# Always Verify! A Survey on Verification Approaches for Test-time Scaling

**Anonymous ACL submission**

## Abstract

Test-time scaling (TTS) has emerged as a new frontier for scaling the performance of Large Language Models. In test-time scaling, by using more computational resources during inference, LLMs can improve their reasoning process and task performance. Several approaches have emerged for TTS such as distilling reasoning traces from another model or exploring the vast decoding search space by employing a verifier. The verifiers serve as reward models that help score the candidate outputs from the decoding process to diligently explore the vast solution space and select the best outcome. This paradigm commonly termed has emerged as a superior approach owing to parameter free scaling at inference time and high performance gains. The verifiers could be prompt-based, fine-tuned as a discriminative or generative model to verify process paths, outcomes or both. Despite their widespread adoption, there is no detailed collection, clear categorization and discussion of diverse verification approaches and their training mechanisms. In this survey, we cover the diverse approaches in the literature and present a unified view of verifier training, types and their utility in test-time scaling.

## 1 Introduction

Large language models (LMs) have largely been driven by scaling up train-time compute through large-scale self-supervised pretraining (Kaplan et al., 2020; Hoffmann et al., 2022). Recently a new paradigm: *test-time scaling* has been evolving where additional computational resources are allocated during inference, allowing models to further refine their predictions and improve performance.

The central idea behind test-time scaling (TTS) is that increasing the compute budget at inference can yield substantial gains in model performance. This additional compute is used to search over the space of possible solutions. The test-time scaling mechanisms can be divided into two classes namely the **verifier-free** approaches which generate reasoning traces from larger model and distill them to a smaller LLM or **verifier-based approaches** which employ an external signal to guide search over solution space to select the best one.

The verification of generated solutions and reasoning paths is central to test-time scaling. Recent studies (Setlur et al., 2025b) have demonstrated that scaling test-time compute without verifiers is suboptimal and the performance gap between *verification-based* scaling and *verifier-free* scaling widens as the allotted compute increases.

Common approaches for TTS include letting the LLM generate a long chain-of-thought trajectory (Wei et al., 2022a; Kojima et al., 2022; OpenAI, 2024; Zeng et al., 2024), or asking the LLM to iteratively refine the solutions that it has generated (Davis et al., 2024; Madaan et al., 2023; Du et al., 2024; Yin et al., 2023, 2024) termed as modifying proposal distribution. Another category of approaches entails sampling multiple candidate solutions and then choosing the best one via majority voting (Chen et al., 2024b; Wang et al., 2023; Brown et al., 2024; Li et al., 2024), ranking with pairwise comparisons (Jiang et al., 2023), or using an external verifier or trained reward model(Kambhampati et al., 2024; Stroebl et al., 2024) (Cobbe et al., 2021; Lightman et al., 2024; Zhang et al., 2024a).

Employing external verifiers or self-verification is crucial for test-time scaling, as they help guide the search process over large reasoning space. *Verification for test-time scaling entails mechanisms or scoring functions used to evaluate the quality or plausibility of different reasoning paths or solutions from the language model during inference, enabling efficient search or selection among them without access to ground-truth labels.*

The verifiers can be divided into process-based namely Process Reward Models (PRMs) (Uesato

| Category | Approach | Approach Description | Type |
|---|---|---|---|
| **Heuristic** | (1) Deepseek-R1 (DeepSeek-AI et al., 2025) | Heuristic check for domain-specific problems | O |
| | (2) RvLLM (Zhang et al., 2025b) | Uses domain specific rules to verify solution candidates | P |
| **Discriminative** | (3) Naive ORM (Cobbe et al., 2021) | Trains solution-level and token-level verifiers on labeled-dataset | O |
| | (4) OVM (Yu et al., 2024) | Train a value model under outcome supervision for guided decoding | O |
| | (5) Naive ORM (Chen et al., 2024c) | SFT a domain-specific LM as a discriminator | O |
| | (6) Naive PRM (Lightman et al., 2024) | SFT an LM as a PRM on each reasoning step over mathematical tasks | O |
| | (7) AutoCv (Lu et al., 2024) | Trains an ORM first to use its confidence variations as step-level annotations to train a PRM | O, P |
| | (8) Math-Shepherd (Wang et al., 2024b) | Trains discriminative PRM using automated process-level annotations | P |
| | (9) DiVeRSe (Li et al., 2023b) | Employs voting and step-wise voting verifier trained using binary cross-entropy loss | P |
| | (10) Alphamath (Chen et al., 2024a) | Trains a process-level verifier using data from MCTS rollouts | P |
| | (11) $Re^2Search++$ (Xiong et al., 2025) | Trains a ceritic/verifier with contrastive loss on preference data | O |
| | (12) VersaPRM (Zeng et al., 2025) | extends to domain beyond math by augmenting MMLU with step-level annotations | P |
| **Generative** | | | |
| - Classical SFT | (13) Generative Verifier (Zhang et al., 2024a) | Exploit the generative ability of LLM-based verifiers via reformulating the verification | O, P |
| | (14) UQ-PRM (Ye et al., 2025) | Quantifies uncertainty of verifier for each step of reasoning | P |
| | (15) WoT (Zhang et al., 2024c) | Multi-Perspective Verification on : Assertion, Process, and Result wiht single verifier | O, P |
| | (16) Multi-Agent Verifiers (Lifshitz et al., 2025) | Multi-Perspective Verification with multiple verifiers | O, P |
| | (17) SyncPL (Liang et al., 2025) | Trains a generative process verifier using criteria based data generation | P |
| - Self-verification (classical SFT) | (18) ReVISE (Lee et al., 2025) | Trains on self-generated training data using SFT to elicit self-verification | O |
| | (19) Self-Reflection Feedback (Li et al., 2025c) | formulate the feedback utilization as an optimization problem and solve during test-time | O |
| | (20) ToolVerifier (Mekala et al., 2024) | Trains a LLM on data generated for verifying tool usage | P |
| | (21) T1 (Kang et al., 2025) | Trains a LLM using SFT by distilling from tool integrated data generation for self-verification | O |
| - RL-based | (22) GenRM (Mahan et al., 2024) | Trains a generative verifier on preference data by unifying RL from human and AI feedback. | P |
| | (23) V-STaR (Hosseini et al., 2024) | Verifier trained on both accurate and inaccurate rationale/reasoning chains | P |
| | (24) VerifierQ (Qi et al., 2024) | Trains process-level verifier using Q-Learning | P |
| | (25) PAV (Setlur et al., 2025a) | Trains based on measure of progress achieved using a step than absolute values | P |
| | (26) RL-Tango (Zha et al., 2025) | Trains a generator and verifier in tandem using RL | P |
| | (27) S2R (Ma et al., 2025) | trains ORM and PRM with RL | O, P |
| **Reasoning** | (28) Heimdall (Shi and Jin, 2025) | RL-trained verifier with pessimistic value estimates for reducing uncertainty | P |
| | (29) Think-J (Huang et al., 2025) | Fine-tunes a LLM on judgment thinking traces using RL | P |
| | (30) ThinkPRM (Khalifa et al., 2025) | proposes to train a generative PRM using a reasoning model on synthetic data | P |
| | (31) J1 (Whitehouse et al., 2025) | Fine-tunes a LRM with RL and classical SFT for verification | O, P |
| | (32) $RL^V$ (Sareen et al., 2025) | Trains a LRM on generative verification and RL objectives jointly | P |
| | (33) DyVe (Zhong et al., 2025c) | Trains a LRM as dynamic verifier through SFT on carefully curated data | P |
| | (34) FlexiVe (Zhong et al., 2025b) | Trains a LRM as dynamic verifier with RL alternating between fast and slow thinking | P |
| | (35) OREO (Wang et al., 2024a) | Trains a value function by optimizing soft Bellman equation for test-time scaling | P |
| **Symbolic** | (36) Deductive PRM (Ling et al., 2023) | Generates symbolic programs for process verification by distilling from stronger LLMs | P |
| | (37) RaLU (Li et al., 2025b) | Trains a verifier to check alignment of natural language to symbolic program | P |
| | (38) SymbCOT (Xu et al., 2024) | Translates natural language rationales to symbolic outputs for verification using logic deduction | P |
| | (39) ENVISIONS (Xu et al., 2025) | Proposes an environemnt guided neuro-symbolic self-training, self-verifying framework | P |
| | (40) START (Li et al., 2025a) | Self-distilled reasoner with external tool calling for verification | P |
| | (41) LMLP (Zhang et al., 2024b) | Learns to generate facts and logic programs over a KB for step-by-step verification | P |
| | (42) FOVER (Kamoi et al., 2025) | trains PRMs on step-level labels automatically annotated by formal verification tools | P |

Table 1: Verifier training mechanisms as applied to TTS. KB- Knowledge Base, O - Outcome, P - Process

et al., 2022; Lightman et al., 2023) if they verify the LLM reasoning process step-by-step or outcome-based - Outcome Reward Models (ORM) (Cobbe et al., 2021) if they are only concerned about correctness of final solution. However, manual annotation of step-level correctness for training PRMs is time-consuming and infeasible leading to new paradigm of synthetic reasoning path generation and annotation of them using various signals such as final ground truth answer (Chen et al., 2024a), LLM uncertainty (Ye et al., 2025) or verifier confidence (Lu et al., 2024).

