

# Reinforcement Learning with Process-Level Feedback for Efficient Mathematical Reasoning in Small Language Models

Banjeet Khosla

Department of Electrical Engineering and Computer Science, University of Missouri,  
Columbia, MO, USA.

banjeet.khosla@missouri.edu

Baljiwan Chander

Department of Computer Science, George Mason University, Fairfax, VA, USA.

baljiwanmail@gmu.edu

## Abstract

Mathematical reasoning remains one of the most demanding capabilities for language models because it requires multi-step inference, intermediate validation, and resistance to superficial pattern matching. Although large models have achieved notable gains through scale and reinforcement learning, their computational, financial, and environmental costs limit broad deployment. Small language models offer advantages in latency, privacy, and local execution, but they often lack the representational capacity and training signal efficiency required for robust mathematical reasoning. This paper examines reinforcement learning with process-level feedback as a system-level strategy for improving mathematical reasoning in small language models. Rather than treating reinforcement learning as a monolithic optimization procedure, the analysis emphasizes feedback acquisition, verifier reliability, trajectory filtering, reward aggregation, curriculum design, deployment infrastructure, and governance. Process-level feedback can provide denser credit assignment than outcome-only rewards, but it introduces annotation cost, verifier latency, reward hacking risk, and new accountability burdens. The paper argues that efficient reasoning improvement depends less on a single algorithm and more on the alignment of feedback granularity, model capacity, data provenance, evaluation protocols, and deployment constraints. It also considers fairness, robustness, sustainability, and policy implications for educational, scientific, and professional settings. The conclusion recommends hybrid outcome and process supervision, auditable verification layers, adaptive compute budgeting, staged deployment, and human oversight. These measures can make reinforcement learning with process-level feedback a practical pathway for small language models without reproducing the scale-dependent pathologies of larger systems.

## Keywords

reinforcement learning; process-level feedback; small language models; mathematical reasoning; efficient training; AI governance; robust deployment; fairness; sustainability.

## 1. Introduction

Mathematical reasoning is a distinctive test of machine intelligence because it combines language understanding, symbolic manipulation, procedural discipline, and self-correction. Large language models have improved substantially on such tasks through scaling, instruction

tuning, and reinforcement learning from human or model feedback [1], [2], [3], [4]. Chain-of-thought prompting and scratchpad methods have further shown that intermediate reasoning traces can improve performance on multi-step problems [5], [6]. Yet these gains are often tied to models with billions of parameters, extensive training compute, and centralized serving infrastructure. For many organizations, especially those concerned with privacy, latency, or cost, such models are impractical. Small language models therefore remain attractive, but their limited capacity makes it difficult to acquire and retain complex mathematical procedures through conventional fine-tuning alone.

Reinforcement learning offers a way to shape model behavior beyond next-token prediction. However, outcome-level reinforcement learning in mathematical domains is inefficient because a final answer provides only a sparse signal: a long trajectory of reasoning may be correct or incorrect, but the reward does not indicate which intermediate step failed. Process-level feedback addresses this limitation by scoring or validating intermediate reasoning states, thereby providing denser supervision and potentially reducing sample complexity. Recent work on TrajLite PPO illustrates this direction by combining scalable reasoning path filtering with alignment for small parameter large language models [7]. The approach is relevant not because it solves every architectural problem, but because it exposes the central system tension: finer feedback can improve learning efficiency, yet it also increases verification cost, infrastructure complexity, and governance obligations.

This paper develops a system-level account of reinforcement learning with process-level feedback for efficient mathematical reasoning in small language models. It does not propose a new algorithm or derive a formal optimization method. Instead, it analyzes the structural trade-offs that arise when process supervision is embedded in a training and deployment pipeline. The discussion spans feedback design, verifier architecture, trajectory filtering, compute allocation, robustness, fairness, accountability, and sustainability. It argues that the effectiveness of process-level reinforcement learning depends on aligning model capacity, feedback granularity, data provenance, evaluation metrics, and operational constraints. A small model may benefit from dense feedback, but only if the feedback channel is reliable, affordable, and governable. The paper proceeds by reviewing foundations, conceptualizing process feedback as a systems signal, examining architecture, analyzing trade-offs, and then considering governance and deployment implications.

