = y_i \sum_{t=1}^{T} \alpha_t h_t(x_i).$$

AdaBoost aims to increase the margin for all samples, reducing the chance of misclassification.

Summary of Key Properties

1. **Sequential Training:** Weak learners are trained iteratively, with weights updated to focus on difficult samples.
2. **Weighting Scheme:** Misclassified samples are emphasized in subsequent iterations.
3. **Generalization:** AdaBoost achieves strong generalization by maximizing margins and minimizing exponential loss.
4. **Flexibility:** It can work with any weak learner as long as the learner achieves performance slightly better than random guessing.

E Details on the EnemW2S Methodology

E.1 Detailed Flowchart

E.2 Important Notations

Easy Data: $\{(\boldsymbol{x}_i^e, \boldsymbol{y}_i^e)\}_{i=1}^{m}$
Hard Data: $\{(\boldsymbol{x}_o^h, \boldsymbol{y}_o^h)\}_{o=1}^{O}$
Total number of Easy Data points: m
Total number of Hard Data points: O
Total EnsemW2S-AdaBoost Rounds: T Weak Teachers: $\{h_\theta^t\}_{t=1}^{T}$
Strong Student (Oracle): u_ϕ

Algorithm 2 AdaBoost (Freund and Schapire, 1997)

Require: Training Dataset $S = \{(x_i, y_i)\}_{i=1}^{m} \sim D^m$
$T =$ AdaBoost iterations
$\vec{D}_1(i) \leftarrow \frac{1}{m} \forall i \in [m]$
for $t \leftarrow 1$ to T **do**
h_t such that $\epsilon_t = \sum_{i=0}^{m} \nVdash\{h_t(x_i) \neq y_i\}\vec{D}_t(i) < \frac{1}{2}$
$\alpha_t \leftarrow \frac{1}{2} \log \frac{1-\epsilon_t}{\epsilon_t}$
$Z_t \leftarrow 2\sqrt{\epsilon_t(1-\epsilon_t)}$
$\vec{D}_{t+1} \leftarrow \frac{1}{Z_t}\vec{D}_t e^{-\alpha_t y_i h_t(x_i)}$
$g \leftarrow \sum_{t=1}^{T} \alpha_t h_t$
end for
Return $h(x) = \text{sign}(g)$

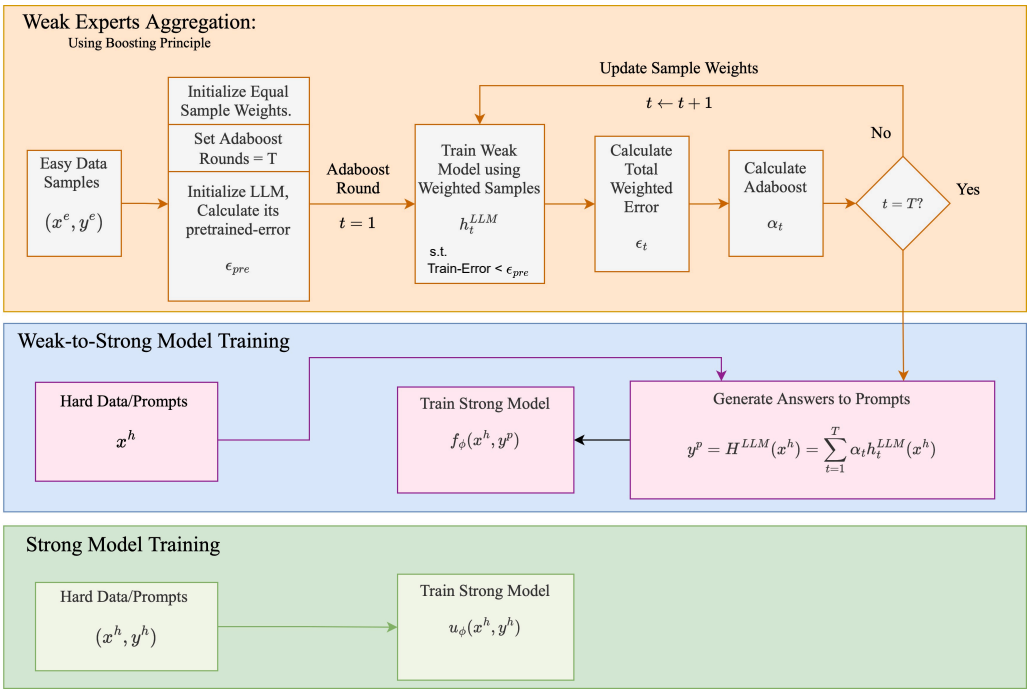

Figure 5: This figure explains our pipeline for easy-to-hard generalization using w2s generalization in complete detail including the algorithm and data flow. We train weak models on easy data and strong models on hard data. A transfer model is trained using pseudo labels generated by the weak model on the hard data. Ultimately, we aim to improve the Performance Gap Recovered (PGR).

Weak-to-Strong model: f_ϕ
Total number of tokens in the answer part of each sample i: k_i
AdaBoost voting parameter: $\{\alpha_t\}_{t=1}^T$
EnsemW2S-AdaBoost token-sample weights for i^{th} sample and j^{th} token: $\{D_t(i,j)\}_{t=1}^T$
Pre-trained Model error: ϵ_{pre}
EnsemW2S-AdaBoost's weighted model error for round t: ϵ_t

E.3 Intuition Behind Prior Term in EmsemW2S

The calculation of α cannot rely solely on error, ϵ, as the traditional Adaboost method is valid only when $\epsilon < 0.5$. Applying the same equation in our context could yield negative α values. We introduce a prior term, $\log(\frac{1}{1-\epsilon_{pre}} - 1)$. Below we provide justification and proof regarding exponential decay of the training error using the new prior term.

Existing works on multi-class classification Adaboost (Hastie et al., 2009) suggest using $\frac{1}{c}$ (where c is the number of classes) in the prior term, $\log(c-1)$, as $\frac{1}{c}$ represents the random performance of the model. However, when c (the number of classes) becomes very large, the $\log(c-1)$ term also grows significantly, causing the α parameters of Adaboost to become nearly identical and, consequently, less useful. To address this, we introduce a pre-training error term, ϵ_{pre}, which represents an upper bound on the sample error. We then use $1-\epsilon_{pre}$ (a lower bound on accuracy) as a replacement for the $\frac{1}{c}$ term, as our model's lowest possible accuracy is $1-\epsilon_{pre}$, not $\frac{1}{c}$.

F [Theorem 1:] Training Error Bound Exponential Decay.

Training Error of the combined weak learner is defined as the following:

$$\text{Err}(H) = \frac{1}{n}\sum_{i=1}^{m}\sum_{j=1}^{k_i}\sum_{t=1}^{T}\alpha_t \nVdash(h_\theta^t(\boldsymbol{x}_i^e, \boldsymbol{y}_i^{e,j-1}) \neq \boldsymbol{y}_i^{e,j}) \tag{1}$$

where T is EmsemW2S rounds, $n = \sum_{i=1}^m k_i$, k_i is the token length in the ith easy example $S^e = \{(\boldsymbol{x}_i^e, \boldsymbol{y}_i^e)\}_{i=1}^m$.

AdaBoost minimizes an upper bound on the exponential loss rather than directly minimizing the classification error. The exponential loss for our case is:

$$L = \sum_{i=1}^{m}\sum_{j=i}^{k_i} D_1(i,j)\cdot e^{\sum_{t=1}^T \alpha_t \nVdash(h_\theta^t(\boldsymbol{x}_i^e, \boldsymbol{y}_i^{e,j-1}) \neq \boldsymbol{y}_i^{e,j})} \tag{2}$$

To analyze how the training error decreases, consider the weight update rule:

$$D_{t+1}(i,j) \leftarrow \frac{1}{Z_t} D_t(i,j) e^{\alpha_t \nVdash(h_\theta^t(\boldsymbol{x}_i^e, \boldsymbol{y}_i^{e,j-1}) \neq \boldsymbol{y}_i^{e,j})} \text{ for all } i \in [m], j \in [k_i] \tag{3}$$

where Z_t is the normalization factor:

$$Z_t = \sum_{i=1}^{m}\sum_{j=i}^{k_i} D_t(i,j)\cdot e^{\alpha_t \nVdash(h_\theta^t(\boldsymbol{x}_i^e, \boldsymbol{y}_i^{e,j-1}) \neq \boldsymbol{y}_i^{e,j})} \tag{4}$$

Now we want the **cumulative training error** after T rounds to be bounded by (More details regarding the following equation is in section F.1):

$$\text{Err}(H) \leq \prod_{t=1}^{T} Z_t^{\frac{1}{\epsilon_{pre}}} \tag{5}$$

Now for $\text{Error}_{\text{train}}$ to exponentially decay with t, $Z_t < 1$. For $Z_t < 1$ we must have $\alpha_t > 0$.

Thus, we want

$$\log\frac{1-\epsilon_t}{\epsilon_t} + \log(\frac{1}{1-\epsilon_{pre}} - 1) > 0$$

$$(\frac{1-\epsilon_t}{\epsilon_t}) * (\frac{1}{1-\epsilon_{pre}} - 1) > 1$$

Solving this we get

$$\epsilon_t < \epsilon_{pre}$$

This forms the our weak learning condition. This basically means that each new weak learner (LLM) that is included in the ensemble should have error less than pretrained error. This error could also be the worst possible error below which any of the LLMs would not go.

F.1 Calculating Training Error Bound for EnsemW2S

In this section we will derive equation 5 of the previous section.

We know our models weights from previous section: $\alpha_t = \ln\big(\frac{1-\epsilon_t}{\epsilon_t}\big) + \ln(\frac{\epsilon_{pre}}{1-\epsilon_{pre}})$. We can break the equation 4 in the following form,

$$Z_t = (1-\epsilon_t) + \epsilon_t\, e^{\alpha_t} = (1-\epsilon_t)(\frac{1}{1-\epsilon_{pre}}) = \frac{\gamma_t - \epsilon_{pre}}{1-\epsilon_{pre}}.$$

Potential. Define the exponential potential and multi-class margin m_i

$$W_{T+1} \;=\; \frac{1}{n}\sum_{i=1}^{n} \exp\big(-\lambda m_i^{(T)}\big) \tag{6}$$

$$m_i^{(T)} = \frac{1}{2}\Big(F^{(T)}_{\boldsymbol{y}_i^{e,j}}(\boldsymbol{x}_i^e, \boldsymbol{y}_i^{e,j-1}) - \max_{k \neq \boldsymbol{y}_i^{e,j}} F_k^{(T)}(\boldsymbol{x}_i^e, \boldsymbol{y}_i^{e,j-1})\Big) \tag{7}$$

where $F_k^{(T)}$ is the accumulated vote after T rounds and $\lambda = \frac{1}{\epsilon_{pre}}$.