While a large number of verification approaches have emerged very recently, there is a lack of clear distinction based on how they are trained, how their training data is collected and how they are employed for test-time scaling. The diversity in verifier types, synthetic data generation and training objectives for verifiers can also be understood from the lens of **asymmetry of verification**: where certain tasks are hard to solve but easy to verify or easy to solve but hard to verify (Liu et al., 2025; Qin et al., 2025).

In this survey, our goal is to present a clear and simple taxonomy for verifiers based on training recipes, objective functions and their inference time utility in test-time scaling. Note that while reward models are also used for verification purposes in pre-training, our scope is limited to verification strategies, particularly for test-time scaling. We observe based on our survey that the verifier design choices can be ascribed to task-specific challenges due to asymmetry of verification. We also present challenges with current approaches and new perspectives on future of verifier training mechanisms. We also discuss a training recipe for verifiers that could emerge from recent advances in self-play (Zhao et al., 2025a) as future directions. Details on how we collect literature for this topic can be found at Appendix A

**Comparison to adjacent surveys** While several surveys focus on Test-Time Scaling (TTS) (Zhang et al., 2025a; Chung et al., 2025; Ji et al., 2025), and reward models for LLM pre-training (Zhong et al., 2025a), they do not adequately cover wide-range of verifier types and their training mecha-

nisms in context of test-time scaling. To the best of our knowledge, our survey is the first to do this as verifiers are cornerstone of TTS and a clear taxonomy of approaches help inform their usage for real-world applications.

## 2 Notions and Concepts in Test-Time Scaling

*Test-time scaling* refers to allocating additional computational resources to language models (LLMs) during inference to improve their outputs without modifying the model weights. Test-time scaling involves three key dimensions: *when verification occurs* (at the outcome or process level), *how exploration is conducted* (in parallel, sequentially, or using hybrid strategies), and *how candidate solutions are selected or refined* (via sampling, reranking, structured search, or self-correction). A clear understanding of these dimensions is critical for robust inference-time reasoning systems.

### 2.1 Scaling Paradigms at Inference

Test-time scaling can be categorized into three paradigms: In **parallel scaling**, the model generates multiple independent outputs simultaneously, often by varying sampling temperature or prompt exemplars to induce diversity (Levy et al., 2023; Brown et al., 2024). These outputs form a candidate set $\mathcal{S} = \{s_1, \ldots, s_k\}$, from which a selection mechanism $\mathcal{V}$ identifies the final answer $s^* = \mathcal{V}(\mathcal{S})$.

**Sequential scaling**, in contrast, decomposes a problem into intermediate steps or sub-questions. Each step builds on the previous one, producing a sequence $\{sq_1, \ldots, sq_T\}$ where each $sq_t = \text{LLM}(sq_{t-1}, c_t)$ depends on the prior reasoning step and contextual information $c_t$ (Madaan et al., 2023; Du et al., 2024).

**Hybrid scaling strategies** integrate both approaches. A model may first generate multiple reasoning paths in parallel, then iteratively refine or select from them using sequential reasoning. Notable examples include beam search (Wang et al., 2024a; Snell et al., 2024), tree/graph/forest-of-thoughts (Yao et al., 2023; Besta et al., 2024; Bi et al., 2025; Wan et al., 2024; Hao et al., 2023), and constrained Monte Carlo Tree Search (MCTS) (Lin et al., 2025), where exploration and refinement are tightly coupled.

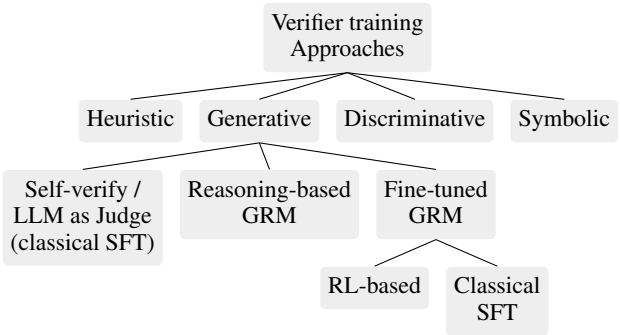


Figure 1: Categorization of verifier training approaches.

### 2.2 Verification Strategies

No matter how the candidate solutions are generated, selecting a reliable answer often requires *verification*. **Outcome verification**, also known as outcome reward modeling (ORM), evaluates whether the final answer is correct without inspecting intermediate steps. These models are relatively easy to train using weak supervision and are commonly used in reranking or filtering (Cobbe et al., 2021; Lightman et al., 2024; Zhang et al., 2024a).

In contrast, **process verification**—or process reward modeling (PRM)—assesses the reasoning path itself, step by step. This approach typically relies on annotated reasoning traces and provides more fine-grained evaluations of model behavior (Uesato et al., 2022; Zhang et al., 2025c; Zheng et al., 2025). Recent work also explores *self-verification*, where the model critiques its own reasoning path without external feedback (Weng et al., 2023; Lee et al., 2025; Kang et al., 2025).

## 3 Training Paradigms and Types of Verifiers

We start by giving an overview of the verifier training procedure as in Figure 2 that is organized along three key dimensions: (1) the *nature of the supervision* (heuristic, supervised, reinforcement learning), (2) the *type of data used* (manual annotations, synthetic rollouts, symbolic derivations), (3) the *output modality* (discriminative vs. generative). We also categorize training strategies for verifiers used in test-time scaling in Figure 1. This taxonomy not only provides a conceptual map of existing methods but also considers design trade-offs that affect performance, scalability, and robustness. Understanding the diversity and limitations of current training strategies is crucial for choosing the right approach for a given task, whether it be math reasoning, fact verification, or open-domain QA.

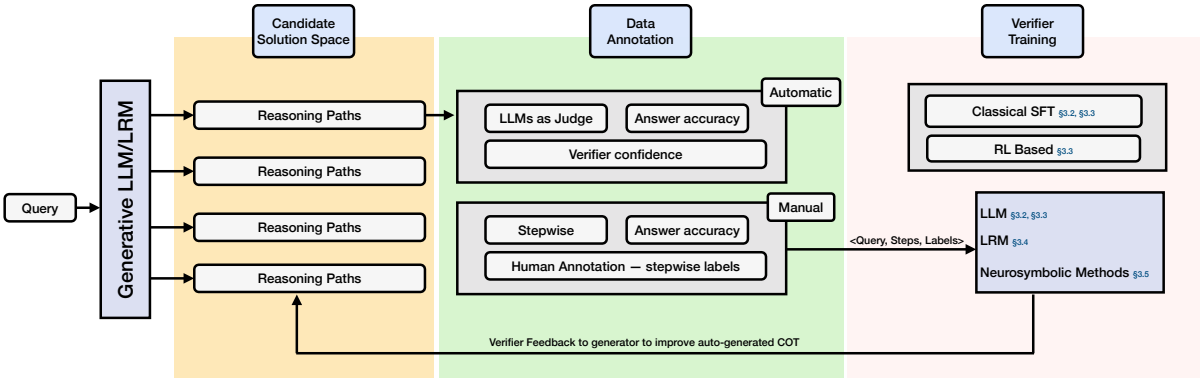


Figure 2: An overview of verifier training mechanisms

Verifier training approaches vary in how they model solution correctness: via rules, scores, rationales, or logic chains and differ in their data needs and robustness. Discriminative models are lightweight but narrow in domain; generative and reasoning-based verifiers better capture complex verification needs, while symbolic verifiers offer formal guarantees.

### 3.1 Heuristic Verifiers

Early verification strategies primarily relied on heuristics to verify final solutions such as output fluency, plausibility, adherence to format (DeepSeek-AI et al., 2025) where LLMs are trained to self-verify using these heuristics as rewards for RL-based training using Group Relative Policy Optimization (GRPO). RvLLM (Zhang et al., 2025b) designs an Expert Specification language (ESL) that can be used by experts to design domain specific rules for verification. However, rule based verification approaches are not scalable as it is time-consuming to author comprehensive rules. They also fail to capture semantic variations in expression (for example: “2 hours" and “120 minutes" imply the same) resulting in incorrect verification.

