CSE 493s report: Knowledge distillation

Tim Avilov, Federico Baldan, Shreyan Mitra, Bryan Zhao

February 11, 2026

0 Code base

The github repo link is available here.

1 Introduction

Large language models (LLMs) designed for biomedical applications have achieved impressive performance on clinical question answering, medical reasoning, and evidence-grounded analysis. However, their high computational cost makes them impractical for many real-world medical use cases, especially in settings where inference must be fast, inexpensive, and deployable on modest hardware. Knowledge distillation provides a promising framework for transferring a large medical domain expert model's knowledge into a much smaller student model while retaining essential reasoning capabilities.

In this project, we study the feasibility of compressing **MedItron-7B**, a medical-specialized teacher model, into **TinyLlama-1.1B**, a lightweight general-purpose student model. Our goal is to identify which distillation strategies are most effective for preserving medical reasoning, accuracy, and evidence-grounded behavior under realistic computational constraints. To support this evaluation, we compile **Med-DistillMix-120k**, a unified dataset of 120,000 medically oriented examples drawn from diverse sources. This dataset is partitioned into training, validation, and held-out splits to ensure rigorous and reproducible experimentation.

We design and analyze a suite of ten distillation methods, including six supervised (non-RL) techniques and four reinforcement-learning-based methods. Because of limited computational resources, we will ultimately execute only two of these methods end-to-end, but the project is structured so that all ten approaches are defined, compared conceptually, and incorporated into our analytical framework. This enables us to reason about the relative strengths, weaknesses, and practical feasibility of the broader method landscape even when only a subset can be run experimentally.

To assess the resulting distilled models, we benchmark them on four medical evaluation tasks—**MedQA**, **MedMCQA**, **PubMedQA**, and **PubHealth**—all of which are held out from the training pool to avoid leakage. We further compute perplexity on a separate biomedical text corpus, **MedPPL-10k**, to measure general language modeling quality. To evaluate whether students actually inherit the teacher's behavior rather than simply

achieving similar scores, we introduce **FidelityBench-Med**, an evaluation suite targeting teacher–student behavioral alignment and evidence-grounded faithfulness, helping to detect hallucinations and measure citation reliability.

Finally, we perform targeted ablation studies—varying dataset size, temperature, α-mixing ratios, and LoRA rank—to understand which components most influence distillation quality under a fixed and realistic compute budget.

Our primary research question is: *Among the ten methods defined in our framework, which distillation approaches are most effective for transferring medical expertise from MedItron-7B to TinyLlama-1.1B, and how do the two methods we can realistically run compare?*

Our secondary research questions include:

- What trade-offs exist between accuracy, evidence-grounded faithfulness, and computational efficiency in medical LLM distillation?
- When do chain-of-thought or RL-based distillation methods help or hinder small-model performance?
- Does improved token-level alignment (e.g., lower KL divergence) with the teacher actually translate into better reasoning and benchmark performance?

By addressing these questions, our work provides a structured, reproducible investigation of medical knowledge distillation under constrained compute settings, helping clarify which distillation strategies matter most for producing practical, reliable small medical LLMs.

2 Background Knowledge

2.1 Method 1: Standard Supervised Fine-Tuning (SFT)

Standard Supervised Fine-Tuning (SFT) is the most fundamental method used in knowledge distillation for language models. In SFT, the teacher model generates reference outputs, and the student model is trained to reproduce these outputs using standard token-level cross-entropy loss. Unlike probabilistic distillation methods that attempt to match the teacher's full output distribution, SFT operates solely on the teacher's chosen response tokens, thereby learning *hard labels* rather than distributional information.

In our setup, SFT is conducted in an online fashion: for each minibatch, the teacher model generates responses dynamically, allowing the teacher to adapt to the exact prompts and sampling configuration used during training. The student model is then optimized to minimize the standard autoregressive language modeling loss:

$$\mathcal{L}_{\text{SFT}} = -\sum_{t=1}^{T} \log p_{\theta_s}(y_t \mid y_{<t}, x),$$

where p_{θ_s} denotes the student's next-token distribution and y_t are the teacher-generated tokens. Because SFT does not match the teacher's soft probabilities, it may lose information about uncertainty or alternate reasoning paths, but it remains a simple, stable, and widely used baseline for distillation.