## **2. Foundations: Small Language Models and Mathematical Reasoning**

The scaling literature has established that model performance often improves predictably with parameter count, data volume, and compute [14], [15]. This relationship has encouraged the development of very large models and has shaped expectations about reasoning ability. However, scaling is not the only pathway to capability. Small language models can be adapted through parameter-efficient methods such as low-rank adaptation and quantized fine-tuning, which reduce memory and training costs while preserving much of the base model’s language competence [17], [18]. Open foundation models have also made it feasible to study smaller architectures in controlled settings [16]. The result is a heterogeneous ecosystem in which small models may be deployed at the edge, embedded in domain applications, or used as components in larger ensembles.

Mathematical reasoning, however, exposes the limitations of small models. Multi-step arithmetic, algebraic manipulation, and proof-like argumentation require maintaining intermediate state, applying rules consistently, and recovering from local errors. Chain-of-thought prompting can elicit intermediate steps [5], and scratchpad training can improve

intermediate computation [6]. Self-consistency methods sample multiple reasoning paths and select the most consistent answer, which improves accuracy at additional inference cost [9]. ReAct and Reflexion-style approaches interleave reasoning with actions or verbal self-feedback, showing that external tools and iterative critique can compensate for limited internal capacity [10], [11]. Bootstrapping methods such as STaR use self-generated reasoning traces to improve performance [8]. These techniques suggest that small models can reason better when supported by structured feedback and external validation, but they also highlight the fragility of relying on self-generated traces without reliable verification.

Reinforcement learning provides a framework for optimizing behavior through reward signals [1]. Trust region and proximal policy optimization methods have become standard for stable policy updates in complex environments [12], [13]. In language modeling, reinforcement learning from human feedback has been used to align outputs with preferences and instructions [2], [3], [4]. Yet mathematical correctness is not merely a preference; it is often verifiable. This distinction creates an opportunity for process-level feedback, because intermediate steps can sometimes be checked by symbolic engines, unit tests, or domain-specific validators. At the same time, verification introduces new dependencies. A model trained against an imperfect verifier may learn to exploit its weaknesses, producing reasoning that appears valid under the verifier but fails in real use. Thus, the design of feedback is inseparable from the design of the verification infrastructure.

### **3. Process-Level Feedback as a Systems Signal**

Process-level feedback refers to any signal that evaluates intermediate reasoning states rather than only final answers. It may come from human annotators who label steps as valid, invalid, or uncertain; from symbolic solvers that check equations or logical entailments; from execution environments that test code or numerical results; or from reward models trained to predict step quality. Outcome-level rewards indicate whether a final answer is correct, but they do not localize error. Process-level rewards can assign credit to specific steps, making it easier for a small model to learn which reasoning moves are productive. This is particularly important when the model has limited capacity, because it cannot easily recover from misleading intermediate states. Dense feedback can therefore improve sample efficiency and reduce the number of full trajectories needed for learning [7], [8].

The systems implications are substantial. First, process-level feedback changes the data pipeline. Instead of storing only prompts and final answers, the system must store reasoning paths, step boundaries, annotations, verifier outputs, and provenance metadata. This increases storage and privacy requirements. Second, it changes the training loop. The policy must be evaluated not only on final correctness but also on step-level compatibility with the verifier. Third, it changes the evaluation regime. A model may achieve high final-answer accuracy while using invalid intermediate steps, or it may produce valid steps but fail to reach a correct conclusion. Evaluation must therefore separate process validity, outcome correctness, and robustness under distribution shift. Fourth, it changes governance. If human annotators generate step-level labels, their labor conditions, instructions, and disagreements become part of the model’s lineage. If automated verifiers are used, their error modes and update cycles must be documented.

