Machine Learning Without Data: Training Small Expert Agents With Minimal Examples - Keynote

Jacek Golebiowski[1,*], Maciej Gryka[1], Vineeth Reddy Guda[1], Gabi Kadlecová[1], Bartek Kruszczyński[1], Selim Nowicki[1], Jonas Verschueren[1] and Usman Zafar[1]

[1]Distil Labs, https:// www.distillabs.ai

Abstract

Large language models (LLMs) have transformed how AI products are built, but their size, cost, and cloud dependency make them impractical in many production environments. Small language models (SLMs) address these constraints, yet training them typically requires large labeled datasets that are expensive to produce. We present an automated platform that turns a natural-language task description and as few as 10–50 labeled examples into a production-ready SLM expert agent. Our approach uses a teacher LLM to generate synthetic training data, applies a cascade of rule-based validators to curate that data, and fine-tunes a compact student model via knowledge distillation. To ensure diversity in the generated data, we model controllable properties of examples as latent variables within a Bayesian framework, sampling from a posterior informed by seed data before conditioning generation. Across two teacher-student pairings evaluated on 10 datasets, distilled 3–4B parameter students consistently match or exceed 70–120B parameter teachers while being 20–30× smaller.

Keywords

knowledge distillation, small language models, synthetic data generation, fine-tuning, expert agents

1. Introduction

Large language models have become the default building block for AI-powered products. However, deploying LLMs in production comes with significant constraints. In industries such as defense, cybersecurity, fintech, and robotics, data often cannot leave the organization's perimeter. Latency budgets are tight, compute is limited, and inference costs at scale can be prohibitive. Small language models solve all of these problems: they can run on private infrastructure, respond in milliseconds, and cost a fraction of cloud LLM inference. The challenge is that off-the-shelf SLMs rarely meet the accuracy bar required for production use.

Fine-tuning can close this gap, but conventional approaches require thousands of labeled examples, which are expensive and time-consuming to produce. In many enterprise settings, domain experts can provide a task description and a handful of representative examples, but not a full training corpus.

We present a platform that bridges this gap by automating the entire pipeline from minimal inputs to a deployed expert agent. Given a plain-English task description and 10–50 seed examples, our system (1) generates diverse synthetic training data using a teacher LLM, (2) curates that data through a cascade of rule-based validators, and (3) fine-tunes a compact student model on the resulting dataset. The key technical contribution is a controlled data generation strategy that models properties of examples as latent variables sampled before generation, preventing the repetitive outputs that plague naive LLM-based data augmentation.

We evaluate on 10 datasets across five task families and two teacher-student pairings (LLaMA 3.3 70B to LLaMA 3.2 3B, and GPT-OSS-120B to Qwen3-4B). The distilled students consistently match or exceed their teachers while being 20–30× smaller, suggesting that the approach generalizes well across common enterprise use cases.

Proceedings of the First Workshop on Small and Efficient Large Language Models for Knowledge Extraction (SmaLLEXT) @ CIKM 2025

*Corresponding author.

jacek@distillabs.ai (J. Golebiowski); maciej@distillabs.ai (M. Gryka); vineeth@distillabs.ai (V. R. Guda); gabi@distillabs.ai (G. Kadlecová); bartek@distillabs.ai (B. Kruszczyński); selim@distillabs.ai (S. Nowicki); jonas@distillabs.ai (J. Verschueren); usman@distillabs.ai (U. Zafar)

2. Method

2.1. Problem Setup and Inputs

The distillation pipeline requires three inputs:

- **Task description**: A plain-English definition of the task, including the expected input/output format and domain-specific constraints. An existing LLM prompt serves as a good starting point.
- **Seed dataset**: 10–50 pairs of (input, output) examples that define the target behavior. This is orders of magnitude less than conventional fine-tuning requires.
- **Additional context** (optional): Unstructured domain documents, logs, or industry literature that guide the teacher toward the correct problem space during data generation.