### 3.2 Discriminative Verifiers

Discriminative verifiers frame the verification task as a supervised classification problem, where models are trained to assign correctness scores to reasoning steps or final answers. These models are commonly used in tasks that are relatively **easy to verify**, where correctness can be inferred from observable outputs. A standard distinction exists between **Outcome Reward Models (ORMs)**, which evaluate only the final answer, and **Process Reward Models (PRMs)**, which assign scores to each step in the reasoning trajectory (Cobbe et al., 2021; Lightman et al., 2023; Yu et al., 2024).

PRMs are typically fine-tuned on datasets with binary step-level annotations (Uesato et al., 2022; Zhang et al., 2025c). Given a reasoning step, the model encodes the input and outputs a binary score using a classification head optimized with cross-entropy loss. Final solution quality is often estimated by aggregating the predicted scores across steps (Snell et al., 2024; Wu et al., 2024).

However, manually creating step-level labels is labor-intensive, prompting the development of automated annotation strategies. In many approaches, a reasoning step is deemed correct if a valid final answer can be generated by an LLM in a fixed number of sampling trials (e.g., via Monte Carlo rollouts) (Wang et al., 2024b; Luo et al., 2024; Chen et al., 2024a). This proxy ties the **answer accuracy** from future rollouts to the value of the current step. Subsequent steps after the first failure are typically discarded.

To avoid information loss by discarding paths, AutoCV (Lu et al., 2024) proposes a confidence-based annotation strategy. An ORM is trained on final-answer correctness, and its confidence change upon adding a new reasoning step is used to estimate that step’s value. A significant drop in confidence suggests a flawed reasoning step, which helps annotate without explicit human annotations.

Despite empirical success, these models often assume stepwise independence. Li et al. (Li and Li, 2025) identify that this independence assumption undermines reward distribution across reasoning trajectories. They reformulate step-level verification as a Markov Decision Process (MDP) and introduce a Q-value ranking objective to model dependencies between steps, enabling more consistent and context-aware feedback.

While early work focused on math and code

reasoning, more recent approaches extend discriminative verifiers to knowledge-intensive tasks. RE$^2$SEARCH++ (Xiong et al., 2025) leverages contrastive learning to distinguish correct intermediate actions from incorrect ones in open-domain QA, offering feedback not only on reasoning correctness but also on auxiliary decisions such as query formulation. VersaPRM (Zeng et al., 2025) augments the MMLU benchmark with synthetic CoT traces and counterfactual variants, expanding PRM training to general-domain tasks beyond math.

### 3.3 Generative Verifiers

Generative verifiers leverage the inherent capabilities of large language models (LLMs), including natural language generation, chain-of-thought (CoT) reasoning (Wei et al., 2022b), and instruction-following, to assess the correctness of solutions. Unlike discriminative verifiers, which produce scalar confidence scores, generative verifiers produce textual critiques or judgments. This approach not only aligns with the pretraining objective of LLMs but also enables greater interpretability, compositionality, and adaptability during inference (Wang et al., 2024c; Brown et al., 2024).

While prompting off-the-shelf LLMs as judges (Zheng et al., 2023a) offers a straightforward way to tap into these capabilities, it often underperforms compared to fine-tuned models on reasoning benchmarks. To overcome this, recent work has focused on training generative verifiers using synthetic supervision or reinforcement learning. We group these approaches into three main categories: (i) supervised fine-tuning (SFT), (ii) reinforcement learning (RL), and (iii) self-verification. These are summarized in Table 1 and Figure 1. Table 4 gives an overview of hypothesis behind some of important training recipes for verifiers.

**Supervised Fine-Tuning (SFT)** Early work in generative verification (Zhang et al., 2024a) fine-tunes LLMs using next-token prediction to jointly generate critiques and correctness assessments for solution candidates. Here, the verifier is trained on synthetic critiques generated for reasoning paths and final answers. Instead of assigning numeric scores, correctness is inferred from the model's token probabilities over natural language outputs.

Other approaches rely on synthetic supervision from stronger LLMs or tool-augmented critiques (Li et al., 2025c; Lee et al., 2025; Kang et al., 2025; Mekala et al., 2024). These are used in *self-verification* pipelines, where models are trained to evaluate their own outputs or those of a fixed generator. These approaches employ Rejection Sampling Fine-tuning (RSFT) where only synthetic trajectories leading to correct solutions are retained for SFT of verifier. SyncPL (Liang et al., 2025) alternatively proposes to augment RSFT with logical consistency based filtering of synthetic trajectories.

While simple and scalable, these methods can suffer from poor robustness and susceptibility to *reward hacking* (Ye et al., 2025), where the verifier learns spurious correlations that inflate reward without genuine correctness. To address these concerns, Ye et al. (2025) introduce **COT-entropy**, a generative verifier with uncertainty quantification. Their method explicitly penalizes overconfident predictions by modeling token-level entropy during reasoning, improving verification reliability.

**Reinforcement Learning (RL) Approaches** Reinforcement learning has emerged as a powerful paradigm for training generative verifiers, offering better alignment with preferences and improved generalization.

V-Star (Hosseini et al., 2024) builds on the StaR framework (Zelikman et al., 2022) by sampling both correct and incorrect reasoning traces and deriving preference data from correctness labels. The verifier policy is then optimized using Direct Preference Optimization (DPO), encouraging the generation of faithful reasoning steps and accurate final answers. GenRM (Mahan et al., 2024) follows a similar design, fine-tuning a generative verifier on synthetic preference data using preference-based loss functions.

VerifierQ (Qi et al., 2024) adopts a different RL objective by training the verifier using Q-learning. Here, the verifier estimates step-level Q-values that reflect the expected reward for continuing from a given reasoning step. Conservative Q-learning is used to avoid overestimation, enabling stable training. This formulation allows parallel ranking of solution candidates and supports hybrid search strategies. Rather than assigning absolute values to steps and ignoring incorrect steps like previous approaches, PAV (Setlur et al., 2025a) measures progress that can be achieved by including a step enabling more exploration when coupled with RL-based training outperforming VerifierQ and other approaches.

**Co-evolution of Generator and Verifier** Fixed-generator settings introduce a bottleneck: if the verifier is trained on poor-quality samples, it risks overfitting or reward hacking. RL-Tango (Zha et al., 2025) addresses this by jointly training a generator-verifier pair using Group Relative Policy Optimization (GRPO). The verifier is trained using outcome-level rewards to predict stepwise correctness, which in turn informs the generator's updates, closing the feedback loop. This co-evolution leads to improved solution quality and better verifier generalization. S2R (Ma et al., 2025) builds on this idea by encouraging the generator to self-verify, using both outcome- and process-level feedback to guide training. Like RL-Tango, it avoids manual step-level annotations by relying on downstream rewards.

### 3.4 Reasoning-Based Generative Verifiers

While generative verifiers capitalize on the natural generation capabilities of LLMs, they often fall short when applied to tasks that are inherently "hard to verify" — a challenge rooted in the *asymmetry of verification*: some problems are easy to solve but difficult to verify, and vice versa. Addressing this limitation has led to the rise of reasoning-based generative verifiers, which employ long-form reasoning and deliberate critique generation, leveraging recent advances in Large Reasoning Models (LRMs) (Huang et al., 2025; Khalifa et al., 2025).

ThinkPRM (Khalifa et al., 2025) is one such model that trains a process verifier (PRM) using LRMs in a generative fashion. It utilizes high-quality synthetic reasoning trajectories, generated via automated step-level labeling, to fine-tune the verifier. Despite using less annotated data, ThinkPRM demonstrates strong data efficiency and performance, outperforming models trained on over 100$\times$ more human-labeled data.

Other efforts extend this approach by refining the "LLM-as-a-judge" paradigm (Zheng et al., 2023a). For example, (Whitehouse et al., 2025; Huang et al., 2025) fine-tune LRMs to reason through and rank candidate responses based on verification-specific objectives. Unlike previous approaches requiring enormous data engineering, OREO (Wang et al., 2024a), frames verification as a byproduct of reinforcement learning. OREO jointly trains a "policy" and a "value function" within an offline RL framework using a LRM. The training optimizes a soft Bellman equation to closely approximate optimal value estimates. The resulting value function acts as an implicit verifier during test-time search, similar to verifier strategies such as RL-Tango.

While reasoning-based generative verifiers like ThinkPRM and OREO are powerful, they often do not explicitly address robustness and out-of-domain generalization. The verifier from (Shi and Jin, 2025) targets these challenges by combining long-form reasoning generation with reinforcement learning on diverse data. To further improve robustness, the paper introduces a *pessimistic verification* strategy, which prefers solutions with lower associated uncertainty. Despite their strengths, most of these methods allocate a fixed compute budget to all verification instances. However, real-world use cases often involve a mix of easy and complex verifications. Allocating equal compute to all instances may lead to inefficiencies, including **overthinking**, where unnecessary reasoning is generated for obvious cases (Chen et al., 2025).

