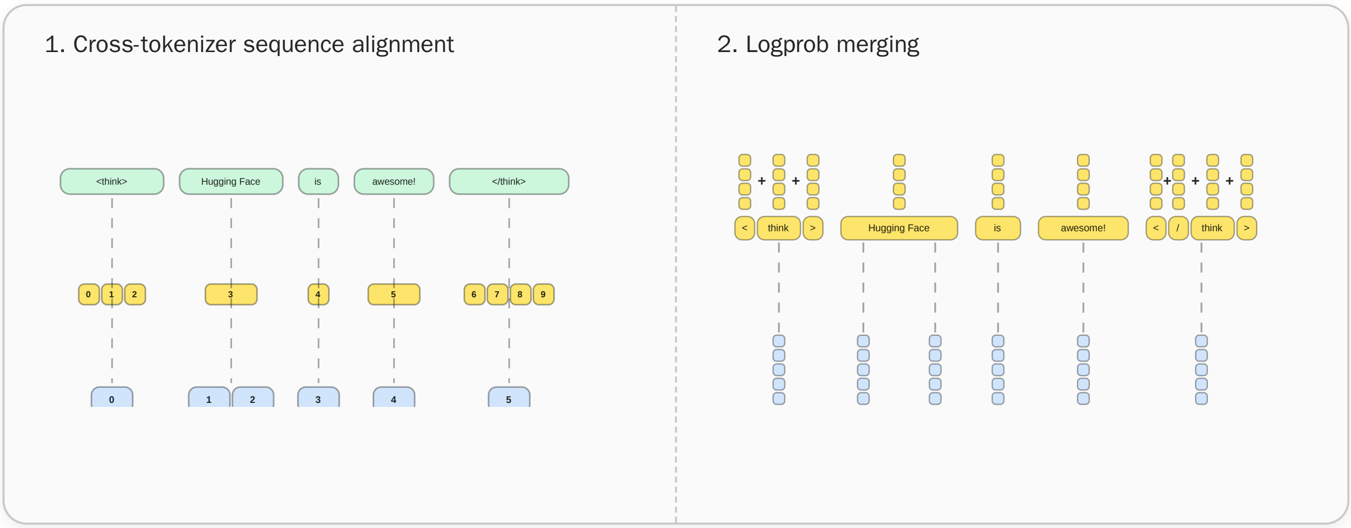
Unlocking On-Policy Distillation for Any Model Family



Apply on-policy distillation to models from different families

AUTHORS

Carlos Miguel Patiño, Kashif Rasul, Quentin Gallouédec, Ben Burtenshaw, Sergio Paniego, Vaibhav Srivastav, Thibaud Frere, Ed Beeching, Lewis Tunstall, Leandro von Werra, Thomas Wolf

AFFILIATION

Hugging Face

PUBLISHED

Oct. 29, 2025

Introduction

On-policy distillation is a highly effective strategy for compressing LLMs, as recently highlighted by Thinking Machines' excellent blog post. The technique trains a small "student" model by transferring knowledge from a high-performing "teacher" model's probability distribution. This allows the student to emulate the teacher's task performance, while significantly reducing size and latency.

In this blog post, we introduce General On-Policy Logit Distillation (GOLD), our method for extending on-policy distillation to address a fundamental weakness: the requirement that the teacher and student models must share the *same* tokenizer vocabulary.

Building on Universal Logit Distillation (ULD) (Boizard et al., 2025), GOLD is highly effective for complex, multi-step reasoning tasks, such as math. Our results show GOLD performs better than ULD and even GRPO.

Our key contributions are:

- Providing an open-source implementation of on-policy distillation methods in TRL (GKD and GOLD) and proving they work for multiple model combinations.
- Extending ULD to the on-policy setting, where we sample completions from the student and align them to the teacher's distribution.
- Implementing new sequence and vocabulary alignment methods that improve distillation performance when the student and the teacher have different tokenizers.

With this foundation in place, let's step back to review the broader landscape of knowledge distillation methods - how on-policy approaches emerged, and why extending them beyond shared tokenizers is critical.

Distillation Methods

Off-policy vs. on-policy distillation

There are two main type of distillation: off-policy and on-policy. Off-policy distillation trains a student model on fixed data (typically the teacher's precomputed logits or text completions), while on-policy distillation involves the teacher providing feedback to the student's own outputs.

Generalised Knowledge Distillation (GKD) (Agarwal et al., 2024) unifies these approaches under a common framework by supporting a range of loss functions that enable training on both static teacher data and trajectories generated by the student. The GKD paper shows that on-policy distillation typically outperforms off-policy methods: a result we confirm later in this post.

On-policy distillation's advantage is twofold. First, as the student model improves, its generations create progressively higher-quality training data, forming a positive feedback loop. Second, this "context alignment" forces the student to learn from the same types of errors and successes it will encounter during inference, rather than from completions generated only by the teacher.

GKD controls this on-policy vs. off-policy data mixture via the λ parameter, where $\lambda = 1$ is fully on-policy and $\lambda = 0$ is fully offline as shown in the equation below

$$\mathcal{L}_{GKD} = (1 - \lambda)\mathcal{L}_{SD} + \lambda\mathcal{L}_{OD}$$

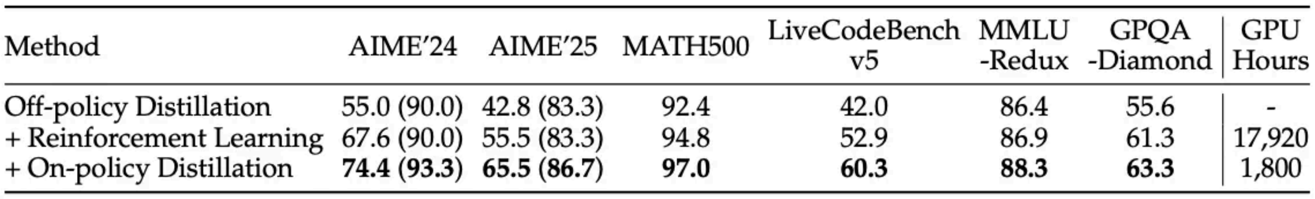
where $\mathcal{L}_{SD}$ is the supervised distillation (SD) that leverages off-policy generations from the teacher and $\mathcal{L}_{OD}$ is the on-policy distillation (OD) using student generations and feedback from the teacher's logits [1].

When compared to RL, GKD also has two main benefits:

1. We don't need to rely on a reward function that gives sparse feedback
2. The method works for small models which initially have low performance in the task we're trying to optimise for.

The reward function requires either a verifiable task or training a reward model to score the completion and only gives feedback about the outcome. There is no explicit information about which part of the process were correct and which require adjustments.