Mathematically, we can therefore write $F_k^{(T)}$ as

$$F_k^{(T)}(\boldsymbol{x}_i^e, \boldsymbol{y}_i^{e,j-1}) = \sum_{t=1}^{T} \alpha_t \mathbf{1}\left[h_\theta^t(\boldsymbol{x}_i^e, \boldsymbol{y}_i^{e,j-1}) = \boldsymbol{y}_i^{e,j}\right]$$

In a similar way we can define the following final hypothesis,

$$H(\boldsymbol{x}_i^e, \boldsymbol{y}_i^{e,j-1}) = \arg\max_{k \in [K]} F_k(\boldsymbol{x}_i^e, \boldsymbol{y}_i^{e,j-1})$$

A short algebraic check (see Zhu et al., 2006) gives the *telescoping relation:* $W_{t+1} = W_t Z_t^{(K-1)/K}$. Therefore,

$$W_{T+1} \;=\; \prod_{t=1}^{T} Z_t^{\lambda}. \tag{8}$$

Bounding the indicator. If $H(\boldsymbol{x}_i^e, \boldsymbol{y}_i^{e,j-1}) \neq \boldsymbol{y}_i^{e,j}$, then $m_i \leq 0$. Hence $\mathbb{1}[H(\boldsymbol{x}_i^e, \boldsymbol{y}_i^{e,j-1}) \neq \boldsymbol{y}_i^{e,j}] \leq e^{-m_i^{(T)}}$ and $e^{-m_i^{(T)}} \leq e^{-\lambda m_i^{(T)}}$ $(\lambda < 1)$,

$$\mathrm{Err}(H) \;=\; \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}[H(\boldsymbol{x}_i^e, \boldsymbol{y}_i^{e,j-1}) \neq \boldsymbol{y}_i^{e,j}] \;\leq\; \frac{1}{n}\sum_{i=1}^{n} e^{-\lambda m_i}. \tag{9}$$

The m_i appearing in Eq. (2) is the margin after all T rounds. Therefore the right-hand side of equation 8 is exactly the uniform-average potential in (7):

$$\boxed{\mathrm{Err}(H) \;\le\; W_{T+1} = \prod_{t=1}^{T} Z_t^{\frac{1}{\epsilon_{pre}}}}$$

Final bound. Using $1 + z \le e^z$ ($z \in [0,1]$) we have $Z_t \le e^{K\gamma_t}$. Here $K = \frac{1}{1-\epsilon_{pre}}$ and $\gamma_t = 1 - \epsilon_t - \frac{1}{K}$

$$\boxed{\mathrm{Err}(H) \;\le\; \exp\Big(-(K-1)\sum_{t=1}^{T}\gamma_t\Big)}$$

Thus, if every $\gamma_t > 0$ the empirical error decays *exponentially* with T.

□

G Margin theory for generalization

For any $\theta > 0$ define the small–margin set $\mathcal{M}(\theta) = \{i : m_i^{(T)} \le \theta\}$. From the derivation above

$$\Pr_{i\in S}\big[m_i^{(T)} \le \theta\big] \;\le\; e^{\lambda\theta}\, W_{T+1} \;\le\; \exp\Big(\lambda\theta - (K-1)\sum_{t=1}^{T}\gamma_t\Big). \tag{B.1}$$

VC-style margin bound (multi-class). Let $d = \mathrm{VCdim}(\mathcal{H})$ for the weak-learner family. A direct extension of the binary margin bound of Schapire, Freund, Bartlett & Lee (1998) to K classes (see Schapire & Singer, 1999; Zhu et al., 2006) states:

Theorem G.1 (Generalisation via Margins) *For any $\theta \in (0,1]$ and $\delta \in (0,1)$, with probability $\ge 1-\delta$ over the draw of an n-sample S, the boosted classifier H satisfies*

$$\Pr_{(x,y)\sim\mathcal{D}}\big[H(\boldsymbol{x}_i^e, \boldsymbol{y}_i^{e,j-1}) \ne \boldsymbol{y}_i^{e,j}\big] \;\le\; \Pr_{i\in S}\big[m_i^{(T)} \le \theta\big] \;+\; \sqrt{\frac{c\big(d\log n + \log(1/\delta)\big)}{n\,\lambda^2\theta^2}},$$

where c is an absolute constant.

Combining (B.1) with Thm. G.1. Using the empirical bound (B.1) inside the theorem yields

$$\boxed{\Pr_{\mathcal{D}}\big[H(\boldsymbol{x}_i^e, \boldsymbol{y}_i^{e,j-1}) \ne \boldsymbol{y}_i^{e,j}\big] \;\le\; \exp\Big(\lambda\theta - (K-1)\sum_{t=1}^{T}\gamma_t\Big) \;+\; \sqrt{\frac{c\big(d\log n + \log(1/\delta)\big)}{n\,\lambda^2\theta^2}}\,.}$$

Choose a fixed margin target $\theta > 0$. Because the first term decays exponentially in T (via $\sum_t \gamma_t$) and the second shrinks as $O(n^{-1/2})$, the overall test error can be made arbitrarily small by taking enough boosting rounds and/or sufficiently many training examples.

G.1 Discussion

- **Empirical error.** Theorem F.1 shows that EnsemW2S drives the training error to zero at an exponential rate provided every weak learner beats random guessing ($\epsilon_t < \epsilon_{pre}$).
- **Generalization.** Theorem G connects this fast margin growth to a quantitative VC-style bound on test error, mirroring the classical binary AdaBoost margin theory.
- **Practical implication.** In practice, once the empirical small-margin fraction $\Pr_{i\in S}[m_i \le \theta]$ plateaus, additional boosting offers diminishing returns unless more data (larger n) or a richer weak-learner class (larger d) is available.

H Experimental Setup

H.1 Cross Validation method for difficulty rating generation

Easy $(\boldsymbol{x}^e, \boldsymbol{y}^e)$ and Hard $(\boldsymbol{x}^h, \boldsymbol{y}^h)$ Data Split. We generate difficulty ratings using cross-validation in n folds, where the model trains on $(n-1)$ splits and tests on the remaining split, repeating for all folds. The errors aggregated across the samples serve as difficulty ratings. Low-rated samples are used for weak model training, while high-rated samples are used to generate strong model training data and testing data at random. This cross-validation approach, applied to both classification and generation tasks, ensures robust difficulty splits. Additional details and plots of difficulty ratings are provided in Figures 6, 7, and 8 in the Appendix. Additionally, the continuous difficulty labels provided by Ding et al. (2024) for math, programming, chess, and reasoning datasets enable flexible easy-hard data splits at any chosen difficulty threshold. As this aspect is independent of our core message, we leave its exploration to future work.

H.2 Difficulty rating for all the datasets

We use GPT-2 for binary classification and pythia-160m for SFT task's easy and hard splitting. We use the same training parameters as used in the training of the actual w2s results.

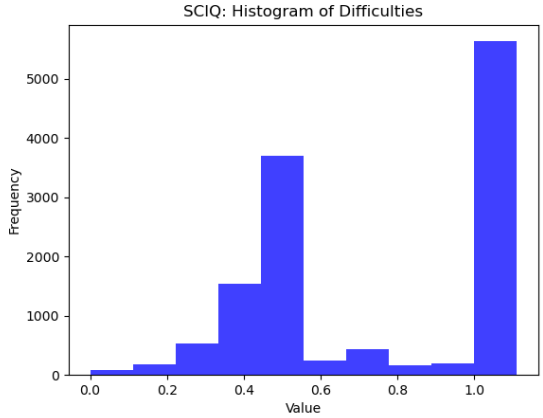

Figure 6: This figure shows the difficulty rating distribution of sciq dataset.

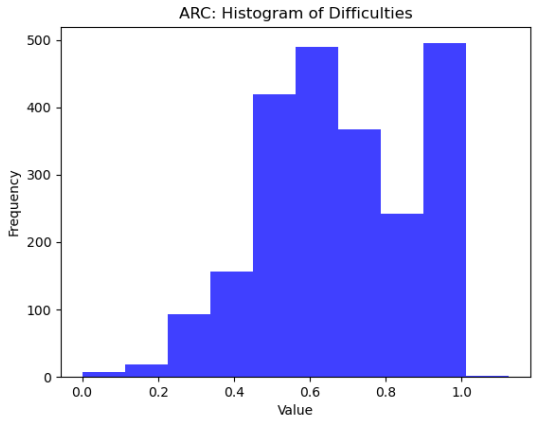

Figure 7: This figure shows difficulty rating distribution of ARC dataset.

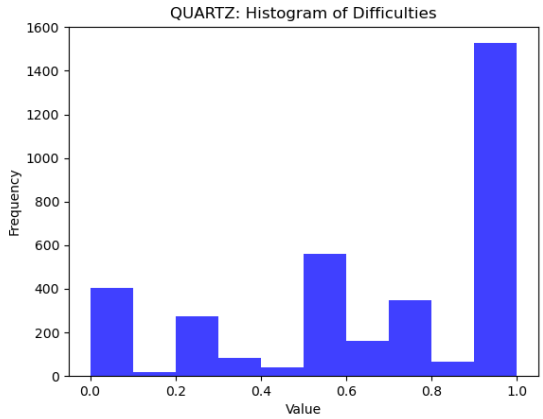

Figure 8: This figure shows difficulty rating distribution of quartz dataset.

H.3 [Ablation Study:] Comparison between probability based combination with logit based combination of the tokens, during generation and evaluation of combined weak experts.

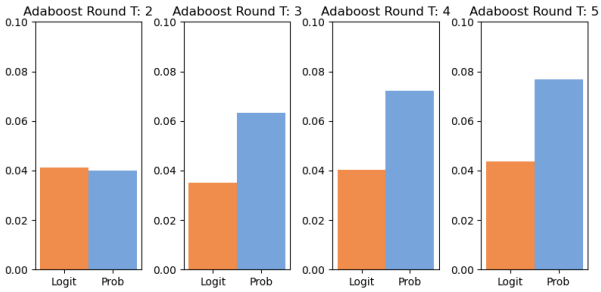

Figure 9: This figure compares probability-based combination with logit-based combination of the tokens across different AdaBoost rounds. Here we show improvement from the baseline where baseline is single model. The orange bars represent logit-based combination, while the blue bars represent probability-based combination, showing that probability-based combination performs better.

