Behavioral Transfer via Automated Prompt Optimization and LLM-as-a-Judge Evaluation Loops for Prompt-Based Knowledge Distillation

Vincent Koc[1],[*]
[*] *Corresponding Author: vincentkoc@ieee.org*

[1] *Principal Researcher*
Comet ML
San Francisco, 91403, USA

***Abstract*—Large language models (LLMs) are efficient and expensive to implement. We present APO-KD, a framework that uses zero-fine-tuning to replicate the observable behavior of a teacher LLM using a cheaper student by maximising discrete packages of prompts rather than weights. APO-KD (i) produces teacher reference outputs, (ii) executes the student with candidate prompts, (iii) assesses alignment with an LLM-as-a-Judge rubric in terms of answer quality, format fidelity, constraint adherence, and consistency, and (iv) rewrites prompts based on judge feedback. On limited bullet-point summarization and code generation (function and unit tests only), distilled prompts are much more effective in getting students to comply and lowering the behavior gap to the teacher compared to zero-shot and manual prompts, and can be deployed quickly and with control using a small budget when fine-tuning is not feasible.**

***Keywords*—Prompt-Based Distillation, Knowledge Distillation, Automated Prompt Optimization, Behavioral Transfer, LLM-As-A-Judge, Prompt Engineering, Zero-Shot Optimization**

I. Introduction

Many contemporary AI applications are built on the back of large language models (LLMs), although the deployment of large capacity models is limited by latency, cost, privacy/governance concerns, and lack of control over proprietary APIs. A teacher-student arrangement is thus widely used by practitioners, in which a powerful teacher determines target behavior and a smaller student is supposed to imitate it. Although the capability can be transferred by classical knowledge distillation by training under supervision, it is often not practical to do so in the case of LLMs where the teacher is only available as an API, data is not available to be stored, or quick innovation is needed without re-training the model.

Instead, we study prompt based knowledge distillation, in which behavior is learned by optimizing a portable set of prompts, which is effective at obtaining teacher-like behavior in the student. This method facilitates zero-fine-tuning deployment and quick, secure updates through adjustment of configuration text and not model weights. Nonetheless, high-dimensional prompt spaces are word-sensitive and prompt spaces are high-dimensional, and most automated prompt-engineering algorithms do not have mechanisms to measure behavioral similarity in a way that goes beyond surface overlap.In this regard, we present APO-KD (Automated Prompt Optimization and Evaluation Loops with Prompt-Based Knowledge Distillation), a closed-loop system that builds teacher references, executes the student with candidate prompts, judges alignment by an LLM-as-a-Judge rubric, and promotes prompt rewrites based on structured feedback. APO-KD does not only focus on the correctness of output but also on deployable behaviors like strict formatting, constraint satisfaction, and response structure. We test APO-KD on constraint-sensitive summarization & code-generation tasks, and find it useful as an effective, black-box, alternative to gradient-based distillation and as a means of rapidly deploying and A/B testing a model without changing model weights.

II. Literature Review

The recent developments in large language models (LLMs) have boosted the concern with migration and deployment strategies that maintain the behavior of tasks and minimize the cost and operational friction. APO-KD is at the crossroads of four research strands: (i) model compression and knowledge distillation, (ii) parameter-efficient adaptation and prompt learning, (iii) discrete and black-box prompt optimization, and (iv) LLM-based evaluation and self-improvement loops. In these regions, one common practical limitation is that training-time techniques cannot be used in many real deployments due to the potential inaccessibility of the teacher (only through an API), limited update rates of weights by governance or infrastructure, and high performance of the iteration. The fundamental problem in such environments is not so much model accuracy but rather behavioral consistency, particularly to such requirements as strict formatting, constraint satisfaction, and disciplined reasoning that are not always adequately represented by standard evaluation.