Process-level feedback is not automatically superior. It can introduce reward hacking, where the policy learns to satisfy superficial verifier criteria rather than genuine mathematical structure. It can also create a mismatch between the verifier and the target domain. A verifier trained on arithmetic may not transfer to geometry or proof writing. Annotation cost may be

high because step-level labeling is more cognitively demanding than final-answer labeling. Verifier latency can dominate training time if every step requires an external call. For small language models, these costs can erode the expected efficiency gains. The key question is not whether process feedback is useful in principle, but under what conditions its benefits exceed its overhead. This depends on the reliability of the verifier, the granularity of feedback, the cost of annotation, the model’s capacity, and the deployment context.

#### **4. Architecture for Efficient Reinforcement Learning with Process Feedback**

A practical architecture for process-level reinforcement learning in small language models must integrate several subsystems. The data layer manages prompts, reasoning traces, step annotations, verifier outputs, and metadata about source, license, and consent. The curriculum layer sequences tasks by difficulty, domain, and feedback density, allowing the model to learn basic procedures before encountering ambiguous or multi-domain problems. The inference layer generates candidate reasoning paths, often with sampling or temperature control to encourage diversity. The verification layer evaluates intermediate steps using symbolic tools, learned reward models, human review, or hybrid combinations. The reward aggregation layer converts step-level signals into a training signal, balancing outcome correctness, process validity, conciseness, and safety. The policy optimization layer updates the model, often using proximal policy optimization or related methods [12], [13]. The evaluation layer measures final accuracy, step validity, calibration, robustness, and cost. The governance layer tracks model versions, verifier versions, annotation instructions, and audit records.

The efficiency of this architecture depends on where computation is spent. If the verifier is expensive, the system may need to filter trajectories before verification. Trajectory filtering can discard obviously malformed paths, redundant samples, or low-confidence candidates, reducing verifier load. Recent work on scalable reasoning path filtering for small parameter models [7] is relevant because it treats filtering as a first-class efficiency mechanism rather than an afterthought. Filtering can also improve stability by preventing the policy from overfitting to noisy or degenerate trajectories. However, filtering introduces its own bias. If the filter favors fluent but shallow reasoning, the model may learn shortcuts. If it favors long chains, the model may waste inference budget. The filter must therefore be evaluated as part of the learning system, not as a neutral preprocessing step.

The choice of policy optimization also matters. Proximal policy optimization is popular because it constrains updates and can handle variable-length trajectories [12]. Trust region methods provide similar stability guarantees but may be more computationally demanding [13]. In small models, the optimizer’s memory footprint, batch size, and rollout length interact with hardware constraints. If the model is quantized or adapted with low-rank methods [17], [18], the training pipeline must account for numerical precision, adapter merging, and evaluation drift. The feedback loop between generation, verification, and policy update can become the dominant cost. A system that verifies every step with a large external model may be slower than outcome-only training, even if it requires fewer trajectories. Thus, efficiency should be measured in wall-clock time, energy, memory, and human labor, not only in sample count.

#### **5. Efficiency, Robustness, and Trade-offs**

Efficiency in process-level reinforcement learning is a multidimensional property. Sample efficiency improves when dense feedback reduces the number of full trajectories needed to learn a procedure. Computational efficiency may worsen if step-level verification is expensive.

Human efficiency may worsen if annotations require domain experts. Energy efficiency may improve if smaller models replace larger ones, but it may decline if training requires repeated verifier calls. The net effect depends on the ratio of verifier cost to model cost, the reuse of verified trajectories, and the ability to cache or distill verification signals. A small model trained with process feedback may be cheaper to deploy but more expensive to train than a larger model fine-tuned with outcome rewards. This trade-off must be evaluated over the full lifecycle.