To address this, DyVe(Zhong et al., 2025c) and FlexiVe (Zhong et al., 2025b) propose adaptive verification frameworks that tailor compute usage to the complexity of the instance. Drawing inspiration from dual-system theories of cognition (Kahneman, 2011), they alternate between "fast thinking" (direct token-level verification) and "slow thinking"(long-form deliberation). DyVe trains on both simple and complex reasoning traces from datasets like MathShepherd (Wang et al., 2024b), with stepwise judgments on whether deeper verification is needed. It fine-tunes an LRM using cross-entropy loss to produce labels for simple tasks and to generate full reasoning paths for complex ones.

In contrast, *FlexiVe* avoids the need for labeled signals about complexity. Instead, it uses *Group Relative Policy Optimization (GRPO)*(DeepSeek-AI et al., 2025) to train a process verifier. The reward function is composite: one component encourages correct verification, and another penalizes unnecessarily long outputs. This incentivizes the verifier to dynamically switch between fast and slow modes based on task difficulty.

### 3.5 Symbolic Verifiers

Despite the advances in process reward models (PRMs) trained using synthetic reasoning traces, these models often suffer from two key limitations: inaccurate step-level annotations due to reasoning drift or hallucinations, and poor generalization to out-of-distribution (OOD) tasks. Symbolic verification methods aim to address these issues by grounding reasoning in formal representations and structured logic, offering correctness guarantees,

and improving robustness. Tasks that lend themselves to formal validation, namely mathematical proofs, logical inference, and code reasoning, are thus considered **easy to verify** under this paradigm.

In this section, we categorize symbolic verification approaches. We distinguish between methods that (1) execute or emulate formal logic systems at inference, (2) translate natural language reasoning into symbolic forms for validation, and (3) augment training data with symbolic verification feedback.

**Symbolic Reasoning at Inference Time** *SymbCOT* (Xu et al., 2024) and *LMLP* (Zhang et al., 2024b) leverage symbolic reasoning to validate or derive intermediate steps during inference. SymbCOT translates chain-of-thought (CoT) responses into first-order logic expressions, which are solved step by step by an LLM acting as a logic engine. A verification module then ensures semantic consistency and logic deduction validity. LMLP generalizes this idea to logic programs over a knowledge base (KB), verifying intermediate conclusions in knowledge-intensive QA tasks.

*START* (Li et al., 2025a) builds a reasoning agent that learns to invoke symbolic tools adaptively. Hints are inserted into solution traces to prompt tool use at key reasoning junctures, and the model is fine-tuned via self-distillation. At inference time, the verifier strategically uses symbolic tools during sequential test-time scaling.

**Symbolic Supervision for Training Verifiers:** *Deductive PRM* (Ling et al., 2023) proposes using symbolic representations, called *natural programs*, which specify premises, conditions, and conclusions to train PRMs. These programs generated using strong LLMs (e.g., GPT-3.5-turbo), with verification labels are distilled into smaller models like Vicuna through supervised fine-tuning (SFT).

*FOVER* (Kamoi et al., 2025) uses formal verification tools such as Z3 and Isabelle to generate step-level annotations. Instead of verifying full proofs, the tools validate individual steps from reasoning traces generated by LLMs. This supervision signal is then used to train PRMs. Notably, FOVER achieves strong OOD generalization on reasoning benchmarks like ANLI, HANS, and MMLU.

**Neuro-Symbolic Self-Training**: *EnVISIONS* (Xu et al., 2025) departs from static annotation paradigms by interacting with an external symbolic environment. The model outputs symbolic representations (e.g., first-order logic), which are executed and scored by the environment. The resulting binary feedback is converted into soft rewards and used for iterative refinement and self-training, enabling verification without human supervision.

## 4 Benchmarks and Evaluations

We discuss different reference-based and reference-free benchmarks specifically created to evaluate process-based verifiers (Lightman et al., 2024; Zheng et al., 2025) or outcome based ones (Liu et al., 2024; Lambert et al., 2024). The benchmark design can also be mapped to *asymmetry of verification* based on the tasks represented. When constructing these benchmarks, researchers generate multiple responses for the same query. During the manual annotation or automated judgment phase, for outcome-based verifier benchmarks, annotators are required to compare final responses or intermediate ones in case of process-level benchmarks and assign scores based on preference, correctness.

**RM-Bench** (Liu et al., 2024) primarily focuses on verifying ORM models and comprises tasks from domains code, math, open-ended chat and safety. Code and math are objective and hence are *easy to verify* through unit tests and final answers, but verifying factual inaccuracies in open-ended chat and assessing whether responses are safe or not are a bit subjective and *hard to verify*. However, the existing ORM benchmarks primarily focus on preference-based evaluation.

**Verifybench** (Yan et al., 2025) bridges this gap by curating a benchmark for reference-based reward models by focusing on objective accurate judgments with respect to a reference solution when compared to pairwise assessments aligning well with real-world applications of verification models such as test-time scaling or training of reasoning models. While this benchmark focuses on moderately easy to verify tasks, authors also introduce Verifybench-Hard which comprises of **hard to verify** instances

**PRM800K** (Lightman et al., 2024) primarily focused on collecting large volume of step-by-step reasoning based verification labels for math problems to benchmark PRMs. But it lacks task diversity and mostly comprises of easy to verify math problems. **PRMBench** (Song et al., 2025) extends upon PRM800K with fine-grained error detection in PRMs. **ProcessBench** (Zheng et al., 2025) proposed to Olympiad level difficulty problems. Unlike existing synthetic benchmarks, Processbench collected fine-grained human feedback. While

math domain is generally considered *easy to verify*, Olympiad problems are more subjective as verification does not only pertain to correctness of final answer but also creativity of the solution rendering it a *hard to verify* task similar to **JudgeBench** (Tan et al., 2025).

**Evaluation Metrics**: Most of the benchmarks verify the correctness of error identification by the verifiers. The ORM benchmarks namely RM-Bench, Rewardbench employ accuracy of verifying the final output with respect to reference annotations as metric. Processbench computes step-level verification accuracy (earliest error detection) with respect to ground truth and employs F1 score to balance between being overly critical and being incapable of identifying errors. PRM800K uses the final task performance as metric to determine quality of the verification rather than the accuracy of verification process itself. Additional details are in **Appendix B with benchmark classification in Table 2** and corresponding metrics in **Table 3**.

## 5 Challenges and Future Directions

**Limited Modalities**: Current verifier approaches for test-time scaling primarily focus on textual modality. This is primarily due to significant advances in test-time scaling being currently limited to text modality (Zhang et al., 2025a; Chung et al., 2025). However, though more recently scaling approaches for visual modality have emerged, corresponding verification approaches has not advanced significantly. While there is some initial research in this area namely VisualPRM (Wang et al., 2025), a future direction could be to focus on developing principled verification methods beyond text.

**Efficiency of verifiers**: A large number of verifier approaches primarily train a LLM or a LRM as ORM or PRM as outlined in this survey in Section 3. However, this requires additional compute compounding over compute used for solution candidates generation using another LLM with test-time scaling. Hence, there is a need for efficient verification approaches. While pruning and quantization are possible approaches for reducing memory requirements, they may lead to sub-par performance. One of the interesting future directions would be to develop novel learning approaches for verifiers employing Small Language Models (SLMs). Alternatively algorithms for assembling an ensemble of expert SLMs which is dynamically determined for a task is interesting so they can achieve robustness and generalization capabilities.

**Generalization gap and Limited Benchmarks**: Many of the current state-of-the-art verifier training approaches still suffer from the out of distribution generalization gap (Zhao et al., 2025b). This also primarily stems from lack of benchmarks representative of diverse real-world tasks, as current benchmarks are majorly focused on code or math. While multi-domain data augmentation strategies (Zeng et al., 2025) have shown some promise, curating data for diverse tasks with manual annotations is time-consuming. Further augmenting them with step-level annotations manually for training verifiers becomes infeasible. While existing data augmentation approaches for training verifiers focus on generating step-level or final accuracy annotations (Zeng et al., 2025; Khalifa et al., 2025; Wang et al., 2024b; Luo et al., 2024; Lu et al., 2024), they still rely on all the queries and corresponding ground truth to be curated through manual annotation. This limits the query and task distribution that the verifier can observe limiting generalization.

**Future Directions**: We envision a generalized framework for training verifiers that would contain the following characteristics: 1) generation of synthetic data with automated annotations for training verifiers 2) produce verifiers for wide-range of natural language reasoning tasks beyond math and code and 3) produce verifiers that ideally help generalize out of distribution. To achieve 2) and 3) it is necessary for the verifier to be trained on diverse tasks and diverse queries corresponding to these tasks. Hence, 1) involves a generating queries for diverse tasks going beyond human annotated task distributions. The recent advances in reinforced self-play (Zhao et al., 2025a), and verification based self-certainty based signals (Zhao et al., 2025c), could lead to an unified framework comprising task proposer, task solver and verifier that co-evolve using specific rewards such as task diversity, step-wise correctness coupled with uncertainty quantification could be used beyond math and code.