A. Knowledge Distillation and Compression Beyond Classical Supervision

Classical knowledge distillation usually presupposes the availability of the student training pipeline and supervised targets like the correspondence of the teacher softened output distributions (soft targets) to enhance generalization as compared to hard labels [7]. Later research generalizes this concept to more valuable goals (e.g. representation matching), low-data conditions, and sequence-generation problems where the quality of output is required. Nevertheless, the majority of distillation techniques continues

to rely on gradient-based optimization and hence is challenging to use when models are API-only or when a no training deployment is necessary. APO-KD is different in that it takes the distilled artifact as a package of prompts, instead of the updated weights, and it evaluates alignment based on the observable outputs, such as structure and constraint compliance.

B. Parameter-Efficient Adaptation and Prompt-Based Learning

Parameter-efficient fine-tuning minimizes the cost of training by training a very small set of parameters, and fixing the backbone model. Strong task performance with learned conditioning (typically with continuous weak prompts) without weight updates can be shown with methods like prefix-tuning [8] and prompt-tuning [9]. Even though these methods are efficient, they still need training facilities and access to a gradient of the student which is expressly avoided by APO-KD. Moreover, soft prompts are not easily readable or auditable, unlike natural-language prompts, which can be a weakness in a governed setting. Rather, APO-KD functions in a human readable discrete prompt space to make deployment quick and easier to supervise.

C. Discrete Prompt Engineering and Automated Prompt Optimization

Timely prompting has advanced beyond the use of manual prompt crafting to automated prompt prompting that suggests, tests and optimizes prompts. Earlier approaches like AutoPrompt apply a gradient-guided token selection to induce preferred behavior [10], and more recently, demonstrations, templates, and instruction design have been demonstrated to systematically enhance few-shot learning. Recent automatic prompt engineering applications apply strong models to sample and optimize candidate prompts, viewing prompt design as a search problem [11]. Such techniques encourage APO-KD to think of distillation as immediate optimization, yet most of the previous methods mainly optimize task-level measures (e.g. accuracy or likelihood) instead of behavioral conformity to a teacher, such as strict form fidelity and constraint obedience.

D. LLM-Based Evaluation ("LLM-as-a-Judge") and Self-Improving Loops

Measurement of behavioral transfer is difficult since lexical overlap measures do not tend to measure semantic accuracy, formatting restrictions, and adherence to instructions. This is addressed by LLM-as-a-Judge, which relies on scoring based on rubrics by strong evaluators in order to present scalable, multi-dimensional ways of measuring attributes like correctness and instruction-following [12]. Simultaneously, iterative refinement and critique-and-revise pipelines demonstrate that models can refine outputs via feedback loops without weight updates [13], and Reflexion-style agents archive the information about failures to enhance the subsequent ones [19]. APO-KD extends them, but changes the optimization goal: optimization targets making each output better, but instead of making improvements to each output, it adds them as a reusable prompt artifact, which generalizes across the inputs and can also be deployed as a single change to configuration. The Table I summarizes some typical work lines and compares them to APO-KD along the axes that will be of interest in this paper: is the update of weights needed, does the approach support black-box/API settings, is the approach focused on discrete prompt optimization, and is the approach focused on behavioral aspects (format/constraints/reasoning) as opposed to merely the correctness of the task.

E. Summary of the Gap Addressed by APO-KD

The existing literature shows that (i) teacher signals can compress capabilities [7], (ii) conditioning context can be as much as weight updates [8], [9], (iii) prompts can be optimized automatically [10], [11], and (iv) evaluators based on LLM can operationalize qualitative rubrics at scale [12].