Robustness is equally important. A process-level reward model can be brittle under distribution shift. If the verifier is trained on one textbook style, it may penalize valid alternative methods. If the policy learns to imitate verifier quirks, it may fail on external benchmarks. Adversarial prompts can exploit verifier blind spots, producing reasoning that passes automated checks but is mathematically invalid. Robustness therefore requires multiple verification channels, uncertainty estimation, and periodic red-teaming. Safety concerns such as specification gaming and negative side effects are relevant here [21]. The system may optimize for a proxy rather than the intended objective, especially when the proxy is a learned reward model. Human oversight remains necessary, but it must be targeted where the verifier is least reliable.

Technical debt is another trade-off. Machine learning systems often accumulate hidden dependencies, unstable data pipelines, and feedback loops that are difficult to maintain [22]. Process-level reinforcement learning adds verifier versions, annotation guidelines, reward model checkpoints, and trajectory stores. If these components are not documented and tested, the system can become fragile. A small model that performs well in a demonstration may fail after a data refresh or verifier update. Governance practices such as model cards and datasheets can reduce this risk by documenting intended use, limitations, and data provenance [24], [25]. These practices are not merely compliance artifacts; they are engineering tools for managing complexity.

## **6. Governance, Fairness, and Policy Implications**

Process-level feedback raises governance questions because it encodes judgments about what counts as valid reasoning. If those judgments are biased, the model may reproduce them. For example, a verifier trained primarily on one cultural or educational tradition may penalize alternative but valid solution methods. A reward model may favor terse reasoning over explanatory reasoning, disadvantaging learners who benefit from elaboration. In educational deployment, a small language model that grades mathematical steps could influence student outcomes, teacher workloads, and access to advanced instruction. Fairness therefore requires attention to representation, error distribution, and contestability. Stakeholders need mechanisms to challenge incorrect feedback, request human review, and audit model behavior across demographic and linguistic groups [19], [20], [23].

Accountability is complicated by the layered nature of the system. The model developer, verifier provider, dataset curator, deployment operator, and end user may all contribute to outcomes. Internal algorithmic auditing frameworks emphasize documenting design choices, testing for disparate impact, and assigning responsibility across the lifecycle [23]. Model cards and datasheets provide structured reporting on performance, limitations, and data characteristics [24], [25]. For process-level reinforcement learning, these documents should include verifier accuracy, step-level disagreement rates, reward hacking incidents, annotation protocols, and known failure modes. Without such records, it is difficult to determine whether a failure originates in the policy, the reward model, the verifier, or the data.

Policy implications extend beyond individual systems. If small language models with process-level feedback become widely used in education, scientific assistance, or professional certification, regulators may need to define minimum standards for verification and disclosure. These standards should be proportionate to risk. A low-stakes tutoring tool may require less oversight than an automated assessment system used for admissions or employment. Sustainability is also a policy concern. Smaller models can reduce inference energy, but process-level training may increase compute if verifiers are repeatedly invoked. Energy-aware scheduling, caching, and distillation can mitigate this. The governance goal should be to capture the efficiency benefits of small models while preventing hidden costs from shifting to annotators, verifiers, or the environment [19], [20], [21], [22].

## **7. Evaluation and Deployment Perspectives**

Evaluation must move beyond final-answer accuracy. A small language model trained with process-level feedback should be assessed on process validity, outcome correctness, robustness to paraphrases, calibration, and cost. Process validity measures whether intermediate steps are mathematically sound. Outcome correctness measures whether the final answer is right. Robustness measures performance under distribution shift, adversarial inputs, and verifier noise. Calibration measures whether the model’s confidence aligns with correctness. Cost measures training time, inference latency, memory, energy, and human annotation burden. A model that improves accuracy by 2 percent but doubles verifier cost may be less useful than a model that improves robustness with modest additional compute.