2.2. Synthetic Data Generation

The core challenge in synthetic data generation is producing examples that are both diverse and realistic. A naive approach of prompting an LLM to "generate more examples like these" tends toward repetitive, modal outputs. We address this by treating data generation as a structured sampling problem: we model controllable properties of examples as latent variables, update their distribution based on seed data, and sample from the resulting posterior before conditioning generation.

Controllable properties. We represent each controllable aspect of the data as a component of a latent vector $\boldsymbol{\theta}$. These properties define the "shape" of each example before it is generated. Table 1 lists the controllable properties used in the generation process.

Table 1
Controllable properties used to guide synthetic data generation.

Property	Description	Example values
Length	Target text length	short, medium, long
Topic	Subject matter or domain area	billing inquiry, tech. support
Difficulty	Complexity of the task instance	simple, moderate, complex
Target class	Which class or tool to generate	(task-specific)

Three-step generation process. We treat synthetic data generation as a probabilistic inference problem with the following three steps:

Step 1: Prior distribution $p(\boldsymbol{\theta})$. We begin with a broad prior distribution over all controllable properties. This prior represents initial beliefs about what kinds of examples are plausible before observing any seed data, and constitutes a hyperparameter of the method.

Step 2: Posterior update $p(\boldsymbol{\theta} \mid D)$. When we observe the small seed dataset D, we extract the values for all properties in $\boldsymbol{\theta}$ from each example using an LLM and update the prior:

$$p(\boldsymbol{\theta} \mid D) \propto p(D \mid \boldsymbol{\theta})\, p(\boldsymbol{\theta}) \tag{1}$$

This posterior captures which property combinations are both realistic and representative given the observed seeds.

Step 3: Conditional generation. For each synthetic example, we first sample properties $\boldsymbol{\theta}^{(j)}$ from the posterior $p(\boldsymbol{\theta} \mid D)$, then convert the sampled values into natural-language mutation instructions (e.g., "moderate" complexity becomes "The generated example should be of moderate complexity"), and finally prompt the teacher LLM with the task description, a few randomly selected seed examples, and the mutation instructions. The teacher generates a new example $\mathbf{x}^{(j)}$ conditioned on all three inputs:

$$\mathbf{x}^{(j)} \sim p(\mathbf{x} \mid \boldsymbol{\theta}^{(j)}) \tag{2}$$

By conditioning generation on samples from $p(\theta \mid D)$, we ensure synthetic examples remain faithful to the distributional characteristics observed in the seeds without over-indexing on those examples, because of the broader prior. Based on benchmarking, we currently default to using complexity as the primary controllable property, as we found that adding more mutation dimensions can degrade results in some cases.

Task-specific adaptations. While the core framework remains consistent across tasks, each task type requires specific adaptations:

- *Classification*: The target class is used as a controllable property. For each generation, we sample seed examples for both the target class and non-target classes, helping the model understand class boundaries.
- *Open-book QA*: Components are sampled from a user-provided knowledge database, and the teacher generates (context, question, answer) triplets grounded in that context.
- *Closed-book QA*: Generation mirrors open-book QA, but the context is removed after generation, leaving only (question, answer) tuples. This forces the student to internalize knowledge during fine-tuning.
- *Tool calling*: The target tool is used as a controllable property. The teacher generates examples conditioned on the full tool catalog, with seed examples for both target and non-target tools.
- *Information extraction*: A two-phase approach first generates diverse synthetic contexts, then produces extraction pairs (input text to structured output) for each context.

Generation volume. We target approximately 10,000 synthetic examples per task. Benchmarking across generation volumes (5k, 10k, 15k) showed 10k offers better results than 5k and comparable results to 15k, making it the optimal time-quality tradeoff.

2.3. Data Validation

Generated data is not always high-quality. Following the approach discussed in Self-Instruct [1], we apply a cascade of rule-based validators to filter synthetic examples before they enter the training set. Examples that pass all validators are appended to the curated dataset; examples that fail any validator are discarded.