H.4 [Ablation Study:] Comparison between different window lengths for "sample and token weighing".

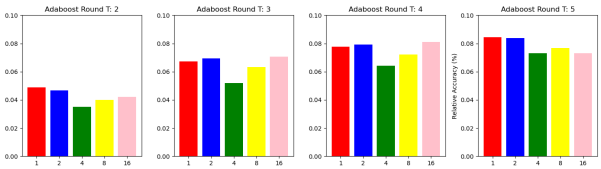

Figure 10: This figure compares different token window lengths for the Pythia 70M model across various AdaBoost rounds. The plots show improvements over the baseline, where the baseline represents a single model. The different bars (red, blue, green, yellow, and pink) correspond to window lengths of 1, 2, 4, 8, and 16, respectively. We observe that, overall, all window lengths perform similarly. Window length in EmsemW2S plays a role only during sampling step.

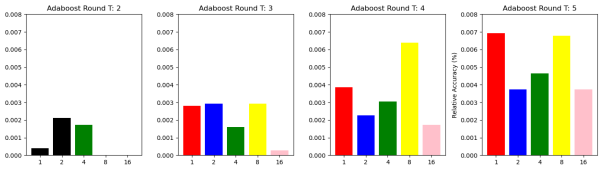

Figure 11: This figure compares different token window lengths for the Pythia 410M model across various AdaBoost rounds. The plots show improvements/decline over the baseline, where the baseline represents a single model. Thus, the black colored bars show decline. The different bars (red, blue, green, yellow, and pink) correspond to window lengths of 1, 2, 4, 8, and 16, respectively. We observe that, overall, all window lengths perform similarly. Window length in EmsemW2S plays a role only during sampling step.

I Cost Analysis of EmsemW2S

Training Cost of Weak Learners: Each weak learner is trained sequentially, as its performance is contingent upon the outputs of the preceding weak learner. Consequently, while the GPU load may be lower because each weak learner is small as compared to the collection of weak-learners, the overall training time is directly proportional to the number of weak learners utilized.

This is because the input and output token count for each weak learner during training remains approximately constant, as suggested by Adaboost. Only the frequency of samples are adjusted based on weights. In EnsemW2S we sample the tokens by token-weights but eventually combine the sampled tokens while masking the ones not sampled, thus keeping the total tokens approximately similar and training time for each weak-learner independent of the tokens sampled. In the practical superalignment case, pre-trained weak learners can be used, which may further mitigate concerns regarding sequentially training the weak learners.

Strong Model Training and Inference: The strong model is trained using labels generated by the weak learners and is evaluated on standard datasets. Therefore, the training cost and inference cost associated with the strong model remains unchanged.

Inference Cost of Weak Learners: The generation process can be executed in parallel as well as sequentially, resulting in a GPU load for generation or clock time for generation respectively, that scales linearly with the number of weak learners. For decoding, once the token-level distributions generated by the weak learners are combined using EmsemW2S algorithm, efficient decoding algorithms like speculative decoding can be employed to produce the final response. However, this is not the focus of this work.

We do agree that the primary increase in computational cost compared to the baseline—where a single weak expert supervises the student—comes from multi-LLM decoding. However, its important to consider that unlike other multi-LLM works (Anonymous, 2025; Du et al., 2023), we do not use an additional reward or judge model for combining multi-LLM outputs, significantly reducing hardware demands. Moreover, we rely on the same labeled data rather than collecting new data to train separate models. For this same reason we use datasets which do not have very long outputs and are option based.

J Results

J.1 Baselines for weak-expert performance comparison

In this section we show one additional baseline to compare weak model's performance. Basically instead of using adaptive weights of the model we use fixed model weights i.e. $\alpha_t = 1 \forall t = [1, T]$. This is similar to bagging. You can find the results in the table 2.

	Qwen-1.5B				Qwen-3B			
	Random (ID)		Easy-Hard (OOD)		Random (ID)		Easy-Hard (OOD)	
	Single Model	5 Models	Single Model	5 Models	Single Model	5 Models	Single Model	5 Models
ARC	46.02	47.00	38.14	38.22	54.02	52.75	42.92	43.00
Quartz	74.23	74.36	72.83	71.55	81.25	76.02	80.99	76.02
Math	47.17	47.20	45.42	46.00	61.74	56.3	50.16	50.02

Table 2: In this table we compare performance of weak experts on additional baseline where we combine all the model (specifically 5 models) by giving each of them equal weight. The column 5 models depicts that. We also mention the baseline shown in the main table which uses the first model. We observe that for all the case the 5-model baseline perform worse or similar to the single model baseline.

J.2 Results on ARC Dataset for generation task

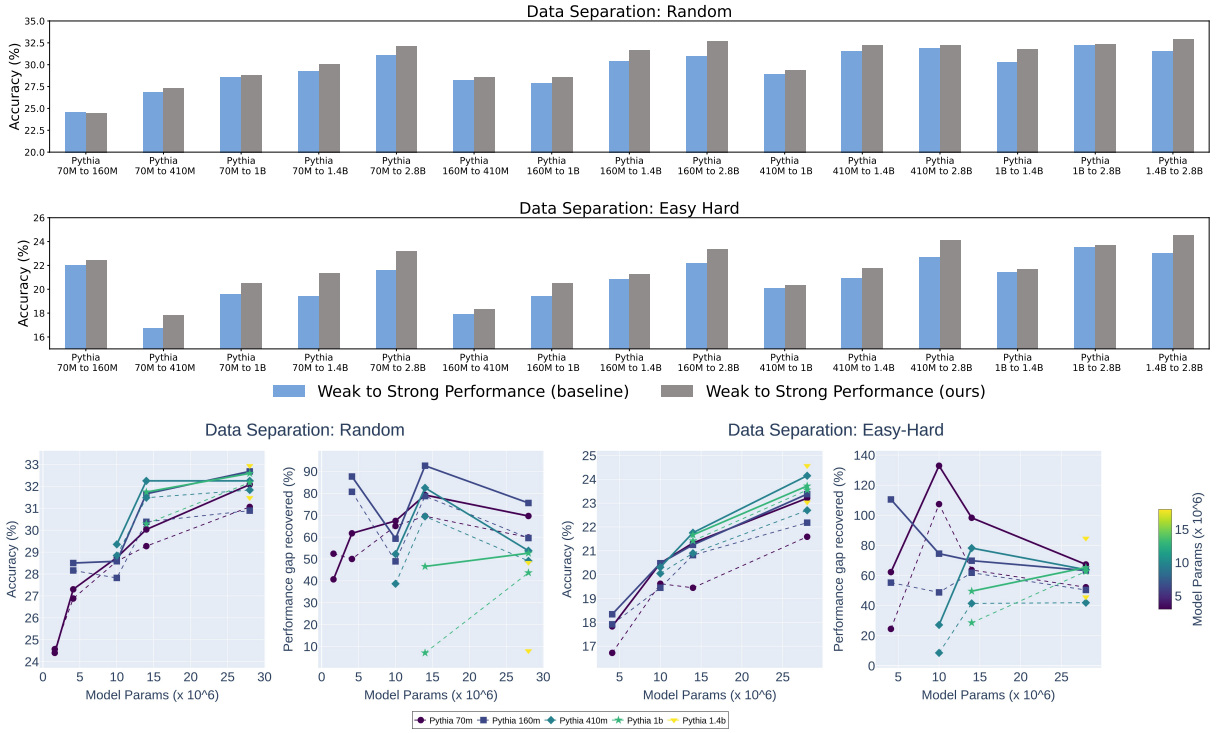

Figure 12: **Generation Task (ARC Data): Top figure** shows a bar plot comparing the w2s generalization of our method (grey) with a baseline (blue) for various combinations of weak and strong model pairs for the SFT task on Q/A data for random data split (top bar-plot) and easy-hard split (bottom bar-plot). **Bottom figure** shows a line plot comparing accuracy and PGR. The left two figures are for random data split, while the right two are for the easy-hard split to show e2h generalization.

J.3 Results on Sciq Dataset for Binary Classification Task

W2S Results with Random Training Data Splits. The baseline of this method is a replication of (Burns et al., 2023). From Figure 13, by applying AdaBoost, we observe a significant improvement in the weak model accuracy, significantly improving the PGR values. In the case of the GPT-2-medium to GPT-2-large pair, we even see the PGR exceeding 100%, meaning that the transfer model has outperformed the strong model's performance. This is the ambitious aim of the w2s generalization problem, and our results show that w2s generalization is achievable.

W2S Results with Easy and Hard Training Data Splits. From Figure 13, we see that applying AdaBoost significantly improves weak model accuracy, thereby enhancing the PGR values. However, for this holistic e2h generalization problem, we are far from reaching the full capability of a strong model. For very small (GPT-2) and large model pairs (GPT-2-xl and above), we do not see improvement in w2s generalization despite the weak models' accuracy improvements. Overall, we observe an improvement of up to 14% in accuracy compared to the baseline and an average improvement of 6.52% and 3% for random and easy-hard splits, respectively.

Scaling Law. In Figure 13 (line plot), we see less PGR recovery for the Qwen-1.8B model even though it is similar in size to GPT-2-xl. Similarly, in the bar plot, we see a drastic difference between the oracle performance of GPT2xl and Qwen-1.8B. This is because the Qwen models series are more capable even after being the same size. Thus, model size is not a good metric, but model capability is a better metric for differentiating between weak and strong models.