## 6 Conclusion

This work presents a survey of verification approaches scoped to the different types and training mechanisms for verifiers when employed for for test-time scaling. We define a clear taxonomy primarily focusing verifier types , training mechanisms, objectives and categorize the literature based on this taxonomy (Table 1). Finally, we highlight challenges and future directions.

## 7 Limitations

While our survey provides a comprehensive overview of verifier mechanisms that aid in test-time scaling, due to the rapid progress in the field, several adjacent works in reward models or reward design may have been overlooked. While reward design for pre-training is an interesting topic, our survey is scoped to reward models /verifiers for test-time scaling as in this setup verifiers guide search over vast candidate space. We will maintain this survey in form of repository to enable addition of new literature as the field progresses. Additionally, we do not focus on different prompt based reasoning mechanisms that briefly touch upon self-verification by asking LLM. While we have briefly discussed some of the important works in this area, adjacent works which do not primarily contribute to new approaches for verification or works where verification is not one of the core focus areas are omitted. This is primarily because fine-tuning based or symbolic approaches have shown to be more robust and calibrated than simply asking LLMs. Covering robust mechanisms for training or employing verifiers is one of the important aspects of our survey.

## 8 Ethical Considerations and Risks

The reliance on LLM-generated reasoning paths in test-time scaling introduces potential risks of reinforcing model biases or factual errors during verifier training with automatically generated data. Although the verifier is trained to select accurate reasoning, it may inherit systematic flaws from the base LLM. However, in our survey we also particularly focus on training mechanisms and verification approaches that help improve robustness of verifiers which would help discard solution candidates with factual errors. However, the problem of hallucination in LLMs that lead to such issues is far from solved and building more robust verification mechanisms could help mitigate this issue.

Second, current landscape of verification approaches do not currently include fairness or bias mitigation across different attributes for tasks like Question Answering. While this is not the scope of this work, we believe this could be an important direction to build trustworthy systems.

## References


Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *AAAI*, pages 17682–17690.

Zhenni Bi, Kai Han, Chuanjian Liu, Yehui Tang, and Yunhe Wang. 2025. Forest-of-thought: Scaling test-time compute for enhancing LLM reasoning. In *Forty-second International Conference on Machine Learning*.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. 2024. Large language monkeys: Scaling inference compute with repeated sampling.

Guoxin Chen, Minpeng Liao, Chengxi Li, and Kai Fan. 2024a. Alphamath almost zero: Process supervision without process. In *Advances in Neural Information Processing Systems*, volume 37, pages 27689–27724. Curran Associates, Inc.

Lingjiao Chen, Jared Quincy Davis, Boris Hanin, Peter Bailis, Ion Stoica, Matei Zaharia, and James Zou. 2024b. Are more llm calls all you need? towards the scaling properties of compound ai systems. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Xingyu Chen, Jiahao Xu, Tian Liang, Zhiwei He, Jianhui Pang, Dian Yu, Linfeng Song, Qiuzhi Liu, Mengfei Zhou, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. 2025. Do NOT think that much for 2+3=? on the overthinking of long reasoning models. In *Forty-second International Conference on Machine Learning*.

Ziru Chen, Michael White, Raymond Mooney, Ali Payani, Yu Su, and Huan Sun. 2024c. When is tree search useful for llm planning? it depends on the discriminator.

Ho-Lam Chung, Teng-Yun Hsiao, Hsiao-Ying Huang, Chunerh Cho, Jian-Ren Lin, Zhang Ziwei, and Yun-Nung Chen. 2025. Revisiting test-time scaling: A survey and a diversity-aware method for efficient reasoning.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems.

Jared Quincy Davis, Boris Hanin, Lingjiao Chen, Peter Bailis, Ion Stoica, and Matei Zaharia. 2024. Networks of networks: Complexity class principles applied to compound ai systems design. *arXiv*.

DeepSeek-AI, Daya Guo, Dejian Yang, and Haowei Zhang et. al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. 2024. Improving factuality and reasoning in language models through multiagent debate. In *Forty-first International Conference on Machine Learning*.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. 2023. Reasoning with language model is planning with world model.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. 2022. Training compute-optimal large language models.

Arian Hosseini, Xingdi Yuan, Nikolay Malkin, Aaron Courville, Alessandro Sordoni, and Rishabh Agarwal. 2024. V-STar: Training verifiers for self-taught reasoners. In *First Conference on Language Modeling*.

Hui Huang, Yancheng He, Hongli Zhou, Rui Zhang, Wei Liu, Weixun Wang, Wenbo Su, Bo Zheng, and Jiaheng Liu. 2025. Think-j: Learning to think for generative llm-as-a-judge.

Yixin Ji, Juntao Li, Yang Xiang, Hai Ye, Kaixin Wu, Kai Yao, Jia Xu, Linjian Mo, and Min Zhang. 2025. A survey of test-time compute: From intuitive inference to deliberate reasoning.

Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. 2023. Llm-blender: Ensembling large language models with pairwise ranking and generative fusion. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14165–14178.

D. Kahneman. 2011. *Thinking, Fast and Slow*. Farrar, Straus and Giroux.

Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Mudit Verma, Kaya Stechly, Siddhant Bhambri, Lucas Paul Saldyt, and Anil B Murthy. 2024. Position: Llms can't plan, but can help planning in llm-modulo frameworks. In *Forty-first International Conference on Machine Learning*.

Ryo Kamoi, Yusen Zhang, Nan Zhang, Sarkar Snigdha Sarathi Das, and Rui Zhang. 2025. Training step-level reasoning verifiers with formal verification tools.

Minki Kang, Jongwon Jeong, and Jaewoong Cho. 2025. T1: Tool-integrated self-verification for test-time compute scaling in small language models.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models.

Muhammad Khalifa, Rishabh Agarwal, Lajanugen Logeswaran, Jaekyeom Kim, Hao Peng, Moontae Lee, Honglak Lee, and Lu Wang. 2025. Process reward models that think.

Takeshi Kojima, Shixiang (Shane) Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *Advances in Neural Information Processing Systems*, volume 35, pages 22199–22213.

Nathan Lambert, Valentina Pyatkin, Jacob Morrison, LJ Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, Noah A. Smith, and Hannaneh Hajishirzi. 2024. Rewardbench: Evaluating reward models for language modeling.

Hyunseok Lee, Seunghyuk Oh, Jaehyung Kim, Jinwoo Shin, and Jihoon Tack. 2025. ReVISE: Learning to refine at test-time via intrinsic self-verification. In *Forty-second International Conference on Machine Learning*.

Itay Levy, Ben Bogin, and Jonathan Berant. 2023. Diverse demonstrations improve in-context compositional generalization. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1401–1422, Toronto, Canada. Association for Computational Linguistics.

Chengpeng Li, Mingfeng Xue, Zhenru Zhang, Jiaxi Yang, Beichen Zhang, Xiang Wang, Bowen Yu, Binyuan Hui, Junyang Lin, and Dayiheng Liu. 2025a. Start: Self-taught reasoner with tools.

Cheryl Li, Tianyuan Xu, and Yiwen Guo. 2025b. Reasoning-as-logic-units: Scaling test-time reasoning in large language models through logic unit alignment.

Junyou Li, Qin Zhang, Yangbin Yu, Qiang Fu, and Deheng Ye. 2024. More agents is all you need. *Transactions on Machine Learning Research*.

Wendi Li and Yixuan Li. 2025. Process reward model with q-value rankings. In *The Thirteenth International Conference on Learning Representations*.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023a. Alpacaeval: An automatic evaluator of instruction-following models. https://github.com/tatsu-lab/alpaca_eval.

Yanyang Li, Michael Lyu, and Liwei Wang. 2025c. Learning to reason from feedback at test-time.

Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. 2023b. Making language models better reasoners with step-aware verifier. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5315–5333, Toronto, Canada. Association for Computational Linguistics.

Xiaobo Liang, Haoke Zhang, Juntao Li, Kehai Chen, Qiaoming Zhu, and Min Zhang. 2025. Generative reward modeling via synthetic criteria preference learning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 26755–26769, Vienna, Austria. Association for Computational Linguistics.

Shalev Lifshitz, Sheila A. McIlraith, and Yilun Du. 2025. Multi-agent verification: Scaling test-time compute with goal verifiers. In *ICLR 2025 Workshop on Modularity for Collaborative, Decentralized, and Continual Deep Learning*.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2024. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*.

Qingwen Lin, Boyan Xu, Zijian Li, Zhifeng Hao, Keli Zhang, and Ruichu Cai. 2025. Leveraging constrained monte carlo tree search to generate reliable long chain-of-thought for mathematical reasoning. *CoRR*, abs/2502.11169.