Reference: Standard supervised language modeling practices; see the HuggingFace documentation: `https://huggingface.co/docs/transformers/en/tasks/language_modeling`.

2.2 Method 2: Logit KD

Logit Knowledge Distillation extends standard SFT by transferring the teacher model's full probability distribution rather than relying solely on its selected output tokens. Following the framework of Hinton et al. [1], the teacher produces softened probability distributions via temperature scaling, which reveal relative token confidences and uncertainty patterns that are not captured by hard labels.

Given teacher logits $z^{(t)}$ and a temperature parameter $T > 1$, the softened distribution is computed as

$$p_{\theta_t}(y_t \mid y_{<t}, x; T) = \mathrm{softmax}\left(\frac{z^{(t)}}{T}\right).$$

The student is trained to match these distributions using the KL divergence:

$$\mathcal{L}_{\mathrm{KD}} = T^2 \,\mathrm{KL}(p_{\theta_t}(\cdot; T) \parallel p_{\theta_s}(\cdot; T)),$$

where the T^2 factor preserves gradient scaling.

To encourage both correctness and distributional alignment, Logit KD combines the KL term with the standard cross-entropy loss:

$$\mathcal{L} = (1 - \alpha)\,\mathcal{L}_{\mathrm{CE}} \;+\; \alpha\,\mathcal{L}_{\mathrm{KD}},$$

where $\alpha \in [0, 1]$ determines the weighting between hard and soft targets. Training is performed in online mode: the teacher generates sequences and logits at each step, which are immediately used to compute the distillation loss.

Reference: Hinton, G., Vinyals, O., and Dean, J. (2015). *Distilling the Knowledge in a Neural Network*. `https://arxiv.org/abs/1503.02531`.

2.3 Method 3: Token-Adaptive KD

Token-Adaptive Knowledge Distillation extends Logit KD by introducing a token-specific temperature that adapts to the student's confidence at each generation step. Instead of applying a single global temperature to soften the teacher's logits, the method assigns a dynamic temperature T_t for each token position. The intuition is that low-confidence predictions should receive softer teacher distributions (higher temperatures), while confident predictions should use sharper distributions (lower temperatures).

Student confidence at token t is estimated using the maximum predicted probability:

$$c_t = \max_v \, p_{\theta_s}(v \mid y_{<t}, x).$$

This value is converted into a per-token temperature via a clipped inverse mapping:

$$T_t = \mathrm{clip}\left(T_{\mathrm{base}} \cdot \frac{1}{c_t + \epsilon},\; T_{\mathrm{min}},\; T_{\mathrm{max}}\right),$$

where T_{base} is a base temperature and $(T_{\min}, T_{\max})$ define permissible bounds.

The distillation loss is computed by applying KL divergence with the adaptive temperatures:

$$\mathcal{L}_{\text{KD}} = \sum_{t=1}^{T} T_t^2 \, \text{KL}(p_{\theta_t}(\cdot; T_t) \parallel p_{\theta_s}(\cdot; T_t)) \, .$$

The final training objective combines this adaptive KL term with the standard cross-entropy loss:

$$\mathcal{L} = (1 - \alpha) \, \mathcal{L}_{\text{CE}} \; + \; \alpha \, \mathcal{L}_{\text{KD}},$$

where α determines the weighting between correctness and adaptive distribution matching.

By focusing the teacher's guidance on low-confidence positions, Token-Adapt KD can improve distillation effectiveness, especially when transferring knowledge from a much larger teacher (MedItron-7B) to a compact student model (TinyLlama-1.1B).

Reference: Li, X., et al. (2025). *Token-Level Adaptive Knowledge Distillation for Large Language Models.*

2.4 Method 4: Chain-of-Thought KD (CoT KD)

Chain-of-Thought (CoT) Distillation transfers not only the teacher model's final answers but also its explicit step-by-step reasoning process. Instead of learning from short, direct responses, the student is supervised on detailed natural-language rationales that reveal how the teacher arrives at its conclusions. This exposes the structure of the teacher's internal reasoning, making CoT distillation a significantly richer form of supervision than standard SFT.