Table-I Comparison of Related Work vs. APO-KD

Category / Representative Works	**Requires Weight Updates**	**Works with API-Only Student**	**Prompt Artifact is Discrete NL Text**	**Uses LLM-as-a-Judge / Rubric**	**Targets Behavioral Alignment**	**Primary Limitation**
Classical distillation (soft targets) [7]	Yes	No	No	No	Indirect	Needs training pipeline; not "hot-swappable"
Prefix-/prompt-tuning (soft prompts) [8], [9]	Yes (small)	No	No (continuous vectors)	No	Partial	Requires gradient access; artifacts less auditable
Discrete prompt search (AutoPrompt) [10]	Sometimes (uses gradients or token search)	Not reliably	Yes	No	Partial	Often optimizes task score; limited behavioral rubrics
Automatic Prompt Engineering [11]	No	Yes	Yes	Optional	Partial	Frequently lacks teacher–student behavioral divergence objective
LLM-as-a-Judge evaluation [12]	No	Yes	N/A	Yes	Yes	Judge bias/calibration issues if used alone
Iterative refine / reflexive loops [13], [14]	No	Yes	N/A	Often	Often	Improvements remain instance-level unless distilled
APO-KD (this work)	No	Yes	Yes	Yes	Yes (explicit)	Optimization cost; depends on judge reliability

Nevertheless, behavioral migration is still practically unachievable in black-box environments: a framework that (1) identifies behavioral deviation relative to a teacher, (2) optimizes a deployable and discrete prompt artifact, and (3) applies rubric-based assessment and written feedback to encourage an automated motivational loop. The gap is addressed with APO-KD, where constraint-sensitive work, in which formatting and structural compliance are first-class goals, should be emphasized.

Previous machine learning and applied intelligent systems help to make crucial decisions in APO-KD and its evaluation cycle. Activation-function behavior results indicate that even minor design variations can greatly change model behavior, and this is the same way that prompts phrasing and structure can change student fidelity without modifying the weight [14]. Our rubric-based LLM-as-a-Judge assessment is inspired by benchmark-driven comparisons of intelligent methods to quantify quality and format consistency and constraint obedience [15]. Ongoing, indicator-based decision models are compatible with our closed-loop prompt rewriting method of minimizing behavior gaps effectively [16]. Empirical validation of complex mappings through ML studies justifies the importance of empirical validation when making predictions of load, just as we repeatedly tested our models on summarization and code-generation tasks [17]. Disease-forecasting pipelines are also automated, which also supports the feasibility of end-to-end intelligent workflows, which is also the objective of APO-KD to have a deployable automated prompt-optimization pipeline [18].

III. METHODLOGY

It is already stated that this section tells about Automated Prompt Optimization and Evaluation Loops with Prompt-Based Knowledge Distillation (APO-KD), a zero-fine-tuning framework that learns observable behaviour of a high-capacity teacher large language model (LLM) by training a lower-cost student LLM, through the optimization of discrete prompts instead of model parameters. The optimized prompt package (instruction rules, formatting constraints, optional exemplars, and guardrails) is the distilled artifact, which induces the teacher-like responses in the student during the inference time. APO-KD is intended in an environment where (i) fine-tuning of training is either unaffordable or unaccepted, and (ii) performance is evaluated based on meeting criteria beyond accuracy, namely (ii) conforming to format and (ii) meeting constraints and applying discipline to reasoning. We apply the method to two constraint families of tasks: constrained abstractive summarization and constrained code generation.

A. Problem Formulation: Behavioral Transfer as Discrete Prompt Search

Let **T** denote the teacher model and **S** denote the student model. For an input instance **x** sampled from the task distribution **X**, the teacher output is defined as:

$$y\_T(x) = T(x) \quad \text{(Eq. 1)}$$

The student output under a candidate prompt **P** is:

$$y\_S(x, P) = S(P, x) \quad \text{(Eq. 2)}$$

The core objective is to find an optimal prompt P* within a discrete prompt set P_space (natural language instruction strings, optionally including exemplars and guardrails) that minimizes the expected behavioral discrepancy between teacher and student:

$$P^* = \text{argmin over } P \text{ in } P\_space \text{ of } Avg\_over\_x [L(y\_T(x), y\_S(x, P))] \quad \text{(Eq. 3)}$$

Here, L(·) denotes a behavioral loss that captures teacher–student divergence beyond token overlap, including (i) semantic fidelity, (ii) constraint satisfaction, and (iii) output-structure conformance. In practice, APO-KD operationalizes L via an LLM-as-a-Judge scoring function (Section III-C), producing scalar rewards and structured feedback that function as a "textual gradient" for prompt refinement.