Zhan Ling, Yunhao Fang, Xuanlin Li, Zhiao Huang, Mingu Lee, Roland Memisevic, and Hao Su. 2023. Deductive verification of chain-of-thought reasoning. In *Advances in Neural Information Processing Systems*, volume 36, pages 36407–36433. Curran Associates, Inc.

Xiaoyuan Liu, Tian Liang, Zhiwei He, Jiahao Xu, Wenxuan Wang, Pinjia He, Zhaopeng Tu, Haitao Mi, and Dong Yu. 2025. Trust, but verify: A self-verification approach to reinforcement learning with verifiable rewards.

Yantao Liu, Zijun Yao, Rui Min, Yixin Cao, Lei Hou, and Juanzi Li. 2024. Rm-bench: Benchmarking reward models of language models with subtlety and style.

Jianqiao Lu, Zhiyang Dou, Hongru Wang, Zeyu Cao, Jianbo Dai, Yingjia Wan, Yinya Huang, and Zhijiang Guo. 2024. Autocv: Empowering reasoning with automated process labeling via confidence variation. *CoRR*, abs/2405.16802.

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Meiqi Guo, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, Jiao Sun, and Abhinav Rastogi. 2024. Improve mathematical reasoning in language models by automated process supervision.

Ruotian Ma, Peisong Wang, Cheng Liu, Xingyan Liu, Jiaqi Chen, Bang Zhang, Xin Zhou, Nan Du, and Jia Li. 2025. $S^2$R: Teaching LLMs to self-verify and self-correct via reinforcement learning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 22632–22654, Vienna, Austria. Association for Computational Linguistics.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Dakota Mahan, Duy Van Phung, Rafael Rafailov, Chase Blagden, Nathan Lile, Louis Castricato, Jan-Philipp Fränken, Chelsea Finn, and Alon Albalak. 2024. Generative reward models. *arXiv*.

Dheeraj Mekala, Jason E Weston, Jack Lanchantin, Roberta Raileanu, Maria Lomeli, Jingbo Shang, and Jane Dwivedi-Yu. 2024. TOOLVERIFIER: Generalization to new tools via self-verification. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 5026–5041, Miami, Florida, USA. Association for Computational Linguistics.

OpenAI. 2024. Openai o1 system card. https://openai.com/index/openai-o1-system-card/.

Jianing Qi, Hao Tang, and Zhigang Zhu. 2024. Verifierq: Enhancing llm test time compute with q-learning-based verifiers.

Tian Qin, David Alvarez-Melis, Samy Jelassi, and Eran Malach. 2025. To backtrack or not to backtrack: When sequential search limits model reasoning.

Paul Röttger, Hannah Kirk, Bertie Vidgen, Giuseppe Attanasio, Federico Bianchi, and Dirk Hovy. 2024. XSTest: A test suite for identifying exaggerated safety behaviours in large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 5377–5400, Mexico City, Mexico. Association for Computational Linguistics.

Kusha Sareen, Morgane M Moss, Alessandro Sordoni, Rishabh Agarwal, and Arian Hosseini. 2025. Putting the value back in rl: Better test-time scaling by unifying llm reasoners with verifiers.

Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. 2025a.

Rewarding progress: Scaling automated process verifiers for LLM reasoning. In *The Thirteenth International Conference on Learning Representations*.

Amrith Setlur, Nived Rajaraman, Sergey Levine, and Aviral Kumar. 2025b. Scaling test-time compute without verification or rl is suboptimal.

Wenlei Shi and Xing Jin. 2025. Heimdall: test-time scaling on the generative verification.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters.

Mingyang Song, Zhaochen Su, Xiaoye Qu, Jiawei Zhou, and Yu Cheng. 2025. Prmbench: A fine-grained and challenging benchmark for process-level reward models.

Benedikt Stroebl, Sayash Kapoor, and Arvind Narayanan. 2024. Inference scaling flaws: The limits of llm resampling with imperfect verifiers. *arXiv*.

Sijun Tan, Siyuan Zhuang, Kyle Montgomery, William Y. Tang, Alejandro Cuadron, Chenguang Wang, Raluca Ada Popa, and Ion Stoica. 2025. Judgebench: A benchmark for evaluating llm-based judges.

Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. 2022. Solving math word problems with process- and outcome-based feedback.

Ziyu Wan, Xidong Feng, Muning Wen, Stephen Marcus Mcaleer, Ying Wen, Weinan Zhang, and Jun Wang. 2024. AlphaZero-like tree-search can guide large language model decoding and training. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 49890–49920. PMLR.

Huaijie Wang, Shibo Hao, Hanze Dong, Shenao Zhang, Yilin Bao, Ziran Yang, and Yi Wu. 2024a. Offline reinforcement learning for llm multi-step reasoning.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024b. Math-shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9426–9439, Bangkok, Thailand. Association for Computational Linguistics.

Weiyun Wang, Zhangwei Gao, Lianjie Chen, Zhe Chen, Jinguo Zhu, Xiangyu Zhao, Yangzhou Liu, Yue Cao, Shenglong Ye, Xizhou Zhu, Lewei Lu, Haodong Duan, Yu Qiao, Jifeng Dai, and Wenhai Wang. 2025. Visualprm: An effective process reward model for multimodal reasoning.

Xinyi Wang, Lucas Caccia, Oleksiy Ostapenko, Xingdi Yuan, William Yang Wang, and Alessandro Sordoni. 2024c. Guiding language model reasoning with planning tokens. In *First Conference on Language Modeling*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

Yuxia Wang, Haonan Li, Xudong Han, Preslav Nakov, and Timothy Baldwin. 2024d. Do-not-answer: Evaluating safeguards in LLMs. In *Findings of the Association for Computational Linguistics: EACL 2024*, pages 896–911, St. Julian's, Malta. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Huai hsin Chi, F. Xia, Quoc Le, and Denny Zhou. 2022a. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022b. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Yixuan Weng, Minjun Zhu, Fei Xia, Bin Li, Shizhu He, Shengping Liu, Bin Sun, Kang Liu, and Jun Zhao. 2023. Large language models are better reasoners with self-verification. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 2550–2575, Singapore. Association for Computational Linguistics.

Chenxi Whitehouse, Tianlu Wang, Ping Yu, Xian Li, Jason Weston, Ilia Kulikov, and Swarnadeep Saha. 2025. J1: Incentivizing thinking in llm-as-a-judge via reinforcement learning.

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. 2024. Scaling inference computation: Compute-optimal inference for problem-solving with language models. In *The 4th Workshop on Mathematical Reasoning and AI at NeurIPS'24*.

Guangzhi Xiong, Qiao Jin, Xiao Wang, Yin Fang, Haolin Liu, Yifan Yang, Fangyuan Chen, Zhixing Song, Dengyu Wang, Minjia Zhang, Zhiyong Lu, and Aidong Zhang. 2025. Rag-gym: Systematic optimization of language agents for retrieval-augmented generation.

Fangzhi Xu, Qiushi Sun, Kanzhi Cheng, Jun Liu, Yu Qiao, and Zhiyong Wu. 2025. Interactive evolution: A neural-symbolic self-training framework for large language models. In *Proceedings of the*

*63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12975–12993, Vienna, Austria. Association for Computational Linguistics.

Jundong Xu, Hao Fei, Liangming Pan, Qian Liu, Mong-Li Lee, and Wynne Hsu. 2024. Faithful logical reasoning via symbolic chain-of-thought. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13326–13365, Bangkok, Thailand. Association for Computational Linguistics.

Yuchen Yan, Jin Jiang, Zhenbang Ren, Yijun Li, Xudong Cai, Yang Liu, Xin Xu, Mengdi Zhang, Jian Shao, Yongliang Shen, Jun Xiao, and Yueting Zhuang. 2025. Verifybench: Benchmarking reference-based reward systems for large language models.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models.

Zihuiwen Ye, Luckeciano Carvalho Melo, Younesse Kaddar, Phil Blunsom, Sam Staton, and Yarin Gal. 2025. Uncertainty-aware step-wise verification with generative reward models. In *ICLR Workshop: Quantify Uncertainty and Hallucination in Foundation Models: The Next Frontier in Reliable AI*.

Zhangyue Yin, Qiushi Sun, Cheng Chang, Qipeng Guo, Junqi Dai, Xuanjing Huang, and Xipeng Qiu. 2023. Exchange-of-thought: Enhancing large language model capabilities through cross-model communication. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 15135–15153.

Zhangyue Yin, Qiushi Sun, Qipeng Guo, Zhiyuan Zeng, Xiaonan Li, Tianxiang Sun, Cheng Chang, Qinyuan Cheng, Ding Wang, Xiaofeng Mou, Xipeng Qiu, and Xuanjing Huang. 2024. Aggregation of reasoning: A hierarchical framework for enhancing answer selection in large language models. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 609–625.