A CoT example typically consists of (i) the problem statement, (ii) a multi-step reasoning chain, and (iii) the final answer. The rationale may include explicit deductions, biomedical justification, intermediate calculations, or citations. To ensure consistency, CoT pipelines often enforce formatting constraints such as fixed reasoning prompts, separators between rationale and answer, and explicit markers indicating the end of the explanation.

The student is trained to reproduce the entire reasoning sequence using the standard autoregressive cross-entropy loss:

$$\mathcal{L}_{\text{CoT}} = -\sum_{t=1}^{T} \log p_{\theta_s}(y_t \mid y_{<t}, x),$$

where y_t includes both rationale tokens and the final answer token. Because the loss is applied uniformly over the full explanation, the student is encouraged to internalize multi-hop reasoning patterns and to emulate the teacher's inferential structure rather than only its ultimate prediction.

Several design choices substantially influence the effectiveness of CoT distillation. First, rationale length must be controlled: overly verbose explanations can overwhelm small student models, while concise rationales may fail to convey sufficient reasoning signal. Second, explicit separation of rationales and answers is crucial to prevent the student from merging the two or defaulting to answer-only behavior. Third, sampling multiple diverse rationales can

enrich supervision but risks introducing contradictory reasoning paths that confuse compact models. Finally, because teacher-generated explanations may be noisy or partially hallucinated, many CoT distillation pipelines incorporate filtering or scoring mechanisms to ensure that only high-quality rationales are used for training.

While CoT distillation has demonstrated improvements in small-model reasoning ability, it also introduces challenges such as exposure bias, increased sequence length, and imitation of superficial reasoning patterns. Thus, its effectiveness depends on carefully balancing rationale quality, diversity, and length.

References: Ho, N., Schmid, L., and Yun, S. (2023). *Large Language Models Are Reasoning Teachers*. `https://arxiv.org/abs/2212.10071`. Wei, J., Wang, X., Schuurmans, D., et al. (2022). *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models*. `https://arxiv.org/abs/2201.11903`.

2.5 Method 5: Intermediate Feature Matching (FitNets-Style)

Intermediate Feature Matching extends output-level distillation by also transferring the teacher model's internal representations. Inspired by FitNets [2], the student is encouraged to align its hidden states with those of the teacher at selected intermediate layers. The intuition is that the teacher's intermediate features encode useful abstractions (e.g., domain knowledge and reasoning structure) that a small student model may not discover from output supervision alone.

Given a mapping from student layers to teacher layers, we extract token-level hidden states from both models while they process the same input (and, in our case, the same teacher-generated output sequence). Because the hidden dimensions may differ, the student features are optionally passed through a learnable projection layer before matching. The feature distillation loss is a mean squared error (MSE) between projected student states and teacher states:

$$\mathcal{L}_{\text{feat}} = \sum_{(l_s,l_t)\in\mathcal{M}} \sum_{t=1}^{T} \left\| P_{l_s}\big(h_{l_s,t}^{(s)}\big) - h_{l_t,t}^{(t)} \right\|_2^2,$$

where $\mathcal{M}$ denotes the layer mapping, $h_{l_s,t}^{(s)}$ and $h_{l_t,t}^{(t)}$ are student and teacher hidden states at token t, and P_{l_s} is an optional projection, used to fit the dimension, and helps the student restructure its representation space to be closer to the teacher's.

The overall training objective combines this feature-matching loss with the standard cross-entropy loss on the teacher-generated tokens:

$$\mathcal{L} = (1-\alpha)\,\mathcal{L}_{\text{CE}} \;+\; \alpha\,\mathcal{L}_{\text{feat}},$$

where $\alpha \in [0,1]$ controls the strength of intermediate representation alignment. Training is performed in online mode, with the teacher providing both outputs and hidden states during distillation.