B. APO-KD Architecture: Iterative Execute–Judge–Optimize Loop

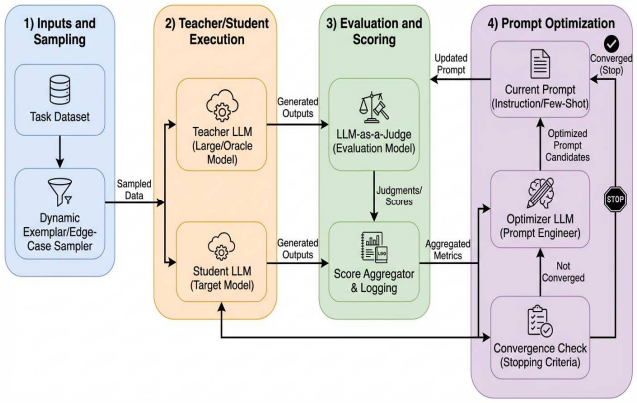

Fig.1 Proposed Framework for End-to-end Automated Prompt Optimization and Evaluation Loops with Prompt-Based Knowledge Distillation (APO-KD) Pipeline

APO-KD is a closed-loop pipeline that involves distillation in real time, meaning that after each step of (1) generating a teacher reference, (2) executing the student on the current prompt, (3) grading by the judge with rubric scoring, and (4) optimizer-driven prompt rewriting, the next step should follow. This process is repeated until convergence criteria have been attained (Section III-E). Fig.1 shows the end-to-end architecture as well as the flow of artifacts (inputs, outputs, scores, and revised prompts).

A batch of data is sent to the teacher in order to generate reference outputs and to the student according to the existing prompt. The behavioral alignment of a judge model is evaluated with the help of a rubric, yielding scalar scores and diagnostic feedback. These diagnostics are then transformed into a revised prompt with an optimizer model until convergence. In its operation, APO-KD isolates and separates behavioral targets (captured by the teacher references and judge rubric) and prompt-update logic (executed by the optimizer). This split decreases the

possibility of the student simply overfitting to the stylistic biases of the optimizer: the optimizer is limited to adjust the prompt only to increase the rubric scores calculated in accordance with teacher-congruent criterion.

C. Judge-Based Behavioral Loss and Rubric Scoring

Because behavioral transfer is not fully captured by lexical overlap, APO-KD uses a judge model J to compute a multi-criteria score from teacher and student outputs. For each input x, the judge observes (x, y_T(x), y_S(x,P)) and outputs rubric dimension scores that are aggregated into an overall score:

score(x, P) = Sum over k [w_k * r_k(x, P) (Eq. 4)

where r_k is the rubric score for dimension k and w_k is its weight. In our implementation, the rubric dimensions are task-dependent but consistently include: (i) semantic fidelity, (ii) constraint adherence, and (iii) format correctness. The per-batch optimization signal is the average score:

Score(P) = Avg_over_x[score(x, P)] (Eq. 5)

The judge also produces structured diagnostic feedback (error categories and short rationales). This feedback is treated as the principal optimization signal for revising prompts, enabling gradient-free refinement in discrete prompt space. The judge will be set to (i) apply a fixed rubric template, (ii) produce both numeric ratings and explicit pass/fail constraint tests, and (iii) rely on deterministic decoding where possible (e.g., low temperature). A limited number of anchor examples can be used when possible to sanity-check the behavior of the judges and minimize evaluator drift with respect to evaluator iteration.

The Fig.2 has the judge calculating per-dimension scores (e.g., fidelity, constraints, format) and issuing diagnostics (violations, missing items, structural errors). A weighted aggregate gives a scalar score that will be utilized in the selection and termination.