On-policy distillation overcomes this limitation by providing feedback from a strong teacher at the *token level*. This approach is especially effective for smaller models, as demonstrated in the Qwen3 (Yang et al., 2025) results below, where on-policy distillation outperforms RL at a fraction of the compute budget:

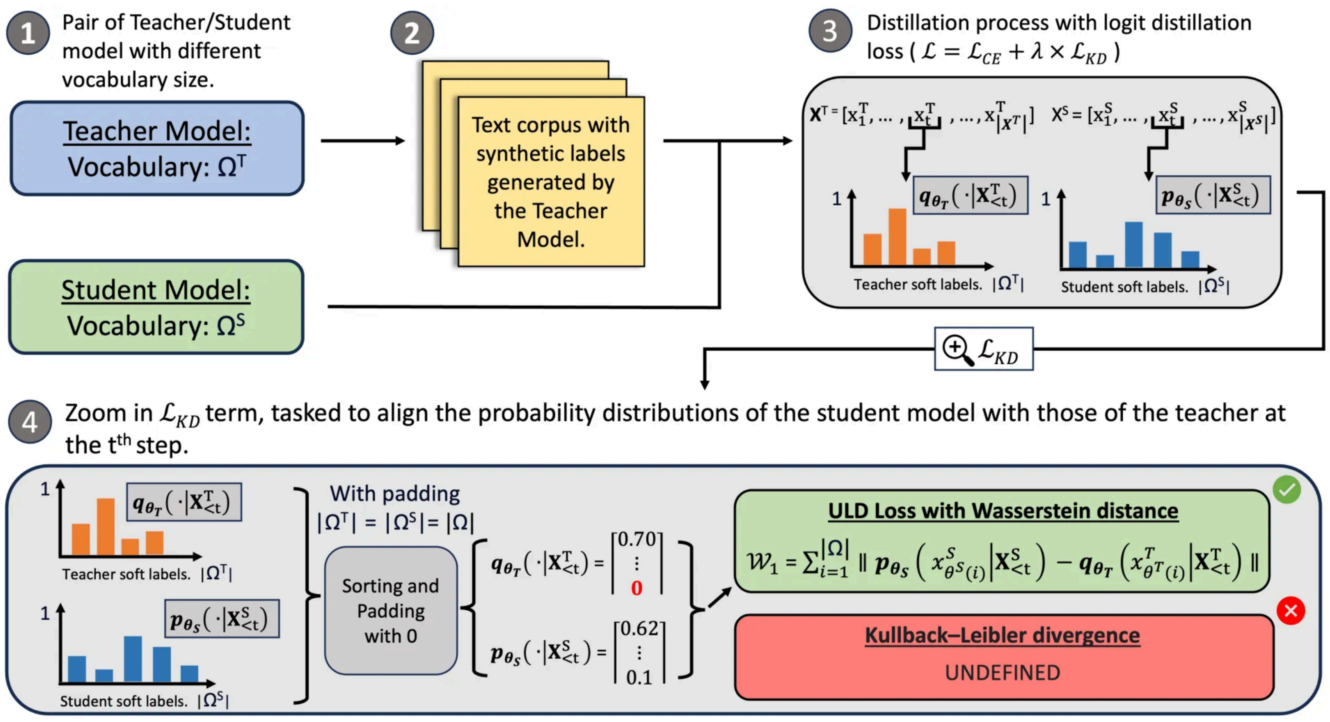


Method	AIME'24	AIME'25	MATH500	LiveCodeBench v5	MMLU -Redux	GPQA -Diamond	GPU Hours
Off-policy Distillation	55.0 (90.0)	42.8 (83.3)	92.4	42.0	86.4	55.6	-
+ Reinforcement Learning	67.6 (90.0)	55.5 (83.3)	94.8	52.9	86.9	61.3	17,920
+ On-policy Distillation	**74.4 (93.3)**	**65.5 (86.7)**	**97.0**	**60.3**	**88.3**	**63.3**	1,800

While GKD establishes a strong foundation for on-policy training, it assumes both models share a tokenizer, a practical constraint we'll now address through Universal Logit Distillation (ULD).

Universal logit distillation

The main limitation with all on-policy distillation methods is that they assume the use of the same tokenizer for both the student and the teacher. The current AI ecosystem spans different model families such as SmolLM, Llama, Qwen, and Gemma, each with their own strengths and shortcomings. Each model family, and even different versions within the same family, uses its own tokenizer, so requiring a single tokenizer can be overly restrictive when selecting student-teacher pairings. Recent work, such as Universal Logit Distillation (ULD), lifts the tokenizer restriction by showing distillation can be performed without needing a perfect alignment between teacher and student vocabularies, albeit in an offline setting.



1 Pair of Teacher/Student model with different vocabulary size.
Teacher Model: Vocabulary: Ω^T
Student Model: Vocabulary: Ω^S
2
Text corpus with synthetic labels generated by the Teacher Model.
3 Distillation process with logit distillation loss ($\mathcal{L} = \mathcal{L}_{CE} + \lambda \times \mathcal{L}_{KD}$)
$X^T = [x_1^T, \dots, x_t^T, \dots, x_{|x^T|}^T]$
$X^S = [x_1^S, \dots, x_t^S, \dots, x_{|x^S|}^S]$
$q_{\theta_T}(\cdot|X_{<t}^T)$
$p_{\theta_S}(\cdot|X_{<t}^S)$
Teacher soft labels. $|\Omega^T|$
Student soft labels. $|\Omega^S|$
$\mathcal{L}_{KD}$
4 Zoom in $\mathcal{L}_{KD}$ term, tasked to align the probability distributions of the student model with those of the teacher at the tth step.
With padding $|\Omega^T| = |\Omega^S| = |\Omega|$
Sorting and Padding with 0
$q_{\theta_T}(\cdot|X_{<t}^T) = \begin{bmatrix} 0.70 \\ \vdots \\ 0 \end{bmatrix}$
$p_{\theta_S}(\cdot|X_{<t}^S) = \begin{bmatrix} 0.62 \\ \vdots \\ 0.1 \end{bmatrix}$
ULD Loss with Wasserstein distance
$\mathcal{W}_1 = \sum_{i=1}^{|\Omega|} \| p_{\theta_S}\left(x_{\theta^S(i)}^S \middle| X_{<t}^S\right) - q_{\theta_T}\left(x_{\theta^T(i)}^T \middle| X_{<t}^T\right) \|$
Kullback–Leibler divergence
UNDEFINED

Figure 1: Previous work, ULD by Boizard et al. demonstrates offline distillation on student and teacher models with unmatched tokenizers. GOLD extends their method to the on-policy setting and addresses two weaknesses: token alignment in step 3 and logit alignment in step 4.

ULD showed that using distillation between models with different tokenizers introduces two key challenges:

1. Sequence misalignment: tokenizers split text differently. As shown in Figure 2, Tokenizer A might create a single “Hugging Face” token, while Tokenizer B creates two separate tokens.
2. Vocabulary misalignment: the same token string receives different IDs. In Figure 1, “awesome!” is ID=2 in Tokenizer A but ID=0 in Tokenizer B.