Length validation. Examples too short or too long relative to the seed distribution are rejected. We compute length statistics from the seeds and reject examples outside an acceptable range (below the 5th or above the 95th percentile of seed lengths).

Format validation. Examples starting with non-alphanumeric characters are rejected as likely malformed. This catches common failure modes such as leading punctuation, markdown formatting tokens, or special characters.

Similarity validation. We reject examples that are either too similar to existing data (near-duplicates) or too dissimilar (off-distribution). For each generated example $\mathbf{x}$, we compute a novelty score:

$$\varphi(\mathbf{x}) = \max_i \text{ROUGE-L}(\mathbf{x}, \mathbf{a}_i) \tag{3}$$

where anchors A = seeds $\cup$ previously accepted synthetic examples. We enforce a novelty window $\tau_{\min} \leq \varphi(\mathbf{x}) \leq \tau_{\max}$, where examples with $\varphi(\mathbf{x}) > \tau_{\max}$ are rejected as near-duplicates and examples with $\varphi(\mathbf{x}) < \tau_{\min}$ are rejected as off-distribution. Default thresholds are $\tau_{\max} = 0.9$ and $\tau_{\min} = 0.1$. The anchor set grows as examples are accepted, so later generations are compared against all previously validated data.

Schema validation. For tasks with structured outputs, we validate that generated outputs conform to the expected schema, checking for correct types, required fields, enum constraints, and unexpected keys. This is particularly important for information extraction and tool calling tasks.

2.4. Model Training

Using the curated dataset (seed + validated synthetic), we specialize the student model using supervised fine-tuning (SFT) with LoRA [2] for parameter efficiency. For classification, we treat the problem as text generation: the model receives the input text and learns to generate the class name as output, leveraging the model's pre-trained understanding of class concepts.

Training uses a standardized configuration across tasks: 4 epochs, learning rate 5×10^{-5}, LoRA rank 64, LoRA alpha 64, targeting the `q_proj` and `v_proj` modules. These defaults were tuned via grid search across all benchmark tasks.

3. Experiments

3.1. Setup

Baselines. We compare four reference points under the same preprocessing, seed split, and evaluation harness:

- **Teacher** (reference): The teacher LLM evaluated with a fixed prompt and *k*-shot examples drawn only from the training split.
- **Trained Student** (ours): The student model trained on seed + curated synthetic data.
- **Seed Student**: The student model trained only on the initial seed set, without synthetic data augmentation.
- **Base Student** (lower bound): The student model prompted with the task description but no fine-tuning.

Task families. We evaluate across five task families to reflect common enterprise patterns: (1) classification, where input text is assigned to fixed categories (TREC, Banking77, Ecommerce, Mental Health); (2) information extraction, where structured objects are extracted from unstructured text (PII Redaction, Documentation); (3) open-book QA, where questions are answered given retrieved documents (HotpotQA, Roman Empire QA); (4) tool calling, where the model selects and parameterizes function calls (Pizza Tool Calling, Git Tool Calling); and (5) closed-book QA, where knowledge is embedded directly into the model during training (SQuAD 2.0).

Metrics. Classification tasks use accuracy. Information extraction, open-book QA, and closed-book QA use a binary LLM-as-a-Judge score. Tool calling tasks use exact-match comparison of predicted and target function calls converted to Python dicts. All results are averaged over 3 runs; standard deviations are reported where ≥ 0.005.

Model pairings. We report two benchmark sets: (1) LLaMA 3.3 70B teacher with LLaMA 3.2 3B student (~23× compression), and (2) GPT-OSS-120B teacher with Qwen3-4B-Instruct-2507 student (~30× compression).

3.2. Results

Benchmark 1: LLaMA 3.3 70B teacher, LLaMA 3.2 3B student. Results are shown in Table 2.

The distilled 3B student matches or exceeds the 70B teacher on 9 of 10 benchmarks. The only exception is Banking77, where the student (0.895) sits within one standard deviation of the teacher (0.91). The largest gains appear in tool calling, where the trained student exceeds the teacher by 20 points on Pizza Tool Calling and 14 points on Git Tool Calling.