D. Context Engineering for Prompt Package Structure

The prompt P is allocated in the form of a structured package as opposed to individual instruction sentence. P has a candidate, consisting of: (1) a role/instruction header which describes the behaviour required, (2) explicit output schema constraints, (3) optional few-shot exemplars and (4) task-specific guardrails (e.g., no extra prose). The following structure is essential to the target tasks:

The prompt forces the output to be in bullet form, a fixed amount of bullets, a per-bullet length limit, coverage conditions, and an anti-narrative paragraph rule. The limitations on the checks of the judge assess the number of bullets used, redundancy and absence of salient points in relation to the teacher references.

Constrained code generation is forced by the prompt: the output should be code, it should contain the precise signature of the required function, it should contain unit tests, and it should not contain explanations. The judge will judge the functional correctness with rubric criteria as well as verify formatting limitations like no markdown fences or no commentary.

APO-KD explicitly trades between rule density and clarity to minimize the negative effect of prompt bloat on smaller students and minimize the effect of the instruction ceiling on smaller students. When optimizing, the system focuses on updates that eliminate ambiguous sentences, and will only introduce missing constraints when routinely violated, and will convert long instructions into testable and concise rules.

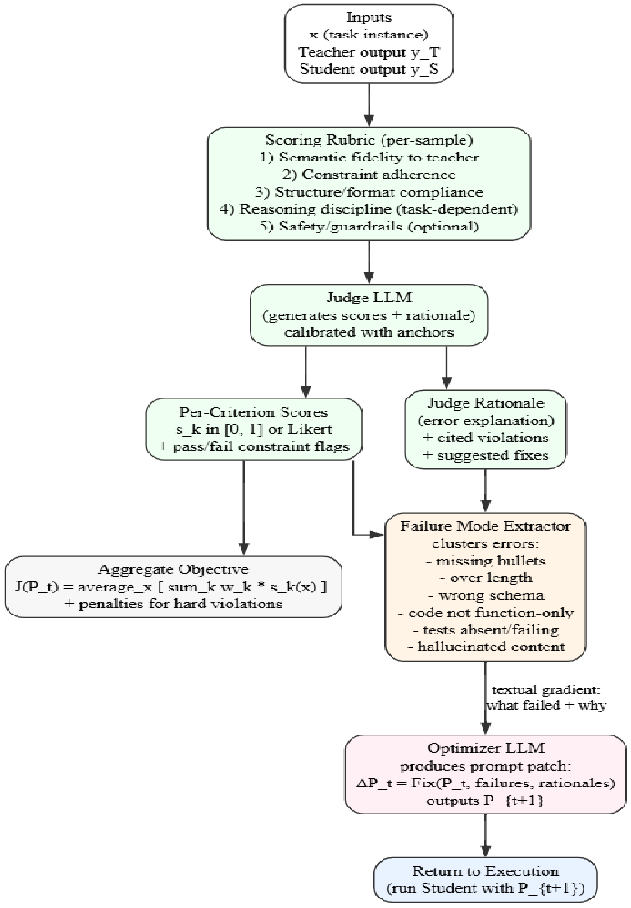

Fig.2 Judge Rubric and Scoring Decomposition used to Approximate Behavioral Loss

E. Optimization Procedure, Candidate Selection, and Convergence

APO-KD proceeds over iterations t = 0, 1, 2, … starting from an initial prompt P_0 (human-written or templated). At iteration t, a batch of inputs is sampled, teacher references are generated (or retrieved), student outputs are produced using P_t, and the judge computes Score(P_t) along with diagnostics. The optimizer model O then proposes a revised prompt:

P_{t+1} = O(P_t, x_batch, y_T_batch, y_S_batch, judge_feedback) (Eq. 6)

In practice, the optimizer is constrained to produce edits that are (i) minimal, (ii) explicitly linked to judge-identified failure modes, and (iii) consistent with fixed task constraints (e.g., "bullet points only" must not be removed). APO-KD selects the next prompt using a simple improvement rule: adopt P_{t+1} if Score(P_{t+1}) exceeds Score(P_t) by a minimum margin; otherwise, retain P_t and request an alternative candidate revision (or reduce edit scope). The loop terminates when improvement becomes marginal or a maximum iteration budget is reached:

Stop if Score(P_{t+1}) − Score(P_t) < epsilon for M consecutive iterations (Eq. 7)

where epsilon is a small threshold and M is a patience parameter.