As shown in the figure below, this token ID mismatch results in different token sequences for the exact same text, where “Hugging Face is awesome!” corresponds to [3, 1, 2] for Tokenizer A and [2, 3, 1, 0] for Tokenizer B. ULD handles these issues by truncating sequences to the minimum length and by sorting and padding the smaller softmax vector to align vocabularies.

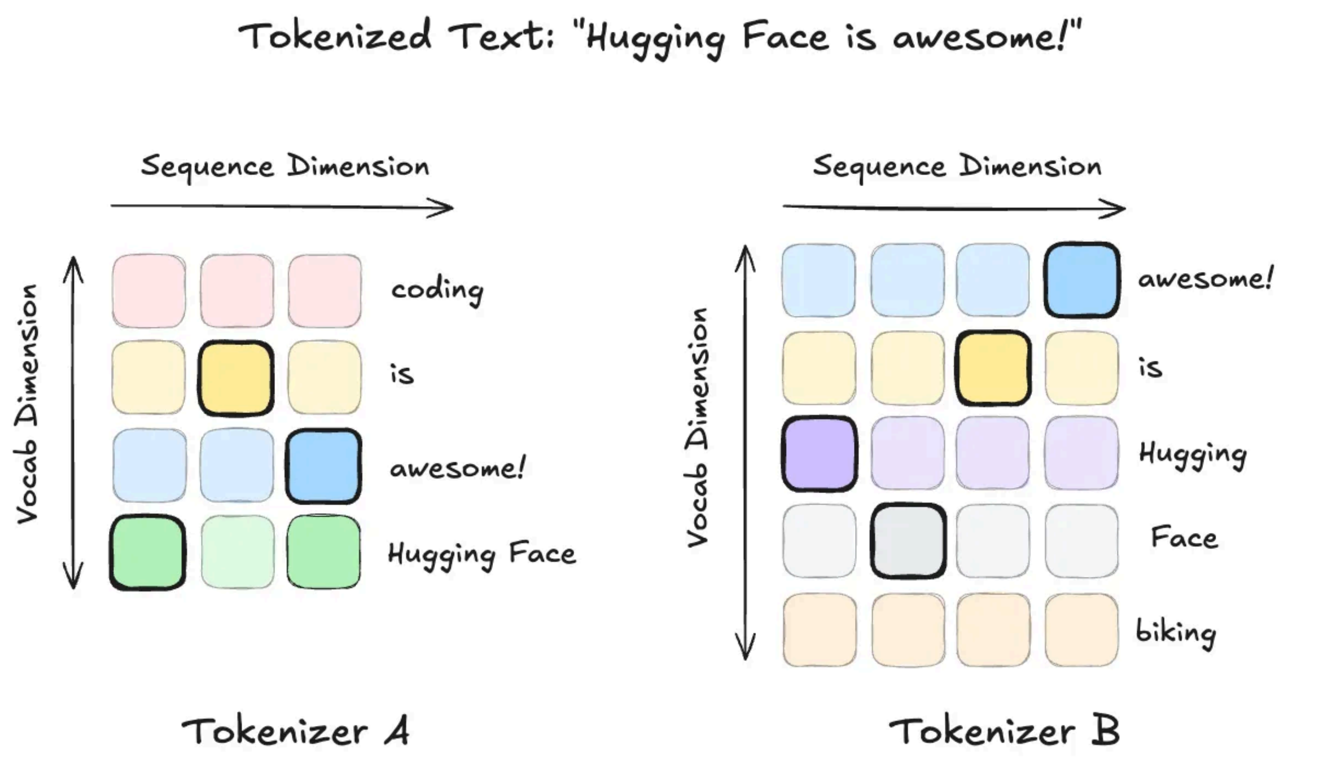


Figure 2: Diagram of sequence and vocabulary misalignments caused by differences between two tokenizers. Tokenizer A has fewer elements in its vocabulary and different token IDs when compared to tokenizer B. The differences cause the same text ("Hugging Face is awesome!") to be represented by token ID sequences with different lengths and elements.

ULD lifts the tokenizer restriction but remains limited to offline setups. Next, we introduce our core contribution, General On-Policy Logit Distillation (GOLD), which extends ULD into the on-policy setting with improved alignment techniques.

General On-Policy Logit Distillation (GOLD)

While Universal Logit Distillation (ULD) allows training models with different tokenizers, its methods for sequence and vocabulary alignment have limitations. We developed General On-Policy Logit Distillation (GOLD), an algorithm that extends ULD by introducing improved vocabulary alignment techniques.

Sequence Alignment

The first limitation we address is ULD’s sequence alignment, which simply truncates sequences to the minimum tokenized length. This simple approach causes two problems:

1. It leads to information loss at the end of the text.
2. It can misalign tokens, causing the distillation of tokens with different semantic meanings at the same sequence index.

This alignment error worsens as tokenization differences increase because a single mismatch at the start of a sequence can propagate and create a cascading semantic error throughout the text.

Instead of truncating, our method identifies the token merges required to equalise the sequence lengths for both tokenizers. We then merge the probabilities at the corresponding positions by multiplying the marginal distribution by scalar conditional probabilities of the actual continuation tokens.

We perform the token merge through scalar multiplication to leverage the autoregressive nature of LLM sampling. Following the example in Figure 3, we want to merge “Hugging” and ” Face” into one token for the sequence in blue. Using the conditional probabilities and the product rule [2], we can merge the probabilities and guarantee sequence alignment regardless of tokenizer discrepancies in the sequence dimension.

Figure 3: Diagram highlighting the differences between ULD and GOLD in the sequence alignment step. Instead of truncating the sequence at the minimum sequence length, we first determine the merges that result in an aligned sequence length between the two tokenizer. We then calculate the sum of the logprobs for the merged token position to get a unified vector with the token distribution for that position in the sequence.

Having resolved sequence mismatches through token merging, we now turn to vocabulary alignment, ensuring logits are comparable even when token IDs differ.

Vocabulary Alignment

Our second extension improves the alignment in the vocabulary dimension by replacing the sorting operation with an operation that leverages a potential one-to-one mapping between the tokenizers. ULD assumes that we cannot map any token between tokenizers, so it performs a sorting operation in the softmax dimension after padding the logits to have the same size. The assumption behind this process is that the softmax distribution is the same, or at least similar, under a different permutation of token IDs.