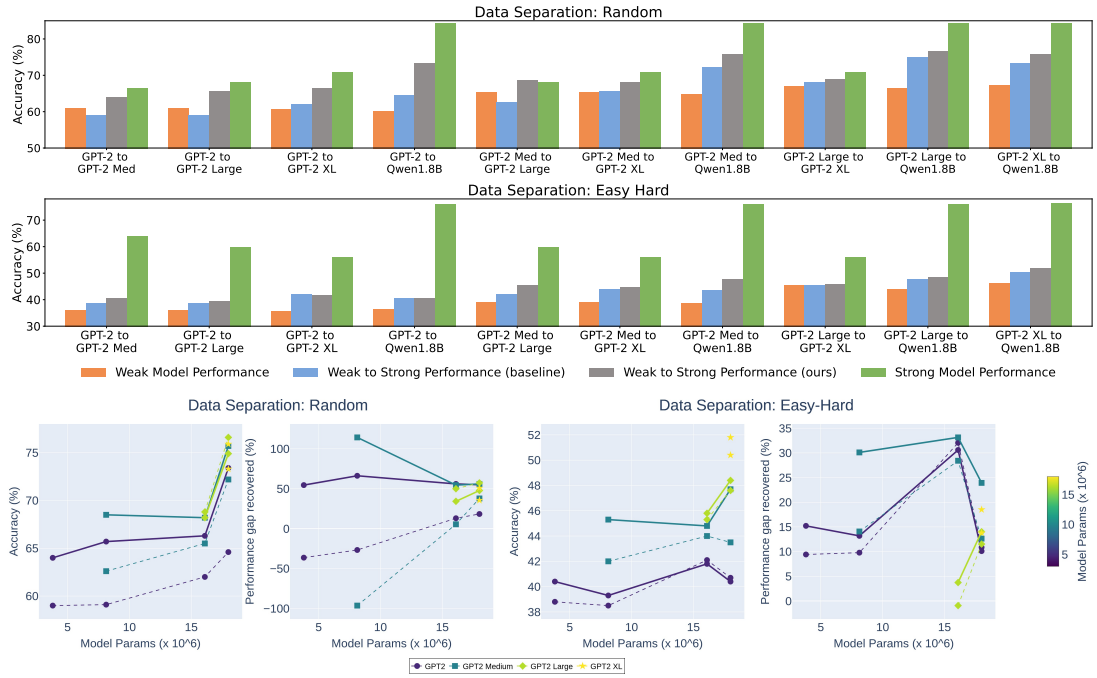

Figure 13: **Binary Classification Task. Top figure** shows a bar plot comparing w2s generalization of our method (grey) with a baseline (blue) from (Burns et al., 2023) using accuracy values(%) for different combinations of weak and strong model pairs for random data split (top bar-plot) and easy-hard split(bottom bar-plot). **Bottom figure** shows a line plot comparing the accuracy and performance gap recovered values (PGR). The left two figures are for random data split, while the right two figures are for the easy-hard split to show e2h generalization.

K Aggregated plots

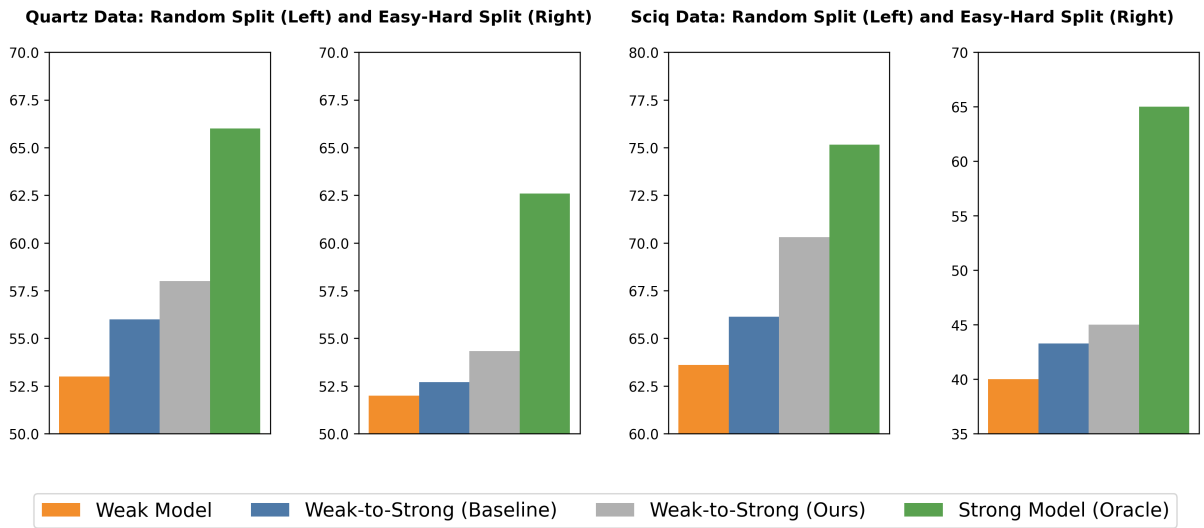

Figure 14: **Aggregated results for Quartz Data on Generation Task and Sciq Data on Binary Classification Task** for both random and easy-hard data splits. We aggregate results for three experimental runs will different seed across all model pairs similar to (Burns et al., 2023).

L Generative Task Details

L.1 Supervised-Fine Tuning task for Quartz Question-Answer Dataset

	Weak Model				Strong Model		
	Token-Avg Acc	Option Acc	Option Acc(on w2s)	α	oracle	Token-Avg Acc	Option Acc
	Pythia-70m				Pythia-160m		
Baseline	17.95 ± 0.44	50.21 ± 0.23	49.7 ± 0.28	10.81 ± 0.04	50.77 ± 0.26	34.3 ± 0.44	51.11 ± 0.23
With Adaboost (T:03)	25.94 ± 0.38	50.64 ± 0.39	49.43 ± 0.25	10.67 ± 0.05	50.77 ± 0.26	34.17 ± 0.36	**51.66 ± 0.45**
	Pythia-70m				Pythia-410m		
Baseline	17.95 ± 0.44	50.21 ± 0.23	49.7 ± 0.28	10.81 ± 0.04	59.18 ± 0.78	50.28 ± 0.44	50.68 ± 0.3
With Adaboost (T:04)	25.22 ± 0.15	50.51 ± 0.53	49.8 ± 0.14	10.68 ± 0.05	59.18 ± 0.78	50.88 ± 0.18	**52.42 ± 0.33**
	Pythia-70m				Pythia-1b		
Baseline	17.95 ± 0.44	50.21 ± 0.23	49.7 ± 0.28	10.81 ± 0.04	63.35 ± 0.3	51.87 ± 0.11	50.89 ± 0.16
With Adaboost (T:05)	26.2 ± 0.06	50.55 ± 0.28	49.65 ± 0.11	10.66 ± 0.04	63.35 ± 0.3	51.83 ± 0.38	**51.83 ± 0.31**
	Pythia-70m				Pythia-1.4b		
Baseline	17.89 ± 0.46	49.87 ± 0.06	49.46 ± 0.35	10.82 ± 0.05	68.83 ± 1.28	51.82 ± 0.05	50.17 ± 0.24
With Adaboost (T:04)	25.32 ± 0.82	50.04 ± 0.37	49.23 ± 0.27	10.7 ± 0.06	68.83 ± 1.28	51.76 ± 0.17	**51.45 ± 0.07**
	Pythia-70m				Pythia-2.8b		
Baseline	18.06 ± 0.39	49.4 ± 0.39	49.73 ± 0.33	10.86 ± 0.02	73.38 ± 1.02	52.28 ± 0.29	50.21 ± 0.23
With Adaboost (T:02)	24.37 ± 0.99	50.13 ± 0.4	49.48 ± 0.21	10.74 ± 0.04	73.38 ± 1.02	52.3 ± 0.14	**51.02 ± 0.22**
	Pythia-160m				Pythia-410m		
Baseline	33.51 ± 0.19	50.81 ± 1.0	49.6 ± 0.27	10.03 ± 0.0	59.18 ± 0.78	50.39 ± 0.3	50.68 ± 0.5
With Adaboost (T:04)	40.85 ± 0.49	51.79 ± 0.48	49.08 ± 0.32	9.81 ± 0.05	59.18 ± 0.78	50.39 ± 0.18	**52.13 ± 0.3**
	Pythia-160m				Pythia-1b		
Baseline	33.51 ± 0.19	50.81 ± 1.0	49.6 ± 0.27	10.03 ± 0.0	63.35 ± 0.3	52.36 ± 0.29	50.6 ± 0.33
With Adaboost (T:02)	40.61 ± 0.8	51.36 ± 0.25	49.93 ± 0.52	9.76 ± 0.05	63.35 ± 0.3	52.45 ± 0.42	**51.92 ± 0.31**
	Pythia-160m				Pythia-1.4b		
Baseline	33.42 ± 0.23	51.4 ± 0.59	49.43 ± 0.41	10.03 ± 0.0	68.83 ± 1.28	52.02 ± 0.2	51.02 ± 0.55
With Adaboost (T:03)	40.87 ± 0.49	51.02 ± 0.18	49.28 ± 0.13	9.75 ± 0.02	68.83 ± 1.28	52.11 ± 0.39	**53.02 ± 0.55**
	Pythia-160m				Pythia-2.8b		
Baseline	33.42 ± 0.23	51.4 ± 0.59	49.43 ± 0.41	10.03 ± 0.0	73.17 ± 0.88	52.82 ± 0.02	51.45 ± 0.5
With Adaboost (T:04)	41.13 ± 0.51	51.23 ± 0.4	49.65 ± 0.14	9.78 ± 0.06	73.17 ± 0.88	52.51 ± 0.3	**51.74 ± 0.17**
	Pythia-410m				Pythia-1b		
Baseline	52.71 ± 0.24	59.27 ± 0.46	55.54 ± 0.49	10.0 ± 0.01	63.35 ± 0.3	53.39 ± 0.2	56.21 ± 0.76
With Adaboost (T:02)	53.39 ± 0.17	58.5 ± 0.33	55.91 ± 0.35	9.69 ± 0.08	63.35 ± 0.3	53.87 ± 0.46	**56.42 ± 0.56**
	Pythia-410m				Pythia-1.4b		
Baseline	52.9 ± 0.09	59.65 ± 0.15	55.66 ± 0.51	9.98 ± 0.02	68.83 ± 1.28	53.33 ± 0.74	56.34 ± 0.9
With Adaboost (T:02)	53.26 ± 0.27	58.8 ± 0.42	56.11 ± 0.34	9.66 ± 0.08	68.83 ± 1.28	54.14 ± 0.63	**57.7 ± 0.61**
	Pythia-410m				Pythia-2.8b		
Baseline	52.13 ± 0.64	58.29 ± 1.1	55.94 ± 0.3	9.89 ± 0.06	73.38 ± 1.02	54.38 ± 0.31	55.74 ± 0.73
With Adaboost (T:04)	53.39 ± 0.19	59.18 ± 0.42	55.32 ± 0.51	9.85 ± 0.05	73.38 ± 1.02	55.71 ± 0.53	**59.01 ± 0.94**
	Pythia-1b				Pythia-1.4b		
Baseline	55.65 ± 0.52	61.99 ± 0.51	58.6 ± 1.13	9.85 ± 0.01	68.62 ± 0.12	55.33 ± 0.31	58.93 ± 0.68
With Adaboost (T:03)	56.81 ± 0.47	62.12 ± 0.43	58.14 ± 0.85	9.74 ± 0.11	68.62 ± 0.12	55.99 ± 0.16	**61.69 ± 0.57**
	Pythia-1b				Pythia-2.8b		
Baseline	55.54 ± 0.6	62.12 ± 0.51	58.55 ± 1.14	9.84 ± 0.01	73.3 ± 0.3	57.26 ± 0.3	61.52 ± 1.38
With Adaboost (T:02)	57.09 ± 0.41	62.84 ± 0.12	59.0 ± 0.62	9.63 ± 0.02	73.3 ± 0.3	58.1 ± 0.08	**63.99 ± 0.93**
	Pythia-1.4b				Pythia-2.8b		
Baseline	57.11 ± 0.45	69.64 ± 0.97	66.87 ± 1.1	9.87 ± 0.02	73.76 ± 0.67	59.34 ± 0.24	67.94 ± 0.78
With Adaboost (T:02)	59.17 ± 0.12	70.66 ± 0.06	67.29 ± 0.77	9.65 ± 0.03	73.76 ± 0.67	59.3 ± 0.34	**68.92 ± 1.06**