Fei Yu, Anningzhe Gao, and Benyou Wang. 2024. OVM, outcome-supervised value models for planning in mathematical reasoning. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 858–875, Mexico City, Mexico. Association for Computational Linguistics.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. 2022. Star: Bootstrapping reasoning with reasoning.

Thomas Zeng, Shuibai Zhang, Shutong Wu, Christian Classen, Daewon Chae, Ethan Ewer, Minjae Lee, Heeju Kim, Wonjun Kang, Jackson Kunde, Ying Fan, Jungtaek Kim, Hyung Il Koo, Kannan Ramchandran, Dimitris Papailiopoulos, and Kangwook Lee. 2025. VersaPRM: Multi-domain process reward model via synthetic reasoning data. In *Forty-second International Conference on Machine Learning*.

Zhiyuan Zeng, Qinyuan Cheng, Zhangyue Yin, Bo Wang, Shimin Li, Yunhua Zhou, Qipeng Guo, Xuanjing Huang, and Xipeng Qiu. 2024. Scaling of search and learning: A roadmap to reproduce o1 from reinforcement learning perspective. *arXiv*.

Kaiwen Zha, Zhengqi Gao, Maohao Shen, Zhang-Wei Hong, Duane S. Boning, and Dina Katabi. 2025. Rl tango: Reinforcing generator and verifier together for language reasoning.

Lunjun Zhang, Arian Hosseini, Hritik Bansal, Mehran Kazemi, Aviral Kumar, and Rishabh Agarwal. 2024a. Generative verifiers: Reward modeling as next-token prediction. In *The 4th Workshop on Mathematical Reasoning and AI at NeurIPS'24*.

Qiyuan Zhang, Fuyuan Lyu, Zexu Sun, Lei Wang, Weixu Zhang, Wenyue Hua, Haolun Wu, Zhihan Guo, Yufei Wang, Niklas Muennighoff, Irwin King, Xue Liu, and Chen Ma. 2025a. A survey on test-time scaling in large language models: What, how, where, and how well?

Yedi Zhang, Sun Yi Emma, Annabelle Lee Jia En, and Jin Song Dong. 2025b. Rvllm: Llm runtime verification with domain knowledge.

YiFan Zhang, Hanlin Zhang, Li Li, and Eric Xing. 2024b. Evaluating step-by-step reasoning through symbolic verification. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 2984–3002, Mexico City, Mexico. Association for Computational Linguistics.

Yongheng Zhang, Qiguang Chen, Jingxuan Zhou, Peng Wang, Jiasheng Si, Jin Wang, Wenpeng Lu, and Libo Qin. 2024c. Wrong-of-thought: An integrated reasoning framework with multi-perspective verification and wrong information. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 6644–6653, Miami, Florida, USA. Association for Computational Linguistics.

Zhenru Zhang, Chujie Zheng, Yangzhen Wu, Beichen Zhang, Runji Lin, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2025c. The lessons of developing process reward models in mathematical reasoning.

Andrew Zhao, Yiran Wu, Yang Yue, Tong Wu, Quentin Xu, Yang Yue, Matthieu Lin, Shenzhi Wang, Qingyun Wu, Zilong Zheng, and Gao Huang. 2025a. Absolute zero: Reinforced self-play reasoning with zero data.

Eric Zhao, Pranjal Awasthi, and Sreenivas Gollapudi. 2025b. Sample, scrutinize and scale: Effective inference-time search by scaling verification. In *Forty-second International Conference on Machine Learning*.

Xuandong Zhao, Zhewei Kang, Aosong Feng, Sergey Levine, and Dawn Song. 2025c. Learning to reason without external rewards.

Chujie Zheng, Zhenru Zhang, Beichen Zhang, Runji Lin, Keming Lu, Bowen Yu, Dayiheng Liu, Jingren Zhou, and Junyang Lin. 2025. ProcessBench: Identifying process errors in mathematical reasoning. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1009–1024, Vienna, Austria. Association for Computational Linguistics.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023a. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA. Curran Associates Inc.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023b. Judging llm-as-a-judge with mt-bench and chatbot arena.

Jialun Zhong, Wei Shen, Yanzeng Li, Songyang Gao, Hua Lu, Yicheng Chen, Yang Zhang, Wei Zhou, Jinjie Gu, and Lei Zou. 2025a. A comprehensive survey of reward models: Taxonomy, applications, challenges, and future.

Jianyuan Zhong, Zeju Li, Zhijian Xu, Xiangyu Wen, Kezhi Li, and Qiang Xu. 2025b. Solve-detect-verify: Inference-time scaling with flexible generative verifier.

Jianyuan Zhong, Zeju Li, Zhijian Xu, Xiangyu Wen, and Qiang Xu. 2025c. Dyve: Thinking fast and slow for dynamic process verification.

# A Literature Compilation

## A.1 Search Strategy

We conducted a comprehensive search on Google Scholar. We first focused on highly relevant Natural language Processing (NLP) venues such as ACL, EMNLP, NAACL, COLM and journals like TACL. Since, a lot of verifier advancements come from the ML community we also focused on venues like NeurIPS, ICLR, AAAI and ICML. Since this area is rapidly moving we also focused on arXiV pre-prints. We employed keywords like "verifiers", "test-time scaling", "self-verification", "process reward models", "outcome reward models" to retrieve relevant surveys and works. We focus on papers till march 2025 and also include some recent paper in July 2025. however, owing to rapid progress in this topic, some latest pre-prints could have been excluded inadvertently from this survey.

## A.2 Selection Strategy

We primarily focus on verification approaches that help verify the final answer (ORM) or step-by-step reasoning process for arriving at solution (PRM) in context of test-time scaling. While reward models which also help value outputs during pre-training have similar objectives they are always not used in guiding exploration or search for solutions. Hence, they are excluded from our survey as our focus is on verification approaches for test-time scaling. this filtering was done based on careful review of abstract, Introduction, Conclusion and Limitations sections of these papers. We also focused on the Methodology section of these papers to compile papers that detailed different training mechanisms for verifiers which is one fo the key focus areas of our survey. After careful review, 42 papers (Table 1) were selected which forms the foundational content of this survey.

# B Additional Benchmark and Evaluation Details

Expanding upon benchmarks discussed in Section 4, we expand upon the statistics of these benchmarks in this section. An overview of benchmarks and their categorization based on verification types and asymmetry of verification are as shown in Table 2 providing a mental model of the discussion in Section 4.

**PRM800K** is among the first large-scale, human-labeled datasets with fine-grained feedback on reasoning steps. The final dataset comprises 800,000 step-level labels across 75,000 solution traces from 12,000 problems.

The PRM800K dataset was curated in two major phases:

- **Phase 1** ($\sim 5\%$ of data): Randomly sampled solution traces generated by an LLM were labeled by human annotators.
- **Phase 2** ($\sim 95\%$ of data): An *active learning* strategy was adopted:
  1. Generate multiple candidate solutions per problem.
  2. Use the current PRM to rank the solutions.
  3. Present the most *convincing incorrect* answers to human annotators for labeling.

  This approach reduces redundant labeling and prioritizes informative, borderline cases.

**Labeling Scheme**:
Each reasoning step in a solution trace is annotated with one of three labels:

- **Positive (+)**: Step is correct and helpful.
- **Neutral (0)**: Step is ambiguous or correct but non-progressive.
- **Negative (–)**: Step contains an error or irrelevant content.

**PRMBench**: PRMBench comprises 6,216 carefully designed problems and 83,456 step-level labels, evaluating models across multiple dimensions, including simplicity, soundness, and sensitivity.

**RM-Bench** It comprises approximately 2,000–3,000 preference instances, across multiple domains: Chat, Code, Math, Safety. It was created to evaluate outcome reward models' sensitivity to subtle content changes and robustness against style bias, which traditional benchmarks overlooked for open-ended chat tasks. it also comprises objective tasks like code and math which are easy to verify.