We find this assumption to be reasonable, but we can exploit tokens present in both vocabularies with a different ID to avoid relying on sorting when there's a direct mapping. For example, we know that "awesome!" is present in both vocabularies in Figure 4, but the token IDs differ. In GOLD's approach, we find those mappings where the token exists in both vocabularies and apply the GKD loss that assumes the same tokenizer. We fall back to the sorting process from ULD for the items in the vocabulary without a perfect match, so that we still consider those unmatched tokens during learning. GOLD's loss is then the result of adding $\mathcal{L}_{GKD}$ from the tokens with one-to-one mappings and $\mathcal{L}_{ULD}$ without a mapping. We allow defining the weights for each term our TRL implementation but include a default that worked well in our experiments.

$$\mathcal{L}_{GOLD}(x, y) = w_1 \mathcal{L}_{GKD} + w_2 \mathcal{L}_{ULD}$$

Figure 4: Diagram highlighting the differences between ULD and GOLD for the vocabulary alignment. GOLD tries to find 1:1 mapping between tokens in both tokenizers and applies the KL divergence loss from the GKD method. We fallback to the ULD process for tokens without a 1:1 mapping. The final loss is a sum of the two terms.

With GOLD's design clarified, we'll now examine how we evaluated it in practice, detailing our experimental setup, tasks, and models.

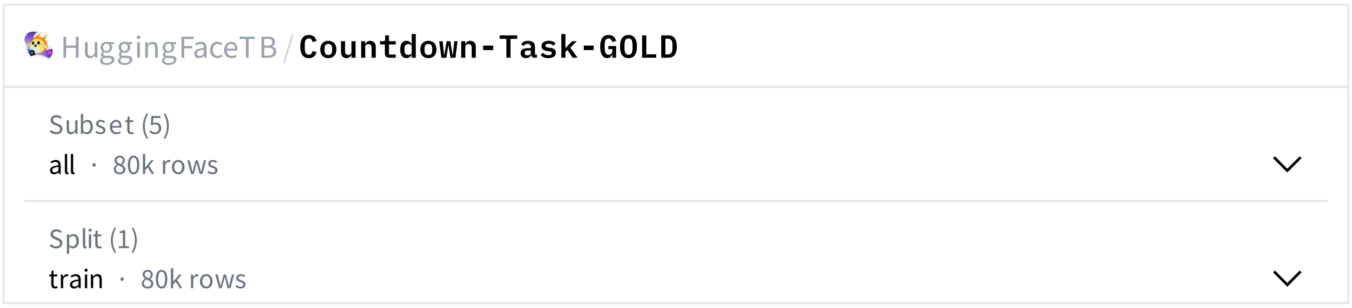
Experimental Setup

Task Definition

We used a math game called Countdown (Gandhi et al., 2024), where the objective is to reach a target value using a group of numbers and four arithmetic operations (+, -, *, /). Additionally, the model must provide the answer using a specific format because we set a strict parser that considers the answer wrong if it can't find the expected format. We only consider the answer as correct if it fulfils all the following conditions:

- Only uses each number once.
- The equation given by the model results in the target.
- The answer is an equation enclosed in the `<answer> </answer>` tags.

Below is an example of the system and user prompts we pass to the model for the task.



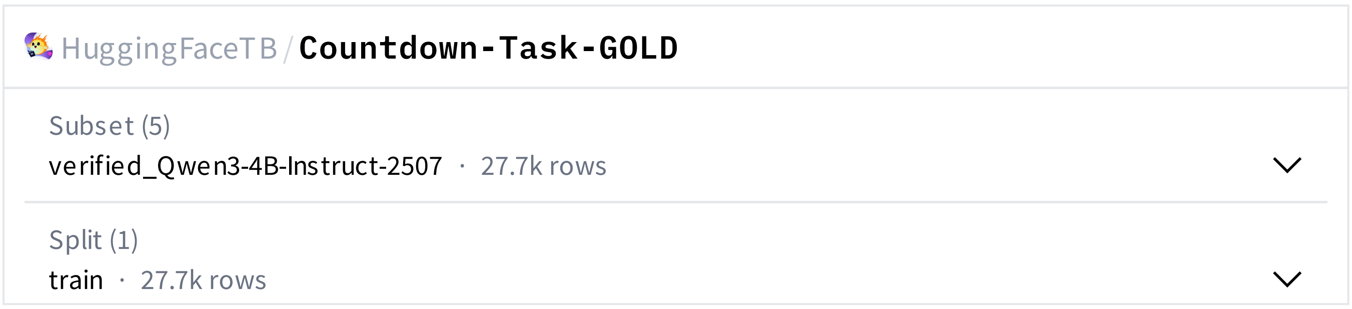
HuggingFaceTB / **Countdown-Task-GOLD**

Subset (5)
all · 80k rows

Split (1)
train · 80k rows

Dataset

We sourced all the prompts from the Jiayi-Pan/Countdown-Tasks-3to4 dataset. Our full dataset contains 80k training prompts and 10k testing prompts selected randomly. We then generated responses from
the `Qwen/Qwen2.5-7B-Instruct` and `Qwen/Qwen3-4B-Instruct-2507` teacher models, including only the prompts that had the correct answers from the teachers. Our published dataset contains 30.4k prompts for `Qwen/Qwen2.5-7B-Instruct` and 27.7k for `Qwen/Qwen3-4B-Instruct-2507` generations. We use the prompts in the training dataset with 30.4k prompts for all the on-policy experiments because we use the student's generations instead of the teacher's completions.


HuggingFaceTB / **Countdown-Task-GOLD**

Subset (5)
verified_Qwen3-4B-Instruct-2507 · 27.7k rows

Split (1)
train · 27.7k rows

Models Used

To test the effects of model size, performance, and token similarity on KD, we established several student-teacher setups. The teachers were all Qwen models of varying sizes, while the students were from three different families: Qwen, Llama, and Gemma. This created a significant performance gap for distillation: all student models had a baseline Countdown score below 0.08, whereas the teachers' scores ranged from 0.35 to 0.76.

Model Type	Model ID	Countdown Score
Student	meta-llama/Llama-3.2-1B-Instruct	0.016
Student	Qwen/Qwen2.5-1.5B-Instruct	0.076
Student	google/gemma-3-1b-it	0.023
Teacher	Qwen/Qwen2.5-7B-Instruct	0.3555
Teacher	Qwen/Qwen3-4B-Instruct-2507	0.7145

Tokenizer Similarity

We hypothesized that GOLD's performance would correlate with vocabulary similarity. To quantify this, we defined a tokenizer similarity metric using the Jaccard index (Intersection over Union, or IoU). In this context, the "intersection" is the count of tokens that can be matched between the two vocabularies, while the "union" is the total count of unique tokens across both.

Tables 1 and 2 below show the difference in tokenizer similarity when we enforce the same token IDs (first table) compared to when we match different token IDs when they correspond to the same token (second table).
The `meta-llama/Llama-3.2-1B-Instruct` and `google/gemma-3-1b-it` tokenizers have 0 similarity with all the teachers in the first case, but we increase it to 0.64 and 0.063 in the second case, respectively.

The tables also show that the tokenizer between Qwen2.5 and Qwen3 versions differs by only a few tokens. In fact, the only difference between the two tokenizers is that Qwen3 is the same tokenizer as Qwen2.5 with four additional tokens `('<think>', '<tool_response>', '</tool_response>', '</think>')`. Since the tokenizer for Qwen3 fully contains the tokenizer from Qwen2.5, we can treat the two tokenizers as equivalent for our experiments.