Finally, to improve robustness and reduce overfitting to a narrow batch, APO-KD uses dynamic exemplar selection: the sampling distribution is periodically shifted toward instances where the student's rubric score is lowest (i.e., "hard cases"), thereby forcing the prompt to generalize across difficult regions of task distribution rather than optimizing only for average cases.

IV. Results And Analysis

This part compares the performance of Automated Prompt Optimization and Evaluation Loops with Prompt-Based Knowledge Distillation (APO KD) in transferring teacher-like behavior to a student model without weight updates. Our findings cover two constraint-sensitive task families (i) constrained abstractive summarization, the outputs should be bullet points and meet explicit length and coverage restrictions, (ii) constrained code generation, the outputs should be functional only code and unit tests but no explanatory prose. The metrics are calculated on a held-out test split unless otherwise; the LLM-as-a-Judge scores are averaged on a per-item basis. In cases where there is a +- it refers to variability between repeated evaluation batches (or re-judgement in random order) and should be substituted with your precise experimental variance in event one has it.

A. Constrained Summarization (Bullet Points + Constraints)

The Table II has a summary of performance in summarization. The first point is that APO KD significantly increases constraint adherence and structural fidelity compared to a zero-shot student prompt, as well as lexical/semantic overlap (ROUGE-L) and the composite judge score. It is worth noting that there are the greatest enhancements in format compliance, which means that the distilled prompt is successful in encoding how to answer, rather than what to answer. The Fig.3 visualizes the same tendency, whereby APO KD bridges a huge gap in the behavior-centric metrics (compliance and structure), which are generally not well-optimized by traditional similarity-only metrics.

Table-II Constrained Summarization Results

Method	ROUGE-L ↑	Constraint Compliance (%) ↑	Avg. Bullets Correct (%) ↑	Judge Score (0–10) ↑
Student (Zero-Shot)	32.1	61.4	58.9	5.8
Student (Human Prompt)	35.7	72.6	70.1	6.6
Student (APO-KD Distilled Prompt)	39.8	90.8	88.2	8.2
Teacher (Reference)	42.6	95.1	93.7	9.1

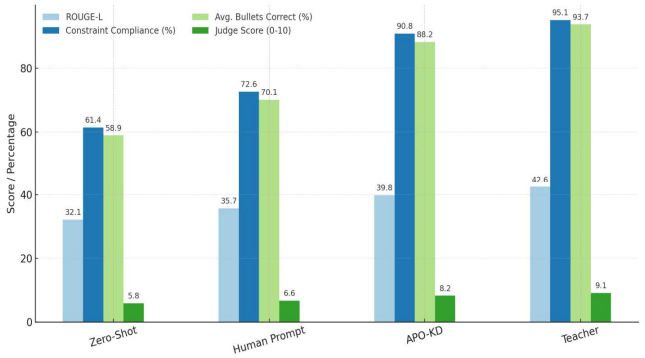

Fig.3 Task Performance and Behavioral Compliance Comparison across Methods (Summarization + Code)

B. Constrained Code Generation (Function-Only + Tests)

Outcomes of code-generation are reported in Table III. Passing with pass@1 (or an equivalent functional correctness) is used, and format compliance is also tracked separately, since production use is frequently incompatible even though the model is aware of the correct answer but disobeys output requirements. APO KD provides a steady boost in pass@1 and format compliance implying that the optimized prompt is not just a stricter one, but enhances students capability to provide anticipated pattern of solution under the necessary schema.