**Verifybench** was primarily curated to benchmark reference-based verifiers. It comprises 2,000 well-balanced question-answer-completion-correctness tuples. Instructions and gold reference answers were sourced from established open datasets across diverse reasoning domains (e.g., GSM8K, MATH500, MultiArith, ProofWriter). For each instruction, multiple candidate completions were generated using both open-source and

| Verification Type | Easy to Verify | Hard to Verify |
|---|---|---|
| **Outcome-level (ORM)** | ✓ **RM-Bench** (code, math) – Objective correctness via unit tests or exact numeric answers.<br>✓ **VerifyBench** – Objective reference-based evaluation against gold answers. | ✓ **RM-Bench** (safety, open-ended QA) – Requires subjective judgment of safety and factual accuracy.<br>✓ **VerifyBench-Hard** – Contains ambiguous references and subtle factual inconsistencies. |
| **Process-level (PRM)** | ✓ **PRM800K, PRMBench** – Step-by-step math reasoning with deterministic correctness; easy to check. | ✓ **ProcessBench** – Olympiad-level math and creative proofs; requires subjective evaluation of reasoning path and novelty.<br>✓ **JudgeBench** comprises difficult datasets across knowledge, reasoning, math with challenging response pairs to verify with preference labels reflecting objective correctness. |

Table 2: Matrix mapping benchmarks to verification type (rows) and verification difficulty (columns), with reasons for categorization. Easy-to-verify tasks are typically objective (unit tests, deterministic solutions), whereas hard-to-verify tasks require subjective or nuanced evaluation (safety, creativity, subtle factuality).

proprietary LLMs (e.g., GPT-4o, Qwen) to cover a spectrum of success levels. Each completion is labeled by human annotators as correct or incorrect based on alignment with the reference answer, without preference comparisons. Verification is binary and absolute, not relative which requires that classifiers predict true/false alignment with ground truth

**ProcessBench** consists of 3,400 test cases which are high quality Olympiad math problems, with all solutions annotated with step-wise labels indicating which steps are accurate or inaccurate by multiple human experts. The expert annotation ensure the data quality and the reliability of evaluation. The annotators were instructed to identify the following types of errors for step-wise reasoning for constructing ProcessBench: (1) Mathematical errors: incorrect calculations, algebraic manipulations, or formula applications. (2) Logical errors: invalid deductions, unwarranted assumptions, or flawed reasoning steps. (3) Conceptual errors: misunderstanding or misapplication of mathematical or problem concepts. (4) Completeness errors: missing crucial conditions, constraints, or necessary justifications that affect the solution's validity.

**JudgeBench** (Tan et al., 2025), is a benchmark for evaluating LLM-based judges on challenging response pairs comprising knowledge, reasoning, math, and coding tasks. JudgeBench converts existing difficult datasets into challenging response pairs with preference labels reflecting objective correctness.

**RewardBench** comprises of instances from different task categories namely open-ended chat (easy and hard), Reasoning (code and math) and a task which tests for safety of LLMs. For open-ended chat prompts, preferred and rejected response pairs are chosen from Alpaca Eval (Li et al., 2023a) and 70 instance from MT-bench (Zheng et al., 2023b)

For the safety category, the goal is to test models' tendencies to refuse dangerous content and to avoid incorrect refusals to similar trigger words. Prompts and chosen, rejected pairs are selected from custom versions of the datasets XSTest (Röttger et al., 2024) and Do-Not-Answer (Wang et al., 2024d).

**Evaluation metrics** Expanding upon metrics discussed in Section 4, Table 3 provides a detailed overview of evaluation metrics employed for verifiers on different benchmarks with the corresponding explanation (rationale).

| Benchmark | Metrics | Rationale |
|---|---|---|
| **PRM800K** | • Step-level classification accuracy (correct/incorrect step labeling)<br>• Downstream final-answer `pass@1` / `pass@k` improvement when PRM guides inference | Provides large-scale human-labeled step correctness for math. PRMs trained on it can re-rank reasoning paths during inference, improving final task accuracy. |
| **ProcessBench** | • Earliest-error detection accuracy<br>• F1 score for identifying first incorrect step<br>• Final-task accuracy improvement when verifier guides inference | Focuses on localizing the exact failure point in reasoning chains. F1 balances false positives/negatives. Evaluates both detection quality and downstream accuracy gains when used in test-time scaling. |
| **VerifyBench, and VerifyBench-Hard** | • Accuracy of binary correct/incorrect judgment vs. reference answer | Evaluates reference-based correctness judgments. The "Hard" split contains subtle or ambiguous examples to test robustness. |
| **RM-Bench** | • Pairwise accuracy (chosen $>$ rejected)<br>• Accuracy breakdown by difficulty (easy, medium, hard) | Evaluates how well outcome reward models rank human-preferred completions over rejected ones, with breakdown by difficulty. |
| **JudgeBench** | pairwise accuracy | JudgeBench leverages a novel pipeline for converting existing difficult datasets into challenging response pairs with preference labels reflecting objective correctness. |
| **RewardBench** | • Pairwise accuracy across prompt–chosen–rejected triples<br>• Domain-specific accuracy breakdown (safety, reasoning, chat, etc.) | Measures preference-based ranking performance across diverse domains; tests cross-domain generalization of reward models. |

Table 3: Evaluation metrics for major verifier benchmarks used in test-time scaling.

| Category | Approach | Hypothesis | Training Recipe |
|---|---|---|---|
| **Heuristic** | (1) Deepseek-R1 (DeepSeek-AI et al., 2025)<br>(2) RvLLM (Zhang et al., 2025b) | Domain specific rules for verification leads to more deterministic outcomes | -<br>- |
| **Generative**<br>- Classical SFT | (3) Generative Verifier (Zhang et al., 2024a)<br>(4) SyncPL (Liang et al., 2025) | Trains a generative process verifier using criteria based data generation | Employs SFT to fine-tune a LLM to generate critiques and correctness label for solution candidates |
| | (5) UQ-PRM (Ye et al., 2025) | Uncertainty quantification of Process verifiers leads to more robust reasoning in LLMs | Stepwise Uncertainty Quantification is included as part of training process for verifiers. |
| | (6) WoT (Zhang et al., 2024c)<br>(7) Multi-Agent Verifiers (Lifshitz et al., 2025) | Verification process is not uni-dimensional and hence the verifier must cover diverse perspectives of this process | Scaling verifiers to cover diverse task-related aspects that need to be verifier |
| - Self-verification (classical SFT) | (8) ReVISE (Lee et al., 2025) | Traiining to self-verify through synthetic data | Training using imitation learning on |
| | (9) Self-Reflection Feedback (Li et al., 2025c) | and external feedback leads to iterative self-improvement and efficient verification | iterative self-reflection based synthetic data |
| | (10) ToolVerifier (Mekala et al., 2024)<br>(11) T1 (Kang et al., 2025) | Self-verification through tool use or data from tool use leads to more robust verification. | SFT of LLM on data from tool use |
| - RL-based | (12) GenRM (Mahan et al., 2024) | Verifier training through preference optimization | Trains generative verifiers using preference optimization |
| | (13) V-STaR (Hosseini et al., 2024) | using automatically curated data or from human feedback offers better alignment to preferred responses. | like Direct Preference Optimization (DPO) |
| | (14) RL-Tango (Zha et al., 2025)<br>(15) S2R (Ma et al., 2025) | Jointly training Generator and Verifier together is better than training a verifier with fixed generator | RL-based Co-evolution of Generator and Verifier |
| **Reasoning** | (16) Heimdall (Shi and Jin, 2025)<br>(17) Think-J (Huang et al., 2025)<br>(18) ThinkPRM (Khalifa et al., 2025)<br>(19) J1 (Whitehouse et al., 2025) | Training a reasoning model on joint objective of reasoning or critique generation and correctness prediction renders it an effective judge for verification tasks | Trains a Large Reasoning Model on generative and solution correctness prediction tasks to serve as judge |
| | (20) $RL^V$ (Sareen et al., 2025) | Scaling generative verifier training by employing | Trains a LRM on generative verification and RL objectives jointly |
| | (21) OREO (Wang et al., 2024a) | reasoning models with RL objectives enables bettwe exploration of solution space than SFT based imitation learning | Trains a value function by optimizing soft Bellman equation for test-time scaling |
| - Adaptive | (22) DyVe (Zhong et al., 2025c)<br>(23) FlexiVe (Zhong et al., 2025b) | Training an adaptive verifier through imitation learning (SFT) or RL to alternate between fast & slow thinking leads to efficient TTS | SFT on curated data<br>RL based adaptive verifier. |
| **Symbolic** | (24) RaLU (Li et al., 2025b)<br>(25) SymbCOT (Xu et al., 2024) | Symbolic reasoning through program generation aids in more structured LLM-based verification. | Trains to generate accurate symbolic programs to aid in verification |
| | (26) START (Li et al., 2025a) | Self-distillation for learning to use symbolic tools | Trains a reasoning agent to invoke symbolic tools adaptively using instruction tuning through self-ditillation. |
| | (27) LMLP (Zhang et al., 2024b) | for step-wise verification improves factuality of generated responses | Trains to use programs that can be run over KB for step-wise verification. |
| | (28) ENVISIONS (Xu et al., 2025) | Neruo-symbolic self-training than distilling from stronger LLMs helps overcome the scarcity of symbolic data, and improves the proficiency of LLMs in processing symbolic language | Self-training framework for verifiers using interactive feedback from symbolic environment than static annotations |
| | (29) Deductive PRM (Ling et al., 2023) | Verifier trained through Distilling symbolic program generation capabilities | Symbolic supervision for training verifiers |
| | (30) FOVER (Kamoi et al., 2025) | & verification labels from formal verification tools leads to deterministic and accurate verification | |

Table 4: Summary of verifier training mechanisms as applied to test-time scaling. KB- Knowledge Base,