Table 3: This table shows weak to strong generalization using random data-splits for quartz dataset. We also study the impact of using ensemble learning methods, which combines weak learners, for weak to strong training. Each model is trained for 5 epochs and uses a learning rate of $5x10^{-5}$. The values in this table are generated by aggregating 3 experiments. We show here mean and Standard Error of the Mean values.

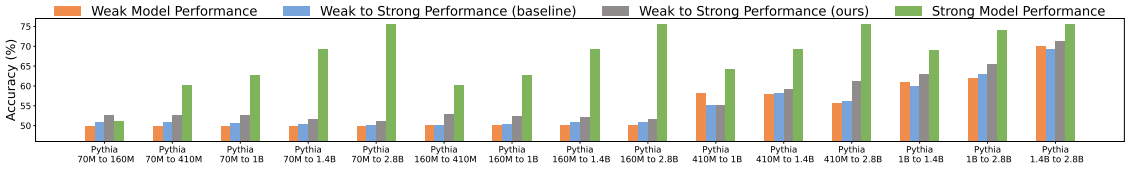

Figure 15: **Quartz Dataset (Random):** This figure shows bar plots comparing accuracy values of weak model performance, w2s model performance (baseline and ours) and strong model performance (oracle) for one specific run of experiments. Values are also mentioned in table 5.

	Weak Model				Strong Model		
	Token-Avg Acc	Option Acc	Option Acc(on w2s)	α	oracle	Token-Avg Acc	Option Acc
	Pythia-70m				Pythia-160m		
Baseline	16.27 ± 0.14	48.0 ± 0.51	49.21 ± 0.05	10.53 ± 0.0	47.11 ± 0.28	29.24 ± 0.18	49.11 ± 0.39
With Adaboost (T:03)	23.31 ± 0.9	47.11 ± 0.31	49.23 ± 0.41	10.43 ± 0.03	47.11 ± 0.28	29.24 ± 0.25	**49.32 ± 0.23**
	Pythia-70m				Pythia-410m		
Baseline	16.27 ± 0.14	48.0 ± 0.51	49.21 ± 0.05	10.53 ± 0.0	52.3 ± 0.39	43.63 ± 0.29	47.32 ± 0.36
With Adaboost (T:04)	23.81 ± 1.01	47.66 ± 0.5	49.06 ± 0.2	10.42 ± 0.02	52.3 ± 0.39	43.53 ± 0.44	**48.13 ± 0.47**
	Pythia-70m				Pythia-1b		
Baseline	16.27 ± 0.14	48.0 ± 0.51	49.21 ± 0.05	10.53 ± 0.0	55.91 ± 0.37	47.48 ± 0.23	47.92 ± 0.23
With Adaboost (T:05)	24.64 ± 0.22	47.49 ± 0.49	49.41 ± 0.38	10.39 ± 0.0	55.91 ± 0.37	45.5 ± 0.74	**49.74 ± 0.24**
	Pythia-70m				Pythia-1.4b		
Baseline	16.07 ± 0.22	48.17 ± 0.43	49.38 ± 0.14	10.58 ± 0.04	65.35 ± 0.66	46.25 ± 0.61	47.96 ± 0.34
With Adaboost (T:04)	23.79 ± 0.55	46.94 ± 0.18	49.58 ± 0.27	10.44 ± 0.04	65.35 ± 0.66	45.53 ± 0.2	**50.68 ± 0.17**
	Pythia-70m				Pythia-2.8b		
Baseline	16.12 ± 0.21	48.85 ± 0.48	49.75 ± 0.32	10.63 ± 0.04	70.2 ± 0.17	48.08 ± 0.18	48.85 ± 0.31
With Adaboost (T:02)	22.96 ± 0.75	47.02 ± 0.12	49.36 ± 0.11	10.5 ± 0.05	70.2 ± 0.17	48.58 ± 0.16	**49.87 ± 0.06**
	Pythia-160m				Pythia-410m		
Baseline	25.61 ± 0.33	47.75 ± 0.35	49.83 ± 0.29	9.96 ± 0.02	52.3 ± 0.39	42.75 ± 0.91	47.75 ± 0.61
With Adaboost (T:04)	29.63 ± 0.55	47.02 ± 0.09	48.47 ± 0.3	9.7 ± 0.09	52.3 ± 0.39	43.78 ± 0.14	**48.42 ± 0.12**
	Pythia-160m				Pythia-1b		
Baseline	25.61 ± 0.33	47.75 ± 0.35	49.83 ± 0.29	9.96 ± 0.02	55.91 ± 0.37	46.08 ± 0.38	49.36 ± 0.53
With Adaboost (T:02)	28.96 ± 0.23	46.43 ± 0.18	48.49 ± 0.11	9.69 ± 0.09	55.91 ± 0.37	44.7 ± 0.58	**49.15 ± 0.73**
	Pythia-160m				Pythia-1.4b		
Baseline	25.76 ± 0.43	47.15 ± 0.15	49.26 ± 0.2	9.96 ± 0.02	65.35 ± 0.66	45.83 ± 0.64	49.7 ± 0.85
With Adaboost (T:03)	28.83 ± 0.84	46.56 ± 0.27	48.17 ± 0.14	9.64 ± 0.06	65.35 ± 0.66	45.4 ± 0.44	**50.0 ± 0.22**
	Pythia-160m				Pythia-2.8b		
Baseline	26.46 ± 0.25	47.49 ± 0.33	48.98 ± 0.14	10.02 ± 0.03	70.2 ± 0.17	48.03 ± 0.13	49.4 ± 0.3
With Adaboost (T:04)	29.61 ± 0.51	46.6 ± 0.25	48.69 ± 0.47	9.54 ± 0.03	70.2 ± 0.17	48.4 ± 0.29	**50.3 ± 0.41**
	Pythia-410m				Pythia-1b		
Baseline	36.73 ± 0.39	51.06 ± 0.39	53.26 ± 0.38	10.07 ± 0.01	55.91 ± 0.37	46.6 ± 0.38	50.72 ± 0.68
With Adaboost (T:02)	38.11 ± 0.44	49.36 ± 0.21	51.66 ± 0.35	9.76 ± 0.14	55.91 ± 0.37	46.4 ± 0.35	**52.09 ± 0.3**
	Pythia-410m				Pythia-1.4b		
Baseline	37.23 ± 0.27	51.11 ± 0.4	53.19 ± 0.42	10.04 ± 0.03	65.35 ± 0.66	47.73 ± 0.78	53.66 ± 0.56
With Adaboost (T:02)	38.31 ± 0.23	50.17 ± 0.44	51.56 ± 0.22	9.53 ± 0.09	65.35 ± 0.66	48.35 ± 0.18	**53.36 ± 0.5**
	Pythia-410m				Pythia-2.8b		
Baseline	37.13 ± 0.23	51.02 ± 0.47	52.87 ± 0.21	10.03 ± 0.03	70.2 ± 0.17	48.48 ± 0.36	54.47 ± 0.16
With Adaboost (T:04)	38.13 ± 0.26	49.87 ± 0.68	51.49 ± 0.28	9.6 ± 0.04	70.2 ± 0.17	49.05 ± 0.14	**55.36 ± 0.47**
	Pythia-1b				Pythia-1.4b		
Baseline	40.3 ± 0.46	54.51 ± 0.73	54.25 ± 0.26	10.33 ± 0.08	66.67 ± 0.72	47.0 ± 0.22	56.76 ± 0.58
With Adaboost (T:03)	40.75 ± 0.67	53.36 ± 0.92	53.61 ± 0.44	11.0 ± 0.72	66.67 ± 0.72	47.25 ± 0.32	**57.23 ± 0.37**
	Pythia-1b				Pythia-2.8b		
Baseline	40.33 ± 0.44	54.08 ± 1.07	54.33 ± 0.19	10.33 ± 0.08	73.09 ± 0.42	49.2 ± 0.2	58.08 ± 0.38
With Adaboost (T:02)	40.53 ± 0.34	52.34 ± 0.09	53.39 ± 0.2	11.68 ± 0.75	73.09 ± 0.42	49.48 ± 0.3	**59.35 ± 0.52**
	Pythia-1.4b				Pythia-2.8b		
Baseline	42.2 ± 1.12	59.69 ± 0.83	62.39 ± 1.06	10.3 ± 0.1	73.17 ± 0.38	51.22 ± 0.5	62.46 ± 0.91
With Adaboost (T:02)	42.98 ± 0.64	59.82 ± 0.51	61.38 ± 0.48	10.52 ± 0.35	73.17 ± 0.38	51.72 ± 0.37	**63.01 ± 0.28**

Table 4: This table shows weak to strong generalization using easy-hard data-splits for quartz dataset. We also study the impact of using ensemble learning methods, which combines weak learners, for weak to strong training. Each model is trained for 5 epochs and uses a learning rate of 5×10^{-5}. The values in this table are generated by aggregating 3 experiments. We show here mean and Standard Error of the Mean values.