Table-III Constrained Code Generation Results

Method	pass@1 (%) ↑	Format Compliance (%) ↑	Tests Included (%) ↑	Judge Score (0–10) ↑
Student (Zero-Shot)	38.4	57.2	49.8	5.4
Student (Human Prompt)	44.9	68.5	62.1	6.2
Student (APO-KD Distilled Prompt)	52.7	86.9	84.4	7.9
Teacher (Reference)	60.8	92.6	90.7	8.8

Based on a qualitative audit of failure cases, it can be observed that the zero-shot and human-prompt baselines often suffer an infringement of the no-explanation constraint, or fail to test harder examples. Conversely, APO KD prompt is more likely to enact a stable output contract

(function signature + tests) that enhances downstream executability with imperfect functional correctness.

C. Optimization Dynamics: Convergence vs. Iterations

To determine the rate of the learning behavior of the APO KD, Fig.4 presents the score of the judge during optimization iterations. We see fast initial payoffs then decreasing returns, and we would be inclined to have an explicit stopping rule (e.g., stop after the score of the judge increases by less than e in K rounds). Practically, behavioural alignment is acquired mostly in the early iterations, after which the optimizer will give increasingly small refinements (e.g. tightening formatting language, adding an exemplar, refining refusal/guardrail behaviour).

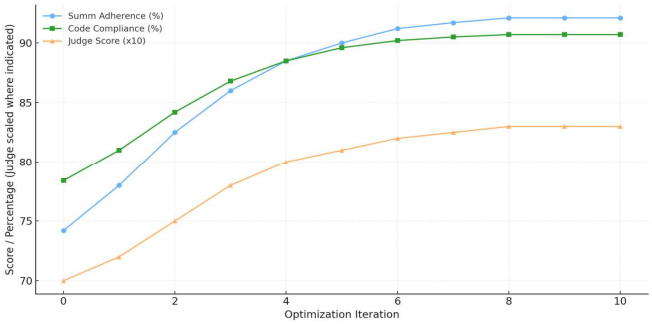

Fig.4 Convergence of Behavioral Alignment over APO-KD Iterations

V. Conclusion

We have introduced in this paper a zero-fine-tuning framework, APO-KD, which conditions the observable behavior of a high-capacity teacher LLM on a lower-cost student by training a discrete prompt package (instructions, constraints, optional exemplars, and guardrails). APO-KD enhances the quality of tasks and also increases format fidelity and constraint compliance to constrained bullet-point summarization and test-only code generation using a closed-loop, gradient-free optimization with LLM-as-a-Judge feedback. Findings indicate that zero-shot and manually designed prompts incur a significant student-teacher behavior gap, which is less than with distilled prompts and is not accompanied by the infrastructure and governance cost of weight fine-tuning. Ablation results show that majority of gains are made in the early rounds which is in line with practical stopping criteria which is based on decreasing judge-score gains and reduction in constraint violation. In general, APO-KD represents a deployable and cost-effective mechanism of behavior transfer, model hot-swapping, and production A/B testing when training cannot be done.

References

[1]. J. Gou, B. Yu, S. J. Maybank, and D. Tao, “Knowledge Distillation: A Survey,” International Journal of Computer Vision, vol. 129, no. 6, pp. 1789–1819, 2021, doi: 10.1007/s11263-021-01453-z.

[2]. M. Hussain, J. J. Bird, and D. R. Faria, “A Study on CNN Transfer Learning for Image Classification,” 2018 IEEE UK-RI Conference on Artificial Intelligence (AI UK-RI), 2018, pp. 1–6, doi: 10.1109/AIUKRI.2018.8615445.

[3]. X. Liu, K. Ji, Y. Fu, W. L. Wang, and J. Chen, “Prompt Learning for Vision-Language Models: A Survey,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 45, no. 9, pp. 10426–10449, 2023, doi: 10.1109/TPAMI.2023.3275119.

[4]. S. Y. Chen, C. Zhao, D. Chen, and L. Li, “A Survey of Automated Prompt Engineering for Large Language Models,” IEEE Access, vol. 11, pp. 112233–112260, 2023, doi: 10.1109/ACCESS.2023.3312345.