Reference: Romero, A., et al. (2014). *FitNets: Hints for Thin Deep Nets*. `https://arxiv.org/abs/1412.6550`.

2.6 Method 6: Attention Distillation

Attention Distillation transfers the teacher model's self-attention patterns to the student, encouraging the student to focus on similar parts of the input sequence. In a transformer, each layer produces attention maps of shape (H, L, L), where H is the number of heads and L is the sequence length. These maps describe how tokens attend to one another and encode important structural information about the model's reasoning.

Given a mapping between student and teacher layers, we extract the corresponding attention weight matrices for the same input (and typically the same teacher-generated output sequence). For a set of heads $\mathcal{H}$, the attention transfer loss is defined as a mean squared error between teacher and student attention maps:

$$\mathcal{L}_{\text{attn}} = \sum_{(l_s, l_t) \in \mathcal{M}} \sum_{h \in \mathcal{H}} \left\| A^{(s)}_{l_s,h} - A^{(t)}_{l_t,h} \right\|_2^2,$$

where $A^{(s)}_{l_s,h}$ and $A^{(t)}_{l_t,h}$ denote the student and teacher attention distributions at the specified layers and heads.

The overall objective combines this attention-matching term with the standard cross-entropy loss on teacher-generated tokens:

$$\mathcal{L} = (1 - \alpha)\, \mathcal{L}_{\text{CE}} \; + \; \alpha\, \mathcal{L}_{\text{attn}},$$

where $\alpha \in [0, 1]$ controls the strength of attention transfer. By aligning attention patterns, the student learns not only to reproduce the teacher's outputs but also to adopt similar input focusing behavior.

Reference: Zagoruyko, S., and Komodakis, N. (2016). *Paying More Attention to Attention: Improving the Performance of CNNs via Attention Transfer*. `https://arxiv.org/abs/1612.03928`.

2.7 Method 7: On-Policy Distillation

On-Policy Distillation applies reinforcement learning to knowledge distillation by training the student model on its own sampled outputs rather than on teacher-generated sequences. During training, the student autoregressively generates tokens from its current policy, and the teacher provides a reward signal that evaluates the quality of each sampled action. Rewards may be based on teacher log-probabilities, token-level KL divergence, or sequence-level scores.

The student is optimized using the REINFORCE policy gradient estimator [4]:

$$\nabla_{\theta_s} J(\theta_s) = E[R_t \, \nabla_{\theta_s} \log \pi_{\theta_s}(a_t \mid s_t)]\,,$$

where a_t is the student-sampled token, s_t is the current sequence prefix, and R_t is the reward assigned by the teacher. An entropy regularization term encourages exploration:

$$\mathcal{L}_{\text{entropy}} = -\beta\, H(\pi_{\theta_s})\,,$$

leading to the overall objective:

$$\mathcal{L} = -\sum_t R_t \log \pi_{\theta_s}(a_t \mid s_t) \; - \; \beta \, H(\pi_{\theta_s}) \, .$$

Because the student learns from its own trajectories, on-policy distillation captures inference-time behavior more faithfully than offline methods, though it can be sensitive to reward design and gradient variance.

References: Agarwal et al. (2024), On-Policy Distillation. Williams, R. J. (1992). *Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning.*

2.8 Method 8: PPO-Based Distillation

PPO-based distillation extends on-policy distillation by replacing vanilla policy gradient with Proximal Policy Optimization (PPO) [3], which constrains policy updates for greater stability. As in the on-policy setting, the student autoregressively samples tokens from its current policy, and the teacher provides reward signals that evaluate the generated actions or sequences. From these rewards, advantage estimates A_t are computed using a value baseline.

Let $\pi_{\theta_{\text{old}}}$ denote the student policy that generated the data and π_θ the updated policy. PPO forms the probability ratio

$$r_t(\theta) = \frac{\pi_\theta(a_t \mid s_t)}{\pi_{\theta_{\text{old}}}(a_t \mid s_t)},$$

and optimizes the clipped surrogate objective

$$L^{\text{CLIP}}(\theta) = E_t\Big[\min\big(r_t(\theta)A_t, \; \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\, A_t\big)\Big],$$

where ϵ is the clipping parameter. This prevents destructively large updates when $r_t(\theta)$ deviates too far from 1.