Table 2
Benchmark 1 results. Teacher: LLaMA 3.3 70B; Student: LLaMA 3.2 3B. Bold indicates best score per row. All values averaged over 3 runs. NA indicates the baseline was not applicable for that task.

Dataset	Task Type	Teacher (↑)	Trained Student (↑)	Base Student (↑)	Seed Student (↑)
Ecommerce	Classification	0.905	**0.91 ± 0.01**	NA	0.88
TREC	Classification	0.85	**0.92 ± 0.005**	NA	0.81
Mental Health	Classification	0.82	**0.85 ± 0.01**	NA	0.80
Banking77	Classification	**0.91**	0.895 ± 0.02	NA	0.76
PII Redaction	Info Extraction	0.85 ± 0.02	**0.87 ± 0.01**	0.54 ± 0.03	0.73 ± 0.01
HotpotQA	Open-book QA	0.93 ± 0.01	**0.95 ± 0.01**	0.80 ± 0.01	0.85
Roman Empire QA	Open-book QA	0.98 ± 0.01	**0.99 ± 0.01**	0.91 ± 0.02	0.93 ± 0.01
Pizza Tool Call.	Tool Calling	0.50 ± 0.04	**0.70 ± 0.03**	0.00	0.21 ± 0.01
Git Tool Call.	Tool Calling	0.81 ± 0.03	**0.95 ± 0.01**	0.03 ± 0.01	–
SQuAD 2.0	Closed-book QA	0.57 ± 0.04	**0.64 ± 0.03**	0.43 ± 0.04	NA

Benchmark 2: GPT-OSS-120B teacher, Qwen3-4B-Instruct-2507 student. Results are shown in Table 3.

Table 3
Benchmark 2 results. Teacher: GPT-OSS-120B; Student: Qwen3-4B-Instruct-2507. Bold indicates best score per row.

Benchmark	Task Type	Teacher (↑)	Trained Student (↑)	Base Student (↑)	Δ
TREC	Classification	0.89	**0.93**	0.51	+0.03
Banking77	Classification	**0.92**	0.89	0.87	−0.03
Docs	Info Extraction	0.82	**0.84**	0.64	+0.02
Ecommerce	Classification	0.88	**0.90**	0.75	+0.03
HotpotQA	Open-book QA	**0.93**	**0.93**	0.88	+0.00
Mental Health	Classification	0.81	**0.82**	0.78	+0.01
Roman Empire QA	Open-book QA	0.75	**0.80**	0.65	+0.05
SQuAD 2.0	Closed-book QA	0.52	**0.71**	0.26	+0.19

The 4B student matches or exceeds the 120B teacher on 7 of 8 benchmarks, with Banking77 again the only exception at 3 points below the teacher. The most dramatic improvement is on SQuAD 2.0, where the student exceeds the teacher by 19 points, demonstrating how fine-tuning embeds domain knowledge more effectively than prompting for closed-book QA tasks.

3.3. Discussion

Several patterns emerge consistently across both benchmark sets.

Synthetic data generation is essential. The gap between the Seed Student and the Trained Student is large across all tasks, often exceeding 10 points. This confirms that the synthetic data pipeline provides signal far beyond what the seed examples alone can offer.

Prompt engineering is not enough. The Base Student, which uses only the task description without fine-tuning, performs far below the Trained Student across all tasks. This is especially pronounced in tool calling, where the Base Student scores near zero.

Students can outperform teachers. This counterintuitive result has two sources. First, the validation pipeline filters out the teacher's mistakes, so the student trains only on the teacher's best outputs. Second, task specialization allows the smaller model to allocate all its capacity to a single

domain, while the teacher is a general-purpose model spreading its capacity across many tasks. This is analogous to LLM self-improvement techniques, where filtered self-generated data improves model performance [1, 3, 4].