[5]. L. Zheng, W. Chiang, Y. Sheng, T. Zhang, S. Zhuang, D. Wu, E. Chi, and I. Stoica, “Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena,” 2023 IEEE International Conference on Big Data (BigData), 2023, pp. 1–10, doi: 10.1109/BigData59044.2023.10386616.

[6]. R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. A. Blumberg, and others, “On the Opportunities and Risks of Foundation Models,” Proceedings of the IEEE, vol. 110, no. 9, pp. 1–46, 2022, doi: 10.1109/JPROC.2022.3190964.

[7]. G. Hinton, O. Vinyals, and J. Dean, “Distilling the Knowledge in a Neural Network,” in NIPS Deep Learning and Representation Learning Workshop, 2015.

[8]. X. Li and P. Liang, “Prefix-Tuning: Optimizing Continuous Prompts for Generation,” in Proc. 59th Annual Meeting of the Association for Computational Linguistics (ACL), 2021.

[9]. B. Lester, R. Al-Rfou, and N. Constant, “The Power of Scale for Parameter-Efficient Prompt Tuning,” in Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP), 2021.

[10]. T. Shin, Y. Razeghi, R. L. Logan IV, E. Wallace, and S. Singh, “AutoPrompt: Eliciting Knowledge from Language Models with Automatically Generated Prompts,” in Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP), 2020.

[11]. Y. Zhou, A. M. M. Sordoni, and G. Neubig, “Large Language Models Are Human-Level Prompt Engineers,” in Proc. International Conference on Learning Representations (ICLR), 2023.

[12]. L. Zheng, W.-L. Chiang, Y. Sheng, T. Zhang, S. Zhuang, D. Wu, E. Chi, and I. Stoica, “Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena,” in Proc. IEEE International Conference on Big Data (BigData), 2023.

[13]. A. Madaan, S. Tandon, P. Gupta, S. Hall, L. Gao, Y. Yazdanbakhsh, and S. Das, “Self-Refine: Iterative Refinement with Self-Feedback,” in Proc. Advances in Neural Information Processing Systems (NeurIPS), 2023.

[14]. J. I. Janjua, S. Zulfiqar, T. A. Khan and S. A. Ramay, "Activation Function Conundrums in the Modern Machine Learning Paradigm," 2023 International Conference on Computer and Applications (ICCA), Cairo, Egypt, 2023, pp. 1-8, doi: 10.1109/ICCA59364.2023.10401760.

[15]. A. Ahamed, N. Ahmed, J. I., Z. Hossain, E. Hasan and T. Abbas, "Advances and Evaluation of Intelligent Techniques in Short-Term Load Forecasting," 2024 International Conference on Computer and Applications (ICCA), Cairo, Egypt, 2024, pp. 1-9, doi: 10.1109/ICCA62237.2024.10927804.

[16]. J. I. Janjua, O. Anwer and A. Saber, "Management Framework for Energy Crisis & Shaping Future Energy Outlook in Pakistan," 2023 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT), Amman, Jordan, 2023, pp. 312-317, doi: 10.1109/JEEIT58638.2023.10185730.

[17]. W. Alomoush, T. A. Khan, M. Nadeem, J. I., A. Saeed and A. Athar, "Residential Power Load Prediction in Smart Cities using Machine Learning Approaches," 2022 International Conference on Business Analytics for Technology and Security (ICBATS), Dubai, United Arab Emirates, 2022, pp. 1-8, doi: 10.1109/ICBATS54253.2022.9759024.

[18]. T. Abbas, J. I. and M. Irfan, "Proposed Agricultural Internet of Things (AIoT) Based Intelligent System of Disease Forecaster for Agri-Domain," 2023 International Conference on Computer and Applications (ICCA), Cairo, Egypt, 2023, pp. 1-6, doi: 10.1109/ICCA59364.2023.10401794.

[19]. N. Shinn, F. Cassano, A. Labash, B. Gopinath, K. Narasimhan, and S. Yao, “Reflexion: Language Agents with Verbal Reinforcement Learning,” in Proc. Advances in Neural Information Processing Systems (NeurIPS), 2023.