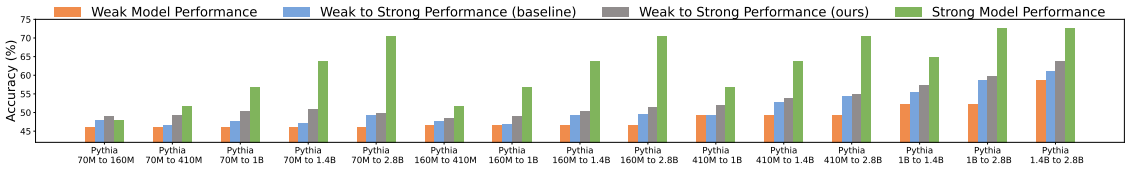

Figure 16: **Quartz Dataset (Easy-Hard):** This figure shows bar plots comparing accuracy values of weak model performance, w2s model performance (baseline and ours) and strong model performance (oracle) for one specific run of experiments. Values are also mentioned in table 5.

Weak Model Size	Strong Model Size	Data Separation: Random W2S Performance		Improv(%)	Data Separation: Easy-Hard W2S Performance		Improv(%)
		Baseline	Ours		Baseline	Ours	
Pythia-70M	Pythia-160M	0.5077	0.5255	3.5%	0.48	0.4898	2%
Pythia-70M	Pythia-410M	0.5077	0.5255	3.5%	0.4643	0.4923	6%
Pythia-70M	Pythia-1B	0.5051	0.5255	4%	0.4758	0.5026	5.6%
Pythia-70M	Pythia-1.4B	0.5026	0.5153	2.5%	0.4719	0.5089	7.8%
Pythia-70M	Pythia-2.8B	0.5	0.5115	2.3%	0.4923	0.4987	1.3%
Pythia-160M	Pythia-410M	0.5	0.5281	5.6%	0.4758	0.4834	1.6%
Pythia-160M	Pythia-1B	0.5026	0.523	4.1%	0.4681	0.4898	4.6%
Pythia-160M	Pythia-1.4B	0.5077	0.5217	2.8%	0.4936	0.5038	2.1%
Pythia-160M	Pythia-2.8B	0.5077	0.5153	1.5%	0.4949	0.5128	3.6%
Pythia-410M	Pythia-1B	0.551	0.551	0%	0.4921	0.5179	5.2%
Pythia-410M	Pythia-1.4B	0.5816	0.5918	1.8%	0.5268	0.537	1.9%
Pythia-410M	Pythia-2.8B	0.5599	0.611	9.1%	0.5434	0.5485	0.9%
Pythia-1B	Pythia-1.4B	0.5982	0.6288	5.1%	0.5536	0.574	3.7%
Pythia-1B	Pythia-2.8B	0.6288	0.6543	4.1%	0.5855	0.5957	1.7%
Pythia-1.4B	Pythia-2.8B	0.6926	0.713	2.9%	0.6161	0.6288	2.1%

Table 5: This table shows weak to strong generalization using random as well as easy-hard data-splits for quartz dataset. As compared to previous tables 3 and 4, here we run experiment once and note the improvement of our method with respect to the baseline.

L.2 Supervised-Fine Tuning task for ARC Question-Answer Dataset

	Weak Model				Strong Model		
	Token-Avg Acc	Option Acc	Option Acc(on w2s)	α	oracle	Token-Avg Acc	Option Acc
	Pythia-70m				Pythia-160m		
Baseline	13.28 ± 0.05	25.31 ± 0.1	25.76 ± 0.94	10.73 ± 0.03	24.12 ± 0.48	26.91 ± 0.1	24.46 ± 0.06
With Adaboost (T:03)	17.93 ± 0.78	24.75 ± 0.76	25.82 ± 0.69	10.68 ± 0.02	24.12 ± 0.48	27.15 ± 0.36	**24.23 ± 0.08**
	Pythia-70m				Pythia-410m		
Baseline	13.28 ± 0.05	25.31 ± 0.1	25.76 ± 0.94	10.73 ± 0.03	28.61 ± 0.08	41.29 ± 0.1	27.25 ± 0.24
With Adaboost (T:04)	17.94 ± 0.88	24.97 ± 0.69	25.82 ± 0.69	10.67 ± 0.04	28.61 ± 0.08	41.61 ± 0.02	**27.27 ± 0.3**
	Pythia-70m				Pythia-1b		
Baseline	13.28 ± 0.05	25.31 ± 0.1	25.76 ± 0.94	10.73 ± 0.03	31.11 ± 0.02	45.13 ± 0.11	28.33 ± 0.18
With Adaboost (T:05)	19.7 ± 1.18	24.92 ± 0.28	26.23 ± 0.49	10.65 ± 0.04	31.11 ± 0.02	45.17 ± 0.11	**28.52 ± 0.09**
	Pythia-70m				Pythia-1.4b		
Baseline	13.35 ± 0.06	25.06 ± 0.14	24.39 ± 0.42	10.77 ± 0.06	32.34 ± 0.3	45.21 ± 0.24	29.86 ± 0.28
With Adaboost (T:04)	19.75 ± 1.16	24.26 ± 0.56	25.7 ± 0.65	10.68 ± 0.05	32.34 ± 0.3	45.33 ± 0.14	**30.35 ± 0.13**
	Pythia-70m				Pythia-2.8b		
Baseline	13.42 ± 0.11	24.63 ± 0.13	23.97 ± 0.55	10.77 ± 0.05	35.18 ± 0.02	48.07 ± 0.12	30.94 ± 0.13
With Adaboost (T:02)	19.88 ± 0.56	24.52 ± 0.49	24.87 ± 0.81	10.68 ± 0.04	35.18 ± 0.02	47.75 ± 0.08	**31.43 ± 0.43**
	Pythia-160m				Pythia-410m		
Baseline	25.5 ± 0.66	24.12 ± 0.45	26.06 ± 0.68	9.89 ± 0.03	29.18 ± 0.04	41.39 ± 0.14	27.5 ± 0.27
With Adaboost (T:04)	31.95 ± 0.47	24.94 ± 0.29	25.88 ± 0.64	9.74 ± 0.03	29.18 ± 0.04	41.28 ± 0.03	**27.7 ± 0.34**
	Pythia-160m				Pythia-1b		
Baseline	25.5 ± 0.66	24.12 ± 0.45	26.06 ± 0.68	9.89 ± 0.03	31.26 ± 0.44	45.12 ± 0.05	28.24 ± 0.18
With Adaboost (T:02)	32.25 ± 0.21	24.52 ± 0.34	26.06 ± 0.57	9.66 ± 0.01	31.26 ± 0.44	45.18 ± 0.14	**28.47 ± 0.24**
	Pythia-160m				Pythia-1.4b		
Baseline	24.74 ± 0.14	23.97 ± 0.36	25.76 ± 0.51	9.86 ± 0.02	32.25 ± 0.35	45.01 ± 0.1	30.55 ± 0.07
With Adaboost (T:03)	32.55 ± 0.21	24.46 ± 0.22	26.12 ± 0.8	9.66 ± 0.01	32.25 ± 0.35	45.23 ± 0.05	**30.86 ± 0.33**
	Pythia-160m				Pythia-2.8b		
Baseline	25.43 ± 0.66	24.34 ± 0.09	26.0 ± 0.32	9.86 ± 0.02	35.44 ± 0.06	47.88 ± 0.02	31.03 ± 0.15
With Adaboost (T:04)	32.6 ± 0.03	24.23 ± 0.18	26.47 ± 0.53	9.66 ± 0.02	35.44 ± 0.06	47.77 ± 0.08	**31.68 ± 0.41**
	Pythia-410m				Pythia-1b		
Baseline	39.76 ± 0.3	27.85 ± 0.52	24.33 ± 0.97	9.39 ± 0.02	30.97 ± 0.08	44.94 ± 0.08	28.9 ± 0.12
With Adaboost (T:02)	40.69 ± 0.14	28.27 ± 0.11	24.33 ± 0.59	9.01 ± 0.04	30.97 ± 0.08	44.76 ± 0.14	**29.41 ± 0.08**
	Pythia-410m				Pythia-1.4b		
Baseline	39.66 ± 0.22	27.82 ± 0.53	24.09 ± 0.8	9.39 ± 0.02	32.82 ± 0.27	45.54 ± 0.03	30.26 ± 0.56
With Adaboost (T:02)	40.82 ± 0.13	28.9 ± 0.21	24.51 ± 0.59	9.01 ± 0.04	32.82 ± 0.27	45.66 ± 0.09	**30.94 ± 0.53**
	Pythia-410m				Pythia-2.8b		
Baseline	39.57 ± 0.24	28.01 ± 0.69	24.69 ± 0.44	9.39 ± 0.01	35.86 ± 0.26	48.06 ± 0.15	31.15 ± 0.3
With Adaboost (T:04)	40.56 ± 0.11	28.7 ± 0.34	25.34 ± 1.12	9.03 ± 0.07	35.86 ± 0.26	48.22 ± 0.12	**31.88 ± 0.27**
	Pythia-1b				Pythia-1.4b		
Baseline	42.31 ± 0.2	30.35 ± 0.24	28.02 ± 0.76	9.53 ± 0.02	32.65 ± 0.43	45.41 ± 0.06	30.26 ± 0.22
With Adaboost (T:03)	43.22 ± 0.13	31.68 ± 0.55	27.79 ± 0.71	9.37 ± 0.01	32.65 ± 0.43	45.44 ± 0.06	**31.28 ± 0.22**
	Pythia-1b				Pythia-2.8b		
Baseline	42.2 ± 0.29	30.46 ± 0.16	27.73 ± 0.89	9.53 ± 0.02	35.12 ± 0.26	48.12 ± 0.06	32.14 ± 0.02
With Adaboost (T:02)	43.61 ± 0.2	31.17 ± 0.93	27.79 ± 0.76	9.26 ± 0.02	35.12 ± 0.26	48.2 ± 0.08	**32.54 ± 0.08**
	Pythia-1.4b				Pythia-2.8b		
Baseline	42.39 ± 0.37	33.42 ± 0.37	30.65 ± 1.82	9.48 ± 0.03	35.12 ± 0.26	48.35 ± 0.11	32.42 ± 0.44
With Adaboost (T:02)	43.58 ± 0.27	33.5 ± 0.22	30.71 ± 1.48	11.07 ± 0.84	35.12 ± 0.26	48.29 ± 0.13	**33.19 ± 0.24**

Table 6: This table shows weak to strong generalization using random data-splits for arc dataset. We also study the impact of using ensemble learning methods, which combines weak learners, for weak to strong training. Each model is trained for 5 epochs and uses a learning rate of $5x10^{-5}$. The values in this table are generated by aggregating 3 experiments. We show here mean and Standard Error of the Mean values.