Table 1: Strict Matching

Student Model	Qwen/Qwen2.5-7B-Instruct	Qwen/Qwen2.5-32B-Instruct	Qw
meta-llama/Llama-3.2-1B-Instruct	0	0	0
google/gemma-3-1b-it	0	0	0
Qwen/Qwen2.5-1.5B-Instruct	1.0	1.0	0.9

Table 2: Token Mapping

Student Model	Qwen/Qwen2.5-7B-Instruct	Qwen/Qwen2.5-32B-Instruct	Qw
meta-llama/Llama-3.2-1B-Instruct	0.64	0.64	0.6
google/gemma-3-1b-it	0.063	0.063	0.0
Qwen/Qwen2.5-1.5B-Instruct	1.0	1.0	0.9

Experiments

GKD with the Same Tokenizer

Our first goal was to validate our GKD implementation by comparing our results with those reported by Agarwal et al. (Agarwal et al., 2024). We focused on comparing the performance of combining on-policy and off-policy learning through ablations of five different λ values, as shown in Figure 5. We used `Qwen/Qwen3-4B-Instruct-2507` as a teacher and `Qwen/Qwen2.5-1.5B-Instruct` as a student. For the offline learning, we generated completions to the prompts using `Qwen/Qwen3-4B-Instruct-2507` beforehand to speed up the training process. We set the temperature $\gamma = 1$ for the student generations and used the forward KL divergence $(\beta = 0)$[3] in $\mathcal{L}_{OD}$.

The results confirm that using at least some degree of on-policy training outperforms the SFT setup. We also see a trend of better performance as we increase λ, with fully on-policy achieving the best overall performance. This behavior confirms the hypothesis that fully on-policy training is better than training with offline data when using models with the same tokenizer.

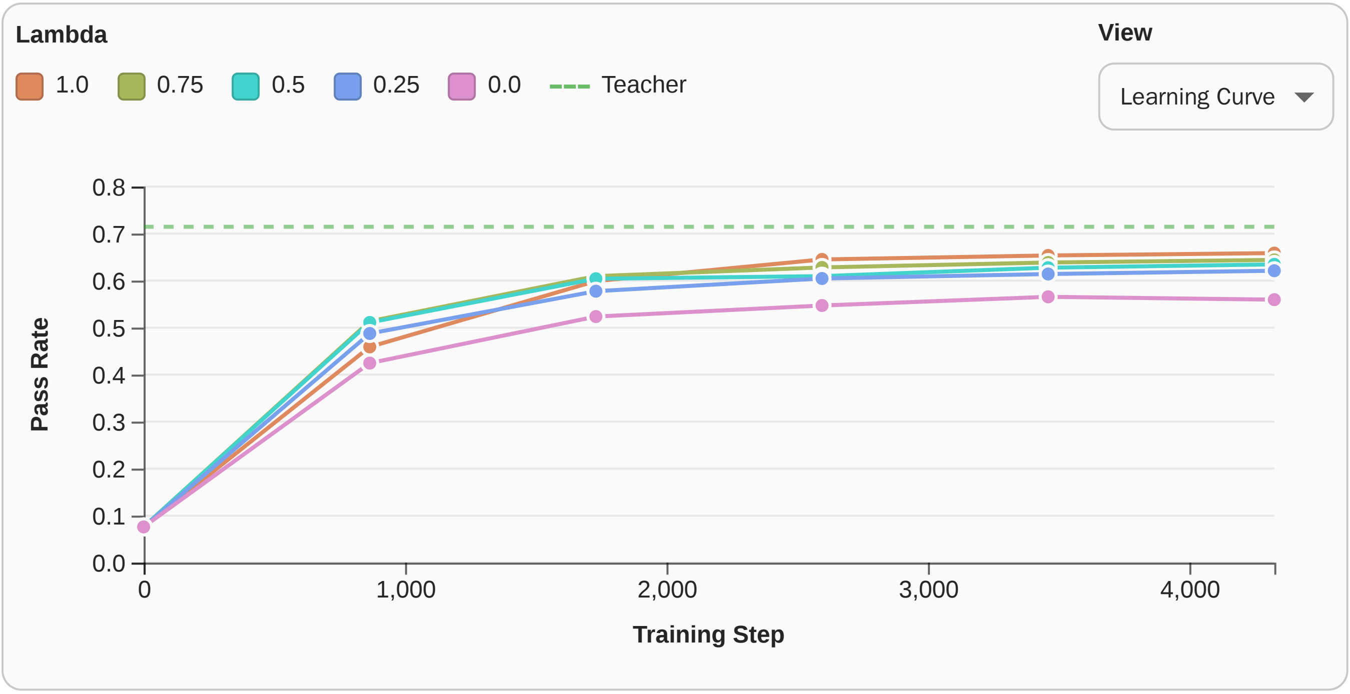


Figure 5: Ablation of the lambda parameter, that controls the blend of the on-policy loss (lambda=1.0) and supervised loss (lambda=0.0).

Distilled teacher knowledge

After testing multiple configurations, we achieved a setup that consistently distilled over 80% of a teacher's performance on the Countdown task. This high distillation ratio held true across multiple teacher models of different sizes (as shown in Figure 6), validating our on-policy GKD implementation.

These results underscore a fundamental point: a student's performance is effectively capped by the teacher's capabilities. This highlights the importance of selecting a strong teacher model to maximize student performance.

