RTTC: 奖励引导的协作测试时计算

J. Pablo Muñoz*
Intel Labs
Santa Clara, CA, USA
pablo.munoz@intel.com
jpablomch@gmail.com

Jinjie Yuan*
Intel Corporation
Beijing, China
jinjie.yuan@intel.com

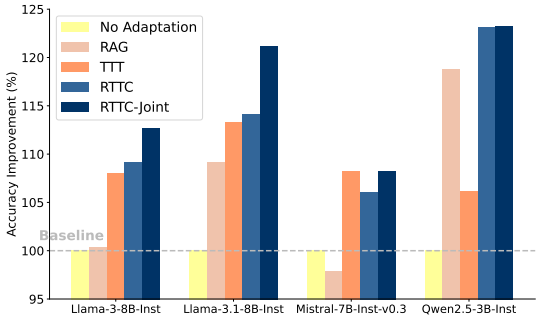

Figure 1: RTTC 在各种 LLM 上的下游任务表现概览。“精度提升”表示在 §?? 中评估的五个下游任务的相对平均得分。

Abstract

测试时计算（TTC）作为一种增强大型语言模型（LLMs）推理性能的强大范式已经出现，利用了测试时训练（TTT）和检索增强生成（RAG）等策略。然而，最佳的适应策略因查询而异，不加区分地应用 TTC 策略会带来大量的计算开销。在这项工作中，我们介绍了奖励引导的测试时计算（RTTC），一个新的框架，通过一个预训练的奖励模型自适应地为每个查询选择最有效的 TTC 策略，以最大化在不同领域和任务中的下游准确性。RTTC 在分布式服务器-客户端架构中运行，仅在必要时从远程知识库中检索相关样本，并在客户端设备上应用 RAG 或轻量级微调。为了进一步减少冗余计算，我们提出了查询状态缓存，它能够在检索和适应层次上高效地重用历史查询状态。跨多个 LLM 和基准的广泛实验表明，RTTC 始终能获得优于原始 RAG 或 TTT 的准确性，验证了自适应及奖励引导 TTC 选择的必要性和 RTTC 在可扩展的高性能语言模型适应中的潜力。

1 介绍

大型语言模型在广泛的任务中取得了显著的性能。然而，它们在推理时的鲁棒性和适应新领域或分布变化的能力仍是开放的挑战。传统

TTC Strategy	Latency	Memory	Accuracy
No Adaptation (Direct Inference)	+	+	+
Retrieval-Augmented Generation (RAG)	++	++	+++
Test-Time Training (TTT)	+++	+++	+++

Table 1: 比较不同测试时计算（TTC）策略的计算成本和准确性提升收益。更多的加号表示更高的成本或更高的准确性。尽管 RAG 和 TTT 都提供了显著的准确性提升，但每种策略都有其优势，具体取决于查询和情境。

上，已经探索了两大范式来提高 LLM 在测试时的性能：检索增强生成（RAG）(Lewis et al., 2020; Gao et al., 2023b)，它通过检索的知识增强模型输入，以及测试时训练（TTT）(Sun et al., 2020; Hardt and Sun, 2024; Hübotter et al., 2024; Snell et al., 2025)，它使用相关样本来调整模型参数。两种方法均已证明了显著的有效性。

然而，RAG 和 TTT 的实际部署提出了两个关注点：准确性和效率。准确性：RAG 或 TTT 的效果在不同查询之间也有差异。有时 RAG 的表现优于 TTT，反之亦然。对于某些输入，模型的直接响应已经足够准确，而对于其他输入，推理阶段可能需要检索增强或自适应微调来提高精度。效率：RAG 和 TTT 都引入了显著的计算开销——RAG 通过扩展输入上下文增加了推理延迟和内存使用，而 TTT 需要额外的微调步骤和用于模型更新的内存。此外，天真地将 RAG 或 TTT 应用于每一个查询可能导致不必要的计算和低效的资源使用。每种策略相关的计算成本和准确性的总结见表格 1 。因此，一种能够在测试时动态选择每个查询的最佳策略的自适应方法对于在最大化性能的同时最小化开销是至关重要的。

为了解决上述挑战，我们提出了奖励引导的测试时计算（RTTC）。该框架动态地为每个查询选择三种策略中的一种：无适应（即，在不适应的情况下返回模型的响应）、检索增强生成（RAG）以及测试时训练（TTT）。RTTC 的核心是一种预训练的奖励模型，它评估候选响应，并以查询自适应的方式引导系统选择最有

*Equal contribution.