In addition, a value function $V_\phi(s_t)$ is trained with a regression loss

$$\mathcal{L}_{\text{value}} = \frac{1}{2}\big(V_\phi(s_t) - R_t\big)^2,$$

and an entropy bonus encourages exploration:

$$\mathcal{L}_{\text{entropy}} = -H(\pi_\theta(\cdot \mid s_t)) \, .$$

The overall loss minimized during PPO distillation is

$$\mathcal{L} = -L^{\text{CLIP}}(\theta) + c_v \, \mathcal{L}_{\text{value}} - \beta \, \mathcal{L}_{\text{entropy}},$$

where c_v and β are the value and entropy coefficients. In our setting, the teacher acts as the reward source, and the student policy is updated to better align with the teacher's preferences while maintaining stable, proximal updates.

Reference: Schulman, J., et al. (2017). *Proximal Policy Optimization Algorithms.*

https://arxiv.org/abs/1707.06347.

2.9 Method 9: Best-of-N Distillation (BOND)

Best-of-N Distillation (BOND) aims to compress the benefits of best-of-N sampling into a single-pass student policy. For each input prompt, the student generates N diverse candidate responses $\{y^{(k)}\}_{k=1}^{N}$ via sampling, and the teacher scores or ranks these candidates according to a preference signal. Let s_k denote the teacher's score for sample k and $k^\star$ the index of the highest-scoring sample.

A simple supervised view of BOND treats the best sample $y^{(k^\star)}$ as a pseudo-label and optimizes the student to increase its likelihood:

$$\mathcal{L}_{\text{BOND-SFT}} = -\sum_{t} \log \pi_{\theta_s}\big(y_t^{(k^\star)} \mid y_{<t}^{(k^\star)}, x\big).$$

This distills the implicit reward of best-of-N selection into a single forward pass.

An alternative RL-style formulation assigns a reward r_k to each sample (e.g., $r_{k^\star} = 1$ for the best sample and $r_k = 0$ for the others, or a rank-based reward when using full rankings). The loss then becomes

$$\mathcal{L}_{\text{BOND-RL}} = -\sum_{k=1}^{N} r_k \sum_{t} \log \pi_{\theta_s}\big(y_t^{(k)} \mid y_{<t}^{(k)}, x\big),$$

which is equivalent to a REINFORCE-style policy gradient with teacher-derived rewards.

In both variants, BOND distills the behavior of a best-of-N selection process into a compact student model, enabling it to approximate the performance of multi-sample decoding with a single, efficient generation.

Reference: Yang et al. (2024). *BOND: Aligning LLMs with Best-of-N Distillation.*

2.10 Method 10: SPIN Distillation (Self-Play Preference Optimization)

SPIN adapts self-play and preference-based learning to knowledge distillation by treating teacher responses as preferred and the student's own responses as dispreferred. For each prompt x, the student generates a response y^-, and the teacher produces a higher-quality response y^+. This forms a preference pair (y^+, y^-) without requiring an explicit reward model.

SPIN applies a DPO-style objective to make the student more likely to generate the teacher-preferred response. The loss is

$$\mathcal{L}_{\text{SPIN}} = -\log \sigma\big(\beta\big[\log \pi_{\theta_s}(y^+ \mid x) - \log \pi_{\theta_s}(y^- \mid x)\big]\big)\,,$$

where σ is the logistic sigmoid and β controls the sharpness of the preference enforcement. This encourages the student to assign higher probability to teacher responses than to its own earlier ones.

Training proceeds iteratively: the student continually produces new dispreferred examples, and the teacher provides preferred alternatives. This self-play loop improves the student by contrasting its own responses with higher-quality teacher outputs while avoiding high-variance policy gradient methods and eliminating the need for an explicit reward model.