The approach generalizes across model families and tasks. Consistent results across both LLaMA and Qwen student models, paired with different teachers, and across five distinct task families, suggest the method is not specific to a particular model architecture or task type.

4. Related Work

Knowledge distillation. The foundational framework for transferring knowledge from large to small models was introduced by Hinton et al. [5]. DistilBERT [6] demonstrated practical transformer distillation, achieving 40% size reduction while retaining 97% of BERT's capability. MiniLLM [7] introduced reverse KL divergence for generative LLM distillation, addressing the problem of students overestimating low-probability teacher regions. Orca [8] pioneered "explanation tuning," training smaller models on rich reasoning traces from GPT-4, with Orca-13B reaching ChatGPT parity on several benchmarks. Hsieh et al. [9] showed that a 770M T5 model could outperform 540B PaLM few-shot by distilling both labels and rationales, using only 0.1% of standard fine-tuning data. Our work extends this line by generating the transfer dataset synthetically rather than requiring a large labeled corpus.

Synthetic data for training small models. Self-Instruct [1] pioneered bootstrapping instruction data from 175 seed tasks to 52K+ examples, establishing that models can effectively generate their own training data. Alpaca [10] demonstrated practical implementation, fine-tuning LLaMA-7B on 52K GPT-generated instructions at minimal cost. WizardLM [11] introduced Evol-Instruct for iteratively increasing example complexity. The Phi series [12, 13] challenged scaling assumptions by showing that curated synthetic "textbook-quality" data enables capabilities typically requiring 10× more parameters; Phi-1 (1.3B) achieved 50.6% on HumanEval using only 7B tokens. LIMA [14] and DEITA [15] further established that quality dramatically trumps quantity, with LIMA achieving GPT-4 preference parity using only 1,000 curated examples and DEITA reaching state-of-the-art with 6K samples. Our contribution differs in the explicit modeling of data properties as controllable variables within a Bayesian framework to ensure diversity, applied specifically to task-specific expert agents.

Model self-improvement and evaluation. Constitutional AI [16] established reinforcement learning from AI feedback, eliminating human labeling by using AI-generated preference labels. Zheng et al. [17] systematically validated LLM-as-a-Judge evaluation, showing GPT-4 achieves over 80% agreement with human preferences. STaR [3] introduced iterative rationale bootstrapping, achieving performance comparable to 30× larger models. DeepSeek-R1 [4] demonstrated that reasoning abilities emerge through pure RL with verifiable binary rewards, with distilled versions at 1.5B–70B parameters retaining strong performance. We use LLM-as-a-Judge for evaluation and plan to incorporate verifiable reward signals in future training stages.

Parameter-efficient fine-tuning. LoRA [2] provides parameter-efficient fine-tuning by training low-rank adapter matrices. We use LoRA throughout our pipeline, finding that rank 64 matches full fine-tuning quality for models in the 3B–8B range while significantly reducing memory requirements. Zephyr [18] demonstrated that alignment itself can be distilled, combining distilled SFT with distilled DPO to surpass LLaMA 2-Chat-70B on MT-Bench with just hours of training.

5. Conclusion

We have presented an automated platform for training small language model expert agents from minimal labeled data. By combining controlled synthetic data generation, cascaded data validation,

and parameter-efficient fine-tuning, we consistently produce 3–4B parameter models that match or exceed the accuracy of 70–120B parameter teachers across five task families and 10+ benchmarks. The approach requires only 10–50 seed examples and produces deployment-ready models in hours, making LLM-level accuracy accessible in environments where cost, latency, privacy, or compute constraints rule out cloud-based LLMs.

6. Declaration on Generative AI

During the preparation of this work, the author used Anthropic Claude Sonnet (claude.ai) to apply grammar and spelling checks, paraphrase and reword. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

References

[1] Y. Wang, Y. Kordi, S. Mishra, A. Liu, N. A. Smith, D. Khashabi, H. Hajishirzi, Self-instruct: Aligning language models with self-generated instructions, Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (2023).