Deployment should be staged. In an edge tutoring scenario, the model might generate steps and use a lightweight verifier to flag likely errors. The user can then request a symbolic check or human review. In an automated assessment scenario, the model might only provide suggestions, with final grading performed by a human. In a scientific assistant scenario, the model might propose derivations but require external computation for validation. These configurations differ in feedback granularity, latency tolerance, and accountability. A single training pipeline cannot serve all of them without adaptation. The system should expose feedback confidence, allow fallback to larger models or symbolic tools, and log interactions for audit.

Cross-domain comparisons are instructive. In software engineering, process feedback comes from compilers, tests, and runtime traces. In mathematics, it comes from symbolic solvers, proof assistants, or expert review. In writing, it may come from rubrics or human preferences. The mathematical case is attractive because correctness is often verifiable, but it is not uniformly verifiable. Arithmetic and algebra are more amenable to automated checking than open-ended proof construction. This variation suggests a tiered approach: use dense process feedback where verification is cheap and reliable, and use outcome or human feedback where verification is expensive or ambiguous. Small language models may benefit most from hybrid signals that combine cheap symbolic checks with occasional expert review. Such an approach aligns with recent work on reasoning path filtering and alignment for small parameter models [7], while also acknowledging the limits of automatic verification.

## **8. Conclusion**

Reinforcement learning with process-level feedback offers a promising route to efficient mathematical reasoning in small language models. By providing denser credit assignment than outcome-only rewards, process feedback can help limited-capacity models learn procedures that would otherwise require extensive data or scale. However, the benefits are

conditional. Process feedback introduces verifier cost, annotation burden, reward hacking risk, and governance complexity. The efficiency of the overall system depends on how feedback granularity, trajectory filtering, policy optimization, and deployment infrastructure are aligned. A small model is not automatically sustainable simply because it has fewer parameters; its training pipeline may consume substantial compute and human labor if verification is poorly designed.

The system-level analysis suggests several design principles. First, feedback should be adaptive. Cheap outcome signals can guide initial learning, while process signals can be applied selectively to difficult or high-stakes problems. Second, verification should be plural. Symbolic tools, learned reward models, and human review each have different error profiles. Third, evaluation should be multidimensional. Accuracy alone cannot capture process validity, robustness, fairness, or cost. Fourth, governance should be embedded in the pipeline. Model cards, datasheets, audit trails, and contestability mechanisms are necessary for accountable deployment. Fifth, deployment should be staged and context-sensitive. Educational, scientific, and professional applications require different levels of oversight.

Future research should examine how process-level feedback interacts with model size, domain complexity, and verifier reliability. It should also investigate methods for distilling verified reasoning into smaller models without preserving verifier dependence. Fairness research should examine whether process supervision advantages some reasoning styles over others. Sustainability research should compare the full lifecycle costs of small-model reinforcement learning against alternative training strategies. Policy research should clarify when automated mathematical reasoning systems require external audit. With careful architecture and governance, reinforcement learning with process-level feedback can make small language models more capable and trustworthy without simply reproducing the scale-intensive trajectory of larger systems.

## References

1. Sutton, R. S., & Barto, A. G. (2018). Reinforcement learning: An introduction (2nd ed.). MIT Press.
2. Christiano, P. F., Leike, J., Brown, T., Martic, M., Legg, S., & Amodei, D. (2017). Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems* (Vol. 30, pp. 4299–4307). Curran Associates.
3. Bai, Y., Jones, A., Ndousse, K., Askell, A., Chen, A., DasSarma, N., Drain, D., Fort, S., Ganguli, D., Henighan, T., Joseph, N., Kadavath, S., Kernion, J., Conerly, T., El-Showk, S., Elhage, N., Hatfield-Dodds, Z., Hernandez, D., Hume, T., ... Kaplan, J. (2022). Training a helpful and harmless assistant with reinforcement learning from human feedback. arXiv preprint arXiv:2204.05862.
4. Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., & Lowe, R. (2022). Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems* (Vol. 35, pp. 27730–27744).
5. Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E. H., Le, Q. V., & Zhou, D. (2022). Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems* (Vol. 35, pp. 24824–24837).