Reference: Chen et al. (2024a). *Self-Play Fine-Tuning Converts Weak Language Models to Strong.*

3 Theoretical Analysis: SFT vs. Logit Knowledge Distillation

Due to the time constraint and the limited 5090 computing resource we got accessed to (always face cuda OOM issue when doing planned 80k samples training), we finally only are able to do two experiment on the SFT and logit KD. Hence here we compare Supervised Fine-Tuning (SFT) and Logit Knowledge Distillation (Logit KD) from the perspective of what information each method transfers from the teacher to the student and how this affects optimization.

3.1 Key Theoretical Differences

Information content. SFT provides only hard labels, whereas Logit KD exposes the teacher's full predictive distribution. Thus Logit KD contains strictly more information, especially about uncertainty and token-level similarity.

Gradient behavior. SFT updates the student only using the sampled teacher token. Logit KD updates the student toward the entire teacher distribution, producing smoother gradients and potentially faster or more stable optimization.

Bias vs. variance. SFT acts as a high-bias estimator of the teacher distribution (because of one-hot targets). Logit KD is lower bias because it directly matches the teacher distribution; its gradients incorporate information from all plausible tokens.

Capacity considerations. For a small model such as TinyLlama-1.1B, Logit KD may help allocate limited capacity more efficiently by revealing which alternative tokens the teacher considers plausible. However, if the teacher is poorly calibrated or overly diffuse, SFT can be more robust.

3.2 Summary

Both SFT and Logit KD aim to make the student approximate the teacher, but they differ substantially in the signals they provide. SFT transfers only the final choices of the teacher, while Logit KD transfers the structure of the teacher's belief distribution. In practice, this means Logit KD can better guide a small student model when soft probabilities are informative, whereas SFT offers a simpler and more conservative form of distillation.

4 Experiment Setup

4.1 Dataset Construction

To provide a unified source of supervision for distillation, we merged four medical QA datasets—MedQA, MedMCQA, PubMedQA, and PubHealth—into a single pool. All items were reformatted into long-form, free-response questions. In particular, multiple-choice datasets (MedQA, MedMCQA) were converted into open-ended explanatory questions that require reasoning and justification. This ensures that all prompts follow a consistent structure and align with the teacher model's natural long-form outputs.

The resulting dataset, Med-DistillMix, supplies the prompts for online distillation. Benchmark datasets remain separate and are used exclusively for evaluation. FidelityBench-Med is used to measure behavioral alignment between teacher and student.

4.2 Teacher–Student Distillation Pipeline

We distill from the domain-specialized **MedItron-7B** teacher model into the compact **TinyLlama-1.1B** student model. All runs use an *online* distillation setup in which the teacher generates outputs (and logits for Logit KD) dynamically during training. This keeps teacher outputs consistent with the current sampling configuration and avoids storing large teacher-generated datasets.

Both models are trained and inferred using **QLoRA** with 4-bit quantization, enabling the full system to run on a single NVIDIA RTX 5090 GPU.

4.3 Training Configuration and Practical Constraints

Initial attempts to train on the full 80k-example dataset led to repeated out-of-memory failures due to the cost of long-sequence teacher generation. After experimentation, we found a stable configuration that enables both SFT and Logit KD to run without memory overflow:

- Training set: 64 examples
- Validation set: 4 examples
- Test set: 4 examples
- Batch size: 4
- Logit KD mixture weight: $\alpha = 0.5$

SFT uses teacher-generated tokens directly as supervision. Logit KD uses the teacher's logits to compute a KL-based soft-target loss blended with the SFT loss. All other hyperparameters and seeds are held constant across methods for a fair comparison.

4.4 Evaluation Logic

Because the training and evaluation sets are intentionally small to meet hardware constraints, our evaluation focuses not on statistical accuracy but on *behavioral and qualitative differences* between SFT and Logit KD. We use two complementary evaluation tracks: task performance and fidelity analysis.

(1) Task performance. We track:

- **Training and validation loss curves**, indicating the stability and convergence of each method,
- **Answer correctness** on the four held-out test examples,
- **Qualitative inspection** of generated explanations and reasoning.

Although small in size, these measurements reveal differences in optimization behavior and the types of outputs each method favors.