[2] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen, LoRA: Low-rank adaptation of large language models, Proceedings of the International Conference on Learning Representations (2022).

[3] E. Zelikman, Y. Wu, J. Mu, N. D. Goodman, STaR: Bootstrapping reasoning with reasoning, Advances in Neural Information Processing Systems (2022).

[4] DeepSeek-AI, DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning, arXiv preprint arXiv:2501.12948 (2025).

[5] G. Hinton, O. Vinyals, J. Dean, Distilling the knowledge in a neural network, NeurIPS 2015 Deep Learning Workshop (2015).

[6] V. Sanh, L. Debut, J. Chaumond, T. Wolf, DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter, NeurIPS 2019 Workshop on Energy Efficient Machine Learning and Cognitive Computing (2019).

[7] Y. Gu, L. Dong, F. Wei, M. Huang, MiniLLM: Knowledge distillation of large language models, Proceedings of the International Conference on Learning Representations (2024).

[8] S. Mukherjee, A. Mitra, G. Jawahar, S. Agarwal, H. Palangi, A. Awadallah, Orca: Progressive learning from complex explanation traces of GPT-4, arXiv preprint arXiv:2306.02707 (2023).

[9] C.-Y. Hsieh, C.-L. Li, C.-K. Yeh, H. Nakhost, Y. Fujii, A. Ratner, R. Krishna, C.-Y. Lee, T. Pfister, Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes, Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (2023).

[10] R. Taori, I. Gulrajani, T. Zhang, Y. Dubois, X. Li, C. Guestrin, P. Liang, T. B. Hashimoto, Stanford Alpaca: An instruction-following LLaMA model, https://github.com/tatsu-lab/stanford_alpaca, 2023.

[11] C. Xu, Q. Sun, K. Zheng, X. Geng, P. Zhao, J. Feng, C. Tao, D. Jiang, WizardLM: Empowering large language models to follow complex instructions, Proceedings of the International Conference on Learning Representations (2024).

[12] S. Gunasekar, Y. Zhang, J. Aneja, C. C. T. Mendes, A. Del Giorno, S. Gopi, M. Javaheripi, P. Kauffmann, G. de Rosa, O. Saarikivi, et al., Textbooks are all you need, arXiv preprint arXiv:2306.11644 (2023).

[13] M. Abdin, S. A. Jacobs, A. A. Awan, et al., Phi-3 technical report: A highly capable language model locally on your phone, arXiv preprint arXiv:2404.14219 (2024).

[14] C. Zhou, P. Liu, P. Xu, S. Iyer, J. Sun, Y. Mao, X. Ma, A. Efrat, P. Yu, L. Yu, S. Zhang, G. Ghosh, M. Lewis, L. Zettlemoyer, O. Levy, LIMA: Less is more for alignment, Advances in Neural Information Processing Systems (2023).

[15] W. Liu, W. Zeng, K. He, Y. Jiang, J. He, What makes good data for alignment? a comprehensive study of automatic data selection in instruction tuning, Proceedings of the International Conference on Learning Representations (2024).

[16] Y. Bai, S. Kadavath, S. Kundu, A. Askell, J. Kernion, A. Jones, A. Chen, A. Goldie, A. Mirhoseini, C. McKinnon, et al., Constitutional AI: Harmlessness from AI feedback, arXiv preprint arXiv:2212.08073 (2022).

[17] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. P. Xing, H. Zhang, J. E. Gonzalez, I. Stoica, Judging LLM-as-a-judge with MT-Bench and Chatbot Arena, Advances in Neural Information Processing Systems (2023).

[18] L. Tunstall, E. Beeching, N. Lambert, N. Rajani, K. Rasul, Y. Belkada, S. Huang, L. von Werra, C. Fournier, N. Habib, N. Sarrazin, O. Sanseviero, A. M. Rush, T. Wolf, Zephyr: Direct distillation of LM alignment, arXiv preprint arXiv:2310.16944 (2023).