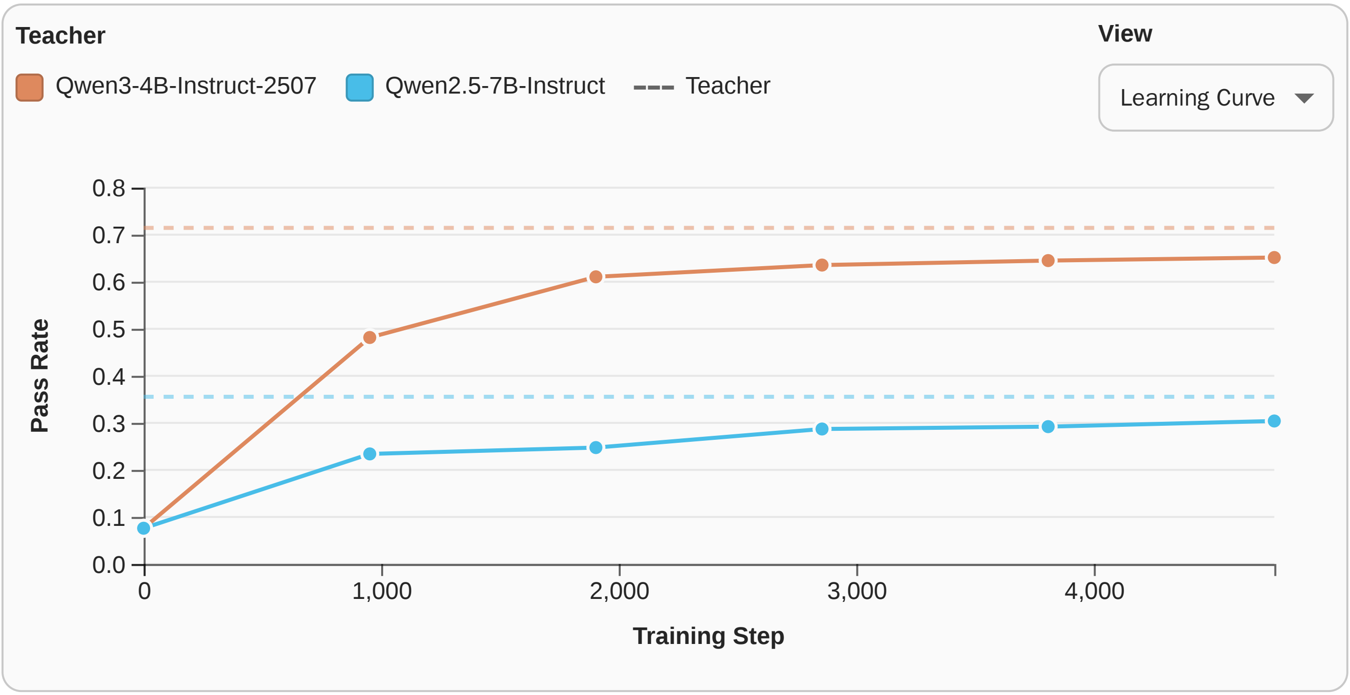


Figure 6: Distillation is stable at different model scales, with Qwen/Qwen2.5-1.5B-Instruct as the student and either Qwen/Qwen2.5-7B-Instruct or Qwen/Qwen3-4B-Instruct-2507 as the teacher. In both cases we are able to recover over 80% of the teacher's performance, which points to the importance of choosing a strong teacher to achieve the best results in KD tasks.

These results validate our GKD implementation. The next question is: can on-policy distillation still succeed when teacher and student use *different tokenizers*?

On-Policy distillation works with different tokenizers

While our GKD implementation recovered over 80% of the teacher's performance, it was limited to teacher-student pairs with matching tokenizers. Our next experiments addressed this limitation by testing distillation across different model families, which use different tokenizers.

This scenario requires methods that can handle vocabulary and sequence misalignments. We therefore compared the baseline ULD method with our proposed GOLD method to evaluate their effectiveness.

Tokenizer similarity impacts performance

Tokenizer similarity dictates the extent to which sequence and vocabulary alignment are required. We hypothesized that lower similarity would correlate with lower task performance, and our results confirm this: GOLD's performance on the Countdown task declines as tokenizer similarity decreases.

This decline is an expected trade-off, as the alignment process for divergent vocabularies inevitably introduces some noise. However, even with this effect, we will show that GOLD (at 0.64 similarity) still outperforms RL methods.

Model	Performance on Countdown	Similarity with Qwen3-4B-Instruct-
Qwen/Qwen2.5-1.5B-Instruct	0.6515	0.999974
meta-llama/Llama-3.2-1B-Instruct	0.4235	0.64
google/gemma-3-1b-it	0.0305	0.063

GOLD outperforms ULD

We tested our extensions by training `meta-llama/Llama-3.2-1B-Instruct` (student) with `Qwen/Qwen3-4B-Instruct-2507` (teacher). The results in Figure 7 show a substantial performance difference between the methods:

- GOLD improved the student's initial performance by 25% and recovered 60% of the teacher's performance.
- ULD improved the student by only 5% and recovered just 10% of the teacher's performance.

This difference is attributable to GOLD's improved alignment techniques. This specific student-teacher pair had 0 similarity under a strict ID match, but our token content matching (from Figure 4) increased this to 0.64. This, combined with our improved sequence alignment (from Figure 3), enabled effective knowledge transfer where ULD failed and produced results competitive with RL methods.

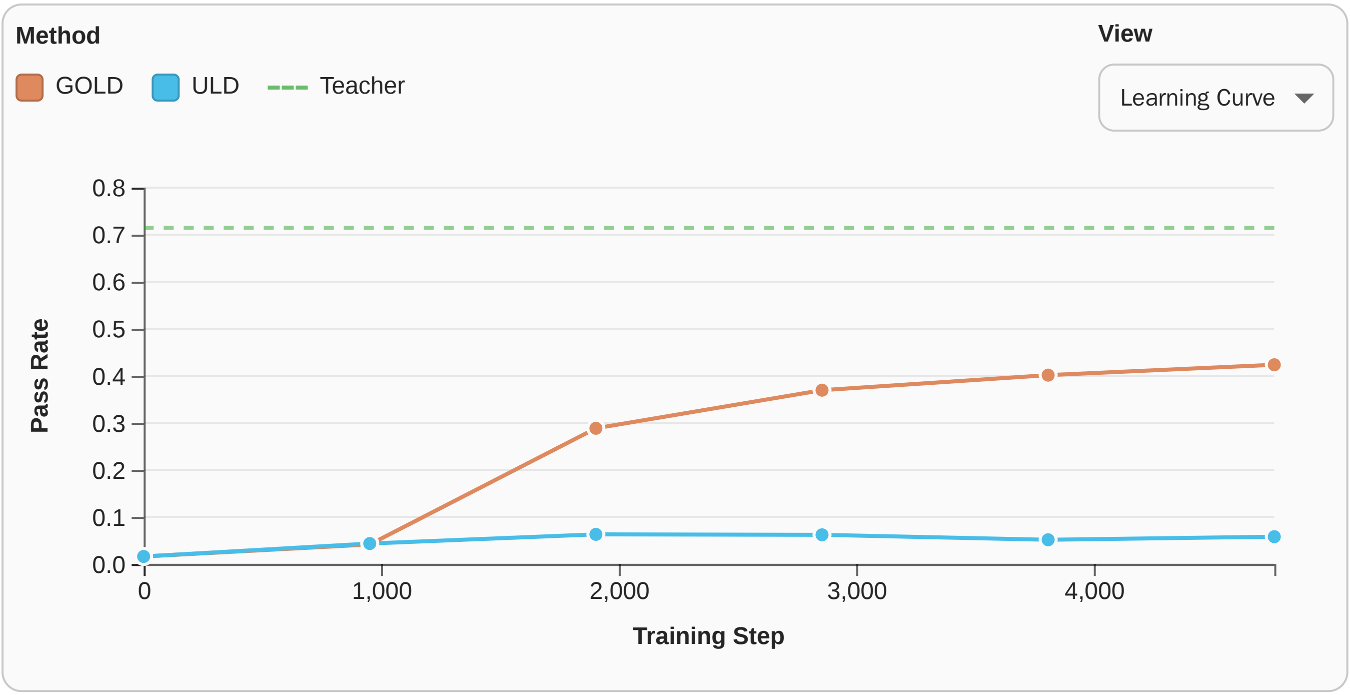


Figure 7: GOLD performs better than ULD when distilling Qwen/Qwen3-4B-Instruct-2507 into meta-llama/Llama-3.2-1B-Instruct. The plot also shows the long warmup in both cases because the model performance has a noticeable improvement only after the 1000th step.