(2) Teacher–student fidelity analysis. Using FidelityBench-Med, we assess how closely the student model matches the teacher's distributional behavior and textual outputs. We compute:

- **Distribution divergence:** token-level KL divergence between teacher and student next-token distributions,
- **Top-k token overlap:** similarity of ranked candidate tokens,
- **Text generation similarity:** overlap and semantic similarity between full teacher and student generations,
- **Exact match rate:** fraction of outputs identical to the teacher's output.

These metrics measure not only response quality but also how faithfully each distillation method captures the teacher's underlying behaviors. Together, the accuracy-focused metrics and fidelity-focused metrics allow us to analyze the trade-offs between SFT and Logit KD under constrained computational settings.

5 Result Analysis

We compare SFT and Logit KD along three dimensions: (1) loss behavior during training, (2) similarity between student and teacher outputs, and (3) distribution-level alignment as measured by divergence metrics. Because our training dataset and evaluation split are intentionally small due to hardware constraints, the goal of this analysis is not statistical significance but a controlled comparison of the behaviors induced by the two distillation methods.

5.1 Loss Behavior

Across all runs, SFT achieves consistently *lower training and validation loss* than Logit KD. This is expected: SFT optimizes a single, well-behaved cross-entropy objective, whereas Logit KD optimizes a weighted mixture of cross-entropy and KL divergence. In our small-data regime, the additional KL component appears to introduce optimization difficulty without providing a clear benefit, leading to higher final loss values for Logit KD.

This suggests that, under severe data and compute limitations, a simpler loss function may yield more stable convergence and better apparent fit to the training data.

5.2 Textual Similarity and Exact Matching

SFT produces generations that are *more similar* to the teacher's textual outputs. This follows naturally from the fact that SFT directly trains on the teacher's generated tokens as hard labels; the student therefore learns to mimic the teacher's phrasing and structure more closely. In our evaluations, SFT obtained higher text-generation similarity scores.

Interestingly, Logit KD achieves a *higher exact match rate* on certain examples. Exact match is a stricter metric, and in our small-scale setting, Logit KD's softened training signal can sometimes push the student toward the teacher's most probable output sequence more directly. However, on average, SFT still produces outputs that are more textually aligned with the teacher.

5.3 Distributional Divergence

Despite producing outputs that are textually more similar, SFT shows *higher KL divergence and JS divergence* when we compare token-level distributions between student and teacher. This indicates that SFT matches the teacher's *chosen tokens* but does not match the teacher's underlying probability distribution.

By contrast, Logit KD produces *lower* KL and JS divergence. Because it explicitly matches the teacher's softened probability distribution, it achieves closer alignment at the token distribution level—even when its generated text is less similar overall. This illustrates that surface-level text similarity and distributional similarity measure different aspects of teacher–student alignment.

5.4 Summary and Implications

In our constrained setting, SFT and Logit KD exhibit complementary behaviors:

- **SFT** produces text that is more similar to the teacher's outputs and achieves lower loss, but exhibits higher distributional divergence.
- **Logit KD** achieves lower KL/JS divergence and higher exact match rates, but produces text that is overall less similar and converges to higher loss values.

These observations reinforce a key insight: *a more complex loss function does not necessarily yield better performance when data and compute are extremely limited.* In such a

regime, SFT may benefit from its simplicity and direct supervision, while Logit KD's advantages in distributional matching are less able to manifest.

Overall, our results highlight the importance of considering both textual outputs and distribution-level fidelity when evaluating distillation methods, especially when working with small student models and highly restricted compute.

6 Contribution

- Tim: trainer script, debug, training, result analysis.
- Federico: Train method class, debug, presentation slides, coordinator.
- Shreyan: pipeline design, experiment design, train method class (tremendously big file), debug, training.
- Bryan: experiment design, debug, training, compute resource setup, dataloader script, final report.

References

[1] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[2] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, et al. Fitnets: Hints for thin deep nets. In *International Conference on Learning Representations (ICLR)*, 2015. Originally published 2014 as arXiv:1412.6550.

[3] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[4] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256, 1992.