6. Nye, M., Andreassen, A. J., Gur-Ari, G., Michalewski, H., Austin, J., Bieber, D., Dohan, D., Lewkowycz, A., Bosma, M., Luan, D., Sutton, C., & Odena, A. (2021). Show your work: Scratchpads for intermediate computation with language models. *arXiv preprint arXiv:2112.00114*.
7. Li, Q. (2026, May). TrajLite PPO: Scalable Reasoning Path Filtering and Alignment for Small Parameter Large Language Models. In *2026 2nd International Conference on Artificial Intelligence and Digital Ethics (ICAIDE)* (pp. 29-32). IEEE.
8. Zelikman, E., Wu, Y., Mu, J., & Goodman, N. D. (2022). STaR: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems* (Vol. 35, pp. 15476–15488).
9. Wang, X., Wei, J., Schuurmans, D., Le, Q. V., Chi, E. H., Narang, S., Chowdhery, A., & Zhou, D. (2022). Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*.
10. Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., & Cao, Y. (2023). ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*.
11. Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K., & Yao, S. (2023). Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems* (Vol. 36, pp. 8634–8652).
12. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.
13. Schulman, J., Levine, S., Abbeel, P., Jordan, M., & Moritz, P. (2015). Trust region policy optimization. In *International Conference on Machine Learning* (pp. 1889–1897). PMLR.
14. Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., & Amodei, D. (2020). Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.
15. Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., de Las Casas, D., Hendricks, L. A., Welbl, J., Clark, A., Hennigan, T., Noland, E., Millican, K., van den Driessche, G., Damoc, B., Guy, A., Osindero, S., Simonyan, K., Elsen, E., ... Sifre, L. (2022). Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*.
16. Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M. A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., & Lample, G. (2023). LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.
17. Dettmers, T., Pagnoni, A., Holtzman, A., & Zettlemoyer, L. (2023). QLoRA: Efficient finetuning of quantized LLMs. In *Advances in Neural Information Processing Systems* (Vol. 36, pp. 10088–10115).
18. Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., & Chen, W. (2022). LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

19. Bender, E. M., Gebru, T., McMillan-Major, A., & Shmitchell, S. (2021). On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency* (pp. 610–623). ACM.
20. Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., Brynjolfsson, E., Buch, S., Card, D., Castellon, R., Chatterji, N., Chen, A., Creel, K., Davis, J. Q., Demszky, D., ... Liang, P. (2021). On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*.
21. Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., & Mané, D. (2016). Concrete problems in AI safety. *arXiv preprint arXiv:1606.06565*.
22. Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., Chaudhary, V., Young, M., Crespo, J. F., & Dennison, D. (2015). Hidden technical debt in machine learning systems. In *Advances in Neural Information Processing Systems* (Vol. 28, pp. 2503–2511).
23. Raji, I. D., Smart, A., White, R. N., Mitchell, M., Gebru, T., Hutchinson, B., Smith-Loud, J., Theron, D., & Barnes, P. (2020). Closing the AI accountability gap: Defining an end-to-end framework for internal algorithmic auditing. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency* (pp. 33–44). ACM.
24. Mitchell, M., Wu, S., Zaldivar, A., Barnes, P., Vasserman, L., Hutchinson, B., Spitzer, E., Raji, I. D., & Gebru, T. (2019). Model cards for model reporting. In *Proceedings of the Conference on Fairness, Accountability, and Transparency* (pp. 220–229). ACM.
25. Gebru, T., Morgenstern, J., Vecchione, B., Vaughan, J. W., Wallach, H., Daumé III, H., & Crawford, K. (2021). Datasheets for datasets. *Communications of the ACM*, 64(12), 86–92.