Having shown that GOLD handles tokenizer differences effectively, we now benchmark it against an RL algorithm, GRPO, test its efficiency and performance.

On-policy distillation outperforms GRPO

On-policy distillation uses student-generated completions to progressively update the training data. Having established this approach is superior to offline methods like SFT (when tokenizers match), we next compared it to other on-policy methods, specifically Group Relative Policy Optimization (GRPO). GRPO is an RL method introduced in the DeepSeek-Math paper (Shao et al., 2024) and later popularized by the Deepseek R1 release (DeepSeek-AI et al., 2025).

We followed Philipp Schmid's tutorial of how to train GRPO for the Countdown task and compared it to the performance of KD distillation. Our reward function was a sum of three components:

1. Format: +1 if the response included the tags correctly.

2. Following Rules: +1 if the model followed the rule of using the numbers provided in the prompt and only using each number once.
3. Correct Equation: +1 if the equation is correct.

The implementation in the tutorial joined the Format and Following Rules reward into a single function, but we found that the results were better when splitting the conditions into two separate reward functions.

Figure 4 shows our results for the scenario with the same tokenizer (above) and different tokenizers (below). For the same tokenizer scenario, we see a that KD outperforms GRPO by a 2x performance! The scenario with different tokenizers has a narrower performance gap between KD and GRPO, but still GOLD performs 20% better than GRPO. These results align with Qwen 3 Technical Report, where on-policy distillation performs similarly or better than RL. However, our results go one step further because we perform better than RL using a student-teacher pairing from different model families and with different tokenizers.

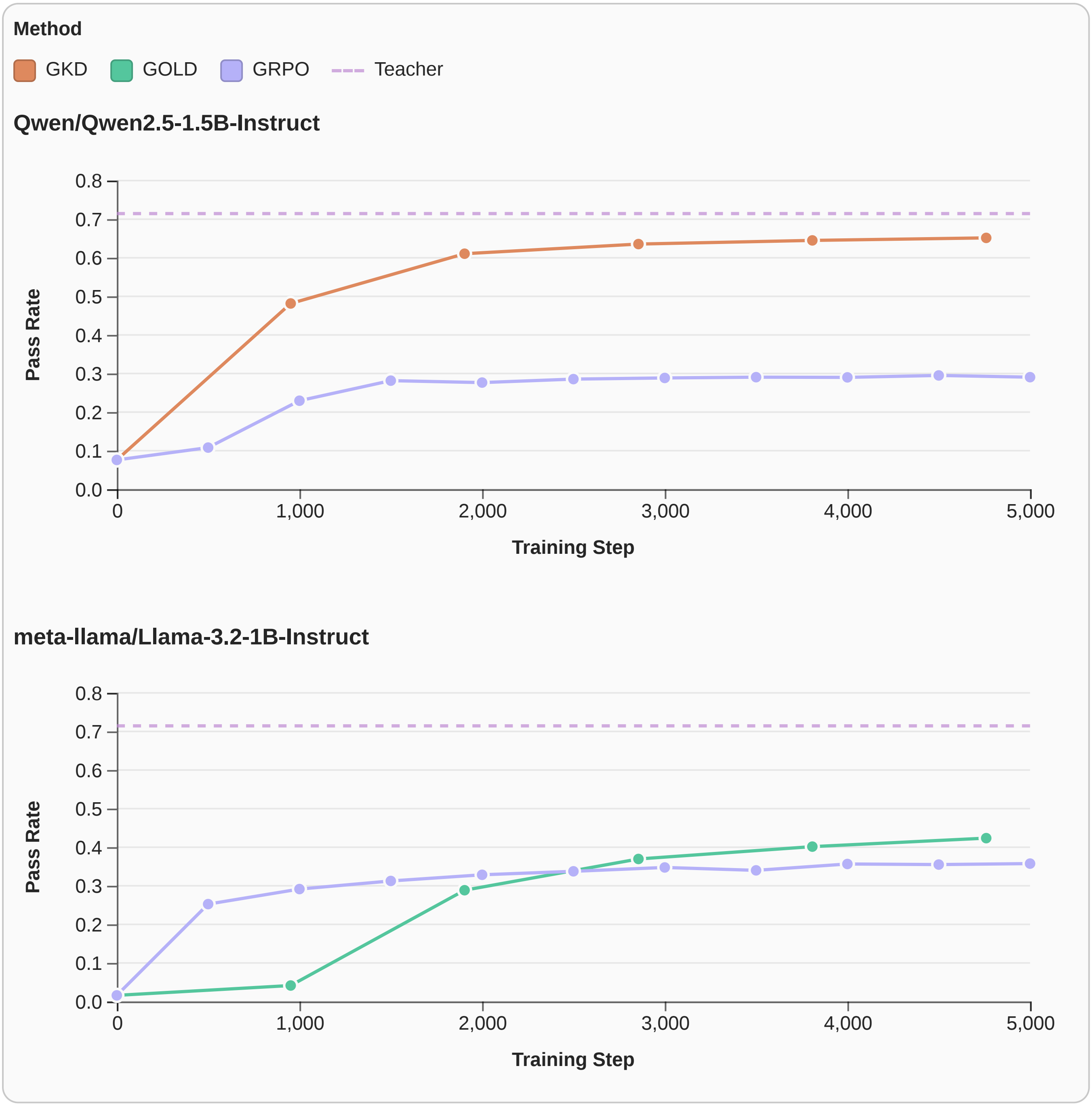


Figure 8: GKD and GOLD perform better than GRPO when training meta-llama/Llama-3.2-1B-Instruct. The gains from distillation are more clear in the GKD because we are able to distill the teacher better, but we still perform better than GRPO with our GOLD approach.

Beyond mathematical reasoning, on-policy distillation also applies to *domain-specific fine-tuning*. Let's explore how the same ideas improve personalization and task adaptation.

Distillation for Domain

In the Thinking Machines blog post, the authors distilled a language more for personalisation. They improved a Qwen3-8B model on an internal domain dataset and evaluation benchmark and restored it's ability on IFEval, the instruction following benchmark. This is useful because models can often lose their instruction following abilities during domain specific fine-tuning with SFT. Thinking Machines achieved this by interleaving phases on continued pre-training on domain specific data (mid-training) and On-policy Distillation with a high quality chat dataset, `allenai/tulu-3-sft-mixture`. As the table below shows, chat performance is restored following on-policy distillation.

To make these results reproducible, we'll now walk through how to implement the full process using open-source datasets and the TRL framework.