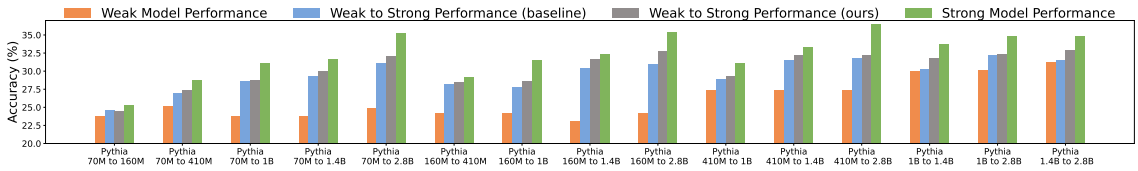

Figure 17: **ARC Dataset (Random):** This figure shows bar plots comparing accuracy values of weak model performance, w2s model performance (baseline and ours) and strong model performance (oracle) for one specific run of experiments. Values are also mentioned in table 7 and 6.

	Weak Model				Strong Model		
	Token-Avg Acc	Option Acc	Option Acc(on w2s)	α	oracle	Token-Avg Acc	Option Acc
	Pythia-70m				Pythia-160m		
Baseline	8.17 ± 0.06	22.5 ± 0.33	27.85 ± 0.57	10.45 ± 0.0	22.3 ± 0.16	17.88 ± 0.11	22.27 ± 0.32
With Adaboost (T:03)	13.35 ± 0.54	22.81 ± 0.29	27.78 ± 0.46	10.35 ± 0.02	22.3 ± 0.16	17.87 ± 0.17	**22.56 ± 0.06**
	Pythia-70m				Pythia-410m		
Baseline	8.17 ± 0.06	22.5 ± 0.33	27.85 ± 0.57	10.45 ± 0.0	19.28 ± 0.15	28.92 ± 0.14	17.06 ± 0.31
With Adaboost (T:04)	14.53 ± 0.72	22.93 ± 0.17	27.96 ± 0.46	10.32 ± 0.0	19.28 ± 0.15	28.84 ± 0.05	**18.0 ± 0.07**
	Pythia-70m				Pythia-1b		
Baseline	8.17 ± 0.06	22.5 ± 0.33	27.85 ± 0.57	10.45 ± 0.0	21.5 ± 0.24	32.05 ± 0.13	19.96 ± 0.15
With Adaboost (T:05)	12.95 ± 0.88	22.58 ± 0.38	28.03 ± 0.21	10.35 ± 0.02	21.5 ± 0.24	31.84 ± 0.08	**20.45 ± 0.06**
	Pythia-70m				Pythia-1.4b		
Baseline	8.23 ± 0.1	22.61 ± 0.42	27.37 ± 0.42	10.45 ± 0.0	21.76 ± 0.14	32.98 ± 0.04	20.45 ± 0.42
With Adaboost (T:04)	12.65 ± 0.05	23.24 ± 0.06	28.32 ± 0.76	10.33 ± 0.01	21.76 ± 0.14	32.95 ± 0.17	**21.28 ± 0.02**
	Pythia-70m				Pythia-2.8b		
Baseline	8.33 ± 0.1	23.24 ± 0.23	27.19 ± 0.47	10.45 ± 0.0	26.59 ± 0.13	35.98 ± 0.09	22.78 ± 0.51
With Adaboost (T:02)	14.28 ± 0.15	23.26 ± 0.22	28.27 ± 0.14	10.37 ± 0.01	26.59 ± 0.13	35.86 ± 0.28	**23.15 ± 0.2**
	Pythia-160m				Pythia-410m		
Baseline	17.46 ± 0.16	21.73 ± 0.35	26.95 ± 0.1	9.61 ± 0.0	19.11 ± 0.37	28.8 ± 0.23	18.15 ± 0.15
With Adaboost (T:04)	20.57 ± 0.1	22.16 ± 0.2	27.19 ± 0.5	9.22 ± 0.02	19.11 ± 0.37	28.9 ± 0.11	**18.43 ± 0.04**
	Pythia-160m				Pythia-1b		
Baseline	17.46 ± 0.16	21.73 ± 0.35	26.95 ± 0.1	9.61 ± 0.0	21.59 ± 0.07	32.06 ± 0.06	19.65 ± 0.1
With Adaboost (T:02)	20.47 ± 0.09	22.27 ± 0.29	27.31 ± 0.51	9.24 ± 0.01	21.59 ± 0.07	32.07 ± 0.12	**20.17 ± 0.14**
	Pythia-160m				Pythia-1.4b		
Baseline	17.61 ± 0.07	22.84 ± 0.58	27.79 ± 0.64	9.61 ± 0.0	22.33 ± 0.34	33.11 ± 0.1	21.19 ± 0.15
With Adaboost (T:03)	20.31 ± 0.24	22.5 ± 0.36	27.79 ± 0.42	9.27 ± 0.06	22.33 ± 0.34	33.01 ± 0.05	**21.25 ± 0.28**
	Pythia-160m				Pythia-2.8b		
Baseline	17.64 ± 0.06	23.09 ± 0.54	27.91 ± 0.59	9.6 ± 0.01	26.82 ± 0.1	35.83 ± 0.36	22.44 ± 0.11
With Adaboost (T:04)	20.3 ± 0.19	23.01 ± 0.43	27.73 ± 0.25	9.26 ± 0.06	26.82 ± 0.1	36.06 ± 0.07	**23.35 ± 0.1**
	Pythia-410m				Pythia-1b		
Baseline	27.3 ± 0.16	18.8 ± 0.21	31.01 ± 0.51	9.24 ± 0.0	21.33 ± 0.04	32.06 ± 0.07	20.05 ± 0.08
With Adaboost (T:02)	28.07 ± 0.12	18.35 ± 0.21	32.2 ± 0.31	8.68 ± 0.09	21.33 ± 0.04	32.36 ± 0.05	**20.34 ± 0.06**
	Pythia-410m				Pythia-1.4b		
Baseline	27.5 ± 0.14	18.54 ± 0.32	31.6 ± 0.21	9.24 ± 0.0	22.36 ± 0.3	33.47 ± 0.07	21.13 ± 0.1
With Adaboost (T:02)	28.09 ± 0.08	18.17 ± 0.28	31.78 ± 0.4	8.67 ± 0.09	22.36 ± 0.3	33.18 ± 0.11	**21.47 ± 0.12**
	Pythia-410m				Pythia-2.8b		
Baseline	27.48 ± 0.13	18.12 ± 0.13	31.66 ± 0.17	9.25 ± 0.01	26.03 ± 0.21	36.13 ± 0.09	23.07 ± 0.18
With Adaboost (T:04)	27.96 ± 0.11	18.09 ± 0.2	31.07 ± 0.27	8.69 ± 0.08	26.03 ± 0.21	35.93 ± 0.09	**24.06 ± 0.15**
	Pythia-1b				Pythia-1.4b		
Baseline	30.64 ± 0.17	21.22 ± 0.72	32.5 ± 0.6	9.38 ± 0.01	22.01 ± 0.21	33.13 ± 0.11	21.5 ± 0.07
With Adaboost (T:03)	30.41 ± 0.42	21.11 ± 0.22	32.68 ± 0.56	10.98 ± 0.78	22.01 ± 0.21	33.31 ± 0.03	**21.53 ± 0.08**
	Pythia-1b				Pythia-2.8b		
Baseline	30.64 ± 0.17	21.22 ± 0.72	32.5 ± 0.6	9.38 ± 0.01	25.51 ± 0.2	36.14 ± 0.11	23.75 ± 0.16
With Adaboost (T:02)	31.11 ± 0.12	21.67 ± 0.18	33.21 ± 0.56	9.4 ± 0.24	25.51 ± 0.2	36.13 ± 0.13	**23.75 ± 0.06**
	Pythia-1.4b				Pythia-2.8b		
Baseline	31.09 ± 0.12	22.27 ± 0.55	34.05 ± 0.1	9.31 ± 0.01	25.26 ± 0.11	36.13 ± 0.05	23.49 ± 0.2
With Adaboost (T:02)	31.56 ± 0.1	21.79 ± 0.44	34.35 ± 0.59	10.89 ± 0.65	25.26 ± 0.11	36.36 ± 0.2	**24.37 ± 0.16**

Table 7: This table shows weak to strong generalization using easy-hard data-splits for ARC dataset. We also study the impact of using ensemble learning methods, which combines weak learners, for weak to strong training. Each model is trained for 5 epochs and uses a learning rate of 5×10^{-5}. The values in this table are generated by aggregating 3 experiments. We show here mean and Standard Error of the Mean values.

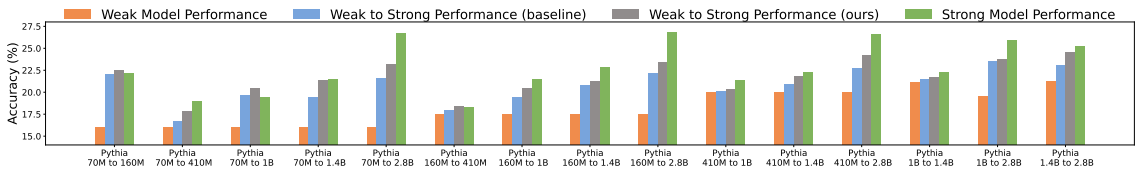

Figure 18: **ARC Dataset (Easy-Hard):** This figure shows bar plots comparing accuracy values of weak model performance, w2s model performance (baseline and ours) and strong model performance (oracle) for one specific run of experiments. Values are also mentioned in table 7 and 6.

M Training error decay for EnsemW2S

Let tokens be indexed by $\nu = (i, j)$ with $i \in [m]$, $j \in [k_i]$ and $n = \sum_{i=1}^{m} k_i$. At round t let

$$\epsilon_t = \sum_{i=1}^{m} \sum_{j=1}^{k_i} D_t(i,j) \mathbb{1}\Big[h_\theta^t\big(\boldsymbol{x}_i^e, \boldsymbol{y}_i^{e,1:j-1}\big) \neq y_i^{e,j}\Big]. \tag{1}$$

Indicator-only weight update. The algorithm uses

$$D_{t+1}(i,j) = \frac{D_t(i,j)\exp\big(\alpha_t \mathbb{1}[h_\theta^t \neq y]\big)}{Z_t^{\text{alg}}}, \qquad \boxed{\alpha_t = \log\frac{1-\epsilon_t}{\epsilon_t} + \log r = \log\frac{r(1-\epsilon_t)}{\epsilon_t}}, \tag{2}$$

with $r = \frac{\epsilon_{\text{pre}}}{1-\epsilon_{\text{pre}}} > 0$ and the *algorithmic* normalizer

$$\boxed{Z_t^{\text{alg}} = (1-\epsilon_t) + \epsilon_t e^{\alpha_t} = (1+r)(1-\epsilon_t) = \frac{1-\epsilon_t}{1-\epsilon_{\text{pre}}} \geq 1}. \tag{3}$$

(Remark: (3) merely reweights the sample; it is not the shrink factor in the proof.)

Votes and margin. Define the votes and the predicted class

$$F_k^{(T)}(i,j) = \sum_{t=1}^{T} \alpha_t \mathbb{1}\Big[h_\theta^t(\boldsymbol{x}_i^e, \boldsymbol{y}_i^{e,1:j-1}) = k\Big], \qquad H(\boldsymbol{x}_i^e, \boldsymbol{y}_i^{e,1:j-1}) = \arg\max_k F_k^{(T)}. \tag{4}$$

Use the r-average multiclass margin

$$\boxed{m_{i,j}^{(T)} = F_{y_i^{e,j}}^{(T)}(i,j) \; - \; \frac{1}{r} \sum_{k \neq y_i^{e,j}} F_k^{(T)}(i,j)}. \tag{5}$$

When the weak learner is correct at (i, j), the margin increases by $+\alpha_t$; when it errs, it decreases by $-\alpha_t/r$:

$$\Delta m_{i,j}^{(t)} = \begin{cases} +\alpha_t, & h_\theta^t = y_i^{e,j}, \\ -\alpha_t/r, & h_\theta^t \neq y_i^{e,j}. \end{cases} \tag{6}$$

Potential and temperature. Let

$$\boxed{\lambda = \frac{r}{r+1} \in (0,1), \qquad W_t = \frac{1}{n}\sum_{i=1}^{m}\sum_{j=1}^{k_i} \exp\big(-\lambda\, m_{i,j}^{(t)}\big)}. \tag{7}$$

Using (6) and this λ,

$$\frac{W_{t+1}}{W_t} = (1-\epsilon_t)e^{-\lambda\alpha_t} + \epsilon_t e^{\lambda\alpha_t/r} =: \Phi_t(\alpha_t). \tag{8}$$

Choosing α_t (consistency with the code). Minimizing Φ_t gives

$$0 = \Phi_t'(\alpha) = -(1-\epsilon_t)\lambda e^{-\lambda\alpha} + \epsilon_t \frac{\lambda}{r} e^{\lambda\alpha/r} \iff \alpha = \frac{r}{r+1}\log\frac{r(1-\epsilon_t)}{\epsilon_t}, \tag{9}$$

which equals α_t in (2). Therefore the *proof* shrink factor is

$$\boxed{Z_t^{\text{proof}} := \Phi_t(\alpha_t) = (r+1)\,(1-\epsilon_t)^{\frac{1}{r+1}} \left(\frac{\epsilon_t}{r}\right)^{\frac{r}{r+1}}}. \tag{10}$$

Telescoping and 0–1 error bound. Since $W_{t+1} = W_t\, Z_t^{\text{proof}}$, we have $W_{T+1} = W_1 \prod_{t=1}^{T} Z_t^{\text{proof}}$ and $W_1 = 1$ (because $F^{(0)} \equiv 0$). If H misclassifies (i,j), then $m_{i,j}^{(T)} \le 0$, whence $\mathbb{1}[H \neq y] \le e^{-\lambda m_{i,j}^{(T)}}$. Thus

$$\boxed{\mathrm{Err}_{\text{train}} = \frac{1}{n}\sum_{i,j} \mathbb{1}[H \neq y] \;\le\; W_{T+1} = \prod_{t=1}^{T} Z_t^{\text{proof}}}. \tag{11}$$

Weak-learning condition and exponential decay. From (10),

$$Z_t^{\text{proof}} < 1 \iff \epsilon_t < \frac{r}{r+1} = \epsilon_{\text{pre}}. \tag{12}$$

If (12) holds for all t, the product in (11) decays exponentially in T.

Note: The reweighting constant Z_t^{alg} in (3) may exceed 1; this does not affect the proof. Decay is governed by Z_t^{proof} in (10).

N Margin theory for Generalization bound (For Token-Level Ensembling)

Define the small-margin set

$$\mathcal{M}(\theta) = \{(i,j) : m_{i,j}^{(T)} \le \theta\}. \tag{13}$$

By Markov's inequality and (7),

$$\Pr_{(i,j)\in S}\big[m_{i,j}^{(T)} \le \theta\big] \;\le\; e^{\lambda\theta}\, W_{T+1} \;=\; e^{\lambda\theta} \prod_{t=1}^{T} Z_t^{\text{proof}}. \tag{14}$$

Let d be the complexity of the weak-learner family (VC dimension, or Natarajan dimension for multiclass). A standard multiclass margin bound (Hastie et al., 2009; Freund and Schapire, 1997) applied to the r-margin with $\lambda = \frac{r}{r+1}$ yields: for any $\theta \in (0,1]$ and $\delta \in (0,1)$, with probability at least $1-\delta$ over an i.i.d. n-token sample,

$$\boxed{\Pr_{(x,y)\sim\mathcal{D}}\big[H(x) \neq y\big] \;\le\; \Pr_{(i,j)\in S}\big[m_{i,j}^{(T)} \le \theta\big] + \sqrt{\frac{c\big(d\log n + \log(1/\delta)\big)}{n\,\lambda^2\theta^2}}}, \tag{15}$$

for a universal constant c. Combining (14) with (15) gives the final bound specialized to your algorithm:

$$\boxed{\Pr_{\mathcal{D}}[H \neq y] \;\le\; \exp(\lambda\theta) \prod_{t=1}^{T} \left[(r+1)\,(1-\epsilon_t)^{\frac{1}{r+1}} \left(\frac{\epsilon_t}{r}\right)^{\frac{r}{r+1}}\right] \;+\; \sqrt{\frac{c\big(d\log n + \log(1/\delta)\big)}{n\,\lambda^2\theta^2}}}, \tag{16}$$

with $\lambda = \frac{r}{r+1}$ and $r = \frac{\epsilon_{\text{pre}}}{1-\epsilon_{\text{pre}}}$.

Remarks. (i) If $r = K-1$ (i.e. $\epsilon_{\text{pre}} = 1 - \frac{1}{K}$), (10)–(16) reduce to the SAMME formulas of (Hastie et al., 2009).
(ii) If tokens are dependent (autoregressive), (11) is unchanged (deterministic on the training set). In (16), one may replace n with an effective sample size n_{eff} under a dependence model (e.g. β-mixing) and optionally weight sequences first to avoid long-sequence dominance.

O Non-IID Margin Generalization

The training-set decay (10)–(11) is deterministic and needs no independence. To pass from empirical margins to *test* error under dependence, we control one quantity:

$$\bar{p}_n(\theta) := \frac{1}{n}\sum_{t=1}^{n} \mathbb{E}\Big[\mathbb{1}\{m_t^{(T)} \le \theta\} \,\Big|\, \mathcal{F}_{t-1}\Big],$$

the average (over time) *conditional* probability that the margin at the next token is at most θ, given the history $\mathcal{F}_{t-1}$. This is the natural "next-step" risk for an autoregressive process; if the process is stationary (or mixing), $\bar{p}_n(\theta)$ coincides with the population probability up to vanishing bias.

Step 1: A Martingale Deviation Bound (Azuma). Let $X_t = \mathbb{1}\{m_t^{(T)} \le \theta\} \in [0,1]$ and define the martingale differences $D_t = X_t - \mathbb{E}[X_t \mid \mathcal{F}_{t-1}]$. Then $\{S_n = \sum_{t=1}^n D_t\}$ is a martingale with $|D_t| \le 1$. By Azuma–Hoeffding, for any $\delta \in (0,1)$, with probability at least $1-\delta$,

$$\bar{p}_n(\theta) = \frac{1}{n}\sum_{t=1}^{n} \mathbb{E}[X_t \mid \mathcal{F}_{t-1}] \;\le\; \frac{1}{n}\sum_{t=1}^{n} X_t + \sqrt{\frac{\log(2/\delta)}{2n}}. \tag{NI-1}$$

Step 2: Bound the Empirical Small-Margin mass by the Potential. By Markov's inequality and the potential definition (7),

$$\frac{1}{n}\sum_{t=1}^{n} X_t \;=\; \Pr_{(i,j)\in S}\big[m_{i,j}^{(T)} \le \theta\big] \;\le\; e^{\lambda\theta}\, W_{T+1} \;=\; e^{\lambda\theta}\prod_{t=1}^{T} Z_t^{\text{proof}}. \tag{NI-2}$$

Step 3: Combine. From (NI-1) and (NI-2), with probability at least $1-\delta$,

$$\boxed{\bar{p}_n(\theta) \;\le\; \exp(\lambda\theta)\prod_{t=1}^{T} Z_t^{\text{proof}} \;+\; \sqrt{\frac{\log(2/\delta)}{2n}}\,.} \tag{NI-3}$$

From margins to classification error. Since $\mathbb{1}[H(X) \ne Y] \le \mathbb{1}[m^{(T)}(X) \le \theta]$ for any $\theta > 0$, the *next-step* misclassification probability satisfies

$$\frac{1}{n}\sum_{t=1}^{n} \mathbb{E}[\mathbb{1}\{H(X_t) \ne Y_t\} \,|\, \mathcal{F}_{t-1}] \;\le\; \bar{p}_n(\theta).$$

Combining with (NI-3) yields the non-IID margin bound:

$$\boxed{\frac{1}{n}\sum_{t=1}^{n} \mathbb{E}[\mathbb{1}\{H(X_t) \ne Y_t\} \,|\, \mathcal{F}_{t-1}] \;\le\; \exp(\lambda\theta)\prod_{t=1}^{T} Z_t^{\text{proof}} \;+\; \sqrt{\frac{\log(2/\delta)}{2n}}\,.} \tag{NI-4}$$

Interpretation. The term on the left is the *test risk for the dependent process* (the average conditional error the ensemble would incur when run along the same data-generating dynamics). If the token process is stationary ergodic (or mixing), this equals the population test error up to $o_n(1)$. Crucially, the first term on the right is identical to the IID case—dependence only adds the simple $O\big(\sqrt{\frac{\log(1/\delta)}{n}}\big)$ slack.