Reproducing in TRL

We can reproduce the above process in TRL and share the implementation using open models and datasets!

We've made some adaptations from the Thinking Machine experiment to use datasets and benchmarks that are available, instead of the "internal document dataset and benchmark":

- The `open-r1/codeforces` dataset as a domain specific dataset.

- The `livecodebench` evaluation benchmark to align with the Codeforces competitive coding task above.
- The same `allenai/tulu-3-sft-mixture` dataset and the IFEVal Benchmark.
- The `Qwen/Qwen3-4B-Instruct-2507` model.

Supervised Fine-Tuning on `open-r1/codeforces`

We fine-tuned the `Qwen/Qwen3-4B-Instruct-2507` model on `open-r1/codeforces` with `SFTTrainer` which improved performance by *35.1%* to *40.3%* on livecodebench. However, the model's IFEval score fell from *83.4%* to *79.5%*, which is common in domain specific fine-tuning.

Note that we trained this model for 1k steps and stopped early. For a more complete study of fine-tuning for advanced reasoning tasks, check out this blog post from the Open R1 project.

Generalized Knowledge Distillation on `allenai/tulu-3-sft-mixture`

Starting from the above checkpoint from SFT, we used the `GKDTrainer` with the `allenai/tulu-3-sft-mixture` dataset which improved performance on IFEval *79.5%* to *82.8%* whilst maintaining an approximate livecodebench score of *39.8%*.

Model	IFEval	LCB
Qwen3-4B	83.4	35.1
Qwen3-4B + Codeforces SFT	79.48	40.29
Qwen3-4B + Codeforces SFT + Tulu3 GKD	82.8	39.8

Results from first finetuning on Codeforces data to improve LCB and then recovering performance on IFEval by distilling the initial Qwen3-4B model.

Building it for yourself

If you want to try out knowledge distillation for yourself on your own use case, or a dataset from the hub, the recipe is available below.

SFT Recipe

Distillation Recipe

Conclusion

In this post, we introduced General On-Policy Logit Distillation (GOLD), a new method that enables effective on-policy knowledge distillation between models, even when the teacher and student do not share the same tokenizer vocabulary. This overcomes a significant limitation of existing on-policy methods like GKD, which require matched tokenizers.

GOLD builds upon the offline ULD method but extends it to the on-policy setting and, critically, addresses its two main weaknesses. First, we replace ULD's naive sequence truncation with a token-merging strategy that multiplies marginal distributions by scalar conditional probabilities. Second, we implement a hybrid vocabulary alignment method that uses a direct-mapping loss for shared tokens and falls back to ULD's sorting method only for unmatched tokens.

Our experiments on the Countdown math task confirm GOLD's advantages. We showed that GOLD significantly outperforms the original offline ULD implementation, recovering 60% of the teacher's performance versus ULD's 10%. Furthermore, GOLD proved superior to other on-policy methods, outperforming a supervised fine-tuning baseline by 15% and a GRPO baseline by 2x. Even in the difficult cross-tokenizer scenario, GOLD still outperformed GRPO by 20%.

These findings demonstrate that GOLD is a powerful and flexible technique for model distillation. It provides a path to distill knowledge from any high-performing teacher to any student, regardless of their tokenizer, offering a more effective and token-efficient alternative to reinforcement learning.

Citation

For attribution in academic contexts, please cite this work as

```
Carlos Miguel Patiño, Kashif Rasul, Quentin Gallouédec, Ben Burtenshaw, Sergio Paniego, Vaibhav
Srivastav, Thibaud Frere, Ed Beeching, Lewis Tunstall, Leandro von Werra, Thomas Wolf (2025).
"Unlocking On-Policy Distillation for Any Model Family".
```

BibTeX citation

```
@misc{patiño2025_unlocking_on_policy_distillation_for_any_model_family,
  title={Unlocking On-Policy Distillation for Any Model Family},
  author={Carlos Miguel Patiño and Kashif Rasul and Quentin Gallouédec and Ben Burtenshaw and
Sergio Paniego and Vaibhav Srivastav and Thibaud Frere and Ed Beeching and Lewis Tunstall and
Leandro von Werra and Thomas Wolf},
  year={2025},

}
```

References

1. Agarwal, R., Vieillard, N., Zhou, Y., Stanczyk, P., Ramos, S., Geist, M., & Bachem, O. (2024). *On-Policy Distillation of Language Models: Learning from Self-Generated Mistakes*. https://huggingface.co/papers/2306.13649↑ back: 1, 2

2. Boizard, N., Haddad, K. E., Hudelot, C., & Colombo, P. (2025). *Towards Cross-Tokenizer Distillation: the Universal Logit Distillation Loss for LLMs*. https://huggingface.co/papers/2402.12030↑

3. DeepSeek-AI, Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., Zhang, X., Yu, X., Wu, Y., Wu, Z. F., Gou, Z., Shao, Z., Li, Z., Gao, Z., ⋯ Zhang, Z. (2025). *DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning*. https://huggingface.co/papers/2501.12948↑

4. Gandhi, K., Lee, D., Grand, G., Liu, M., Cheng, W., Sharma, A., & Goodman, N. D. (2024). *Stream of Search (SoS): Learning to Search in Language*. https://huggingface.co/papers/2404.03683↑

5. Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y. K., Wu, Y., & Guo, D. (2024). *DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models*. https://huggingface.co/papers/2402.03300↑

6. Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., Zheng, C., Liu, D., Zhou, F., Huang, F., Hu, F., Ge, H., Wei, H., Lin, H., Tang, J., ⋯ Qiu, Z. (2025). *Qwen3 Technical Report*. https://huggingface.co/papers/2505.09388↑

Footnotes

1. The full GKD loss is then formally defined as: $\mathcal{L}_{GKD} := (1-\lambda)\mathbb{E}_{(x,y)\sim(X,Y)}[\mathcal{D}_{JSD(\beta)}] + \lambda\mathbb{E}_{x\sim X}[\mathbb{E}_{y\sim p_S(.|x)}[\mathcal{D}_{JSD(\beta)}]]$.

   ↑

2. The details of why we can merge the probabilities using the chain rule. For the merged distribution at position i: $P_{\text{merged}}(y) = P(y \mid x) \times P(\text{token}_1 \mid x) \times P(\text{token}_2 \mid \text{token}_1, x) \times \ldots$ This correctly computes the joint probability of the actual generated sequence while providing a reasonable approximation for counterfactual tokens.

   ↑

3. The β parameter then controls the generalized Jensen-Shannon divergence between the student (S) and teacher (T) distributions, calculated via the following loss summed over the sequence and averaged over the batch: $\mathcal{D}_{\text{JSD}(\beta)}(p_S, p_T) = \beta \cdot D_{\text{KL}}(p_S \| \pi) + (1-\beta) \cdot D_{\text{KL}}(p_T \| \pi)$ where $\pi = \beta \cdot p_S + (1-\beta) \cdot p_T$.

   ↑