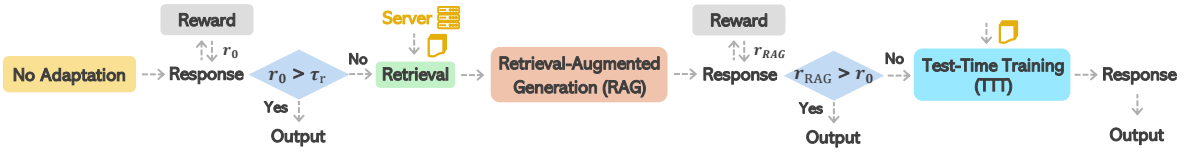

Figure 2: R eward-guided T est- T ime C ompute（RTTC ）的工作流程。对于每个查询，预训练的奖励模型评估候选响应并选择最佳适应策略（无适应、RAG 或 TTT）。τ_r 是一个预定义的阈值。

效的适应策略（见 Figure 2 ）。这种奖励引导的协作使得 RTTC 能够自适应地利用最适合每个查询的策略，在不同领域和任务中实现稳健的下游性能提升。与之前只静态应用 RAG 或 TTT 的方法不同，我们的方法引入了一种有原则的决策机制以最大化性能。此外，RTTC 还引入了查询状态缓存（QSC ）机制，以进一步优化测试时的效率。QSC 利用历史查询嵌入及其相关联的检索样本或微调后的模型状态（例如，LoRA (Hu et al., 2022) ），可能绕过重复检索和微调的需求，从而减少计算开销和延迟。总体而言，我们的主要贡献是：

1. 我们介绍了 RTTC ，这是一种奖励引导的协作测试时间计算框架。我们设计了一种有效的决策过程，该过程利用预训练的奖励模型来自适应地选择最佳推理策略，从而实现稳健的 LLM 适应。

2. 我们进一步提出了一种查询状态缓存 (QSC) 机制，该机制复用历史查询信息，从而在推理过程中减少冗余计算和延迟。

3. 大量实验表明，RTTC 一直优于基线，在多个 LLMs 和下游任务中取得了更高的准确率。

总之，我们的工作首次在一个奖励引导、查询自适应的框架下统一了直接推理、RAG 和 TTT。本文的其余部分组织如下。我们在 §2 中讨论相关工作。然后，§3 描述了 RTTC 系统，而 §?? 展示了在各种下游任务上的结果。我们的最终想法在 §?? 中。

2 相关工作

测试时计算。测试时计算技术被提出作为一种替代方案，用于通过扩大模型参数来提高模型性能 (Snell et al., 2025) 。显著的策略包括连锁思维（CoT）提示 (Wei et al., 2022) 和少样本学习 (Brown et al., 2020) 。CoT 提示通过引导模型完成中间步骤来分解复杂任务，增强其处理复杂查询的能力。少样本学习帮助模型以仅仅几个例子适应新任务。其他测试时计算方法包括验证模型的结果，例如通过代码执行 (Brown et al., 2025) 。

检索增强生成（RAG）。RAG 已成为通过在推理时结合外部知识来提升模型性能的一个重要范式 (Lewis et al., 2020; Gao et al., 2023b) 。通过检索相关文档并增强模型的输入，RAG 使大语言模型能够访问超出其预训练语料库的最新或特定领域的信息。这种方法在开放领域问答、事实验证和知识密集型任务中表现出了强大的结果。由于检索和较长的输入序列，RAG 引入了额外的计算开销，并且其有效性可能因检索内容的质量和任务的不同而有所变化。

测试时训练（TTT）。TTT (Hardt and Sun, 2024; Hübotter et al., 2024; Akyürek et al., 2024) 已被证明能通过在推理时对检索到的样本进行微调来有效地使模型适应分布变化。最近的研究，如 SIFT (Hübotter et al., 2024) ，已经改进了 TTT 的检索策略，而 Omni-ARC (IronbarArc24, 2024) 则利用 TTT 在 ARC-AGI 挑战赛 (ARC-AGI, 2025; Chollet, 2019) 中达到了最新的结果。尽管有这些进展，TTT 可能会带来显著的计算和内存成本。

系统考虑。在分布式环境中，之前的研究探索了使用分布式索引和多服务器架构来扩展检索和适应性，以加速对大型数据集的查询处理 (Hardt and Sun, 2024; Douze et al., 2024) 。尽管这些方法专注于高效的数据访问和检索，我们的工作强调了适应性测试时间计算和下游任务性能。RTTC 可以灵活地结合这些分布式检索技术，以在需要时进一步优化效率。

接下来，我们研究 RTTC ，这一系统通过奖励导向的协同测试时间计算来增强模型性能。

3 RTTC 系统

本节概述了奖励引导的协作测试时计算 (RTTC) 的架构和工作流程，该系统旨在优化大型语言模型 (LLM) 的性能。RTTC 在一个奖励驱动的、查询自适应的框架内统一了直接推理、检索增强生成 (RAG) 和测试时训练 (TTT)，利用远程多域知识库实现稳健的适应性。整体工作流程如图 2 所示，并在算法 ?? 中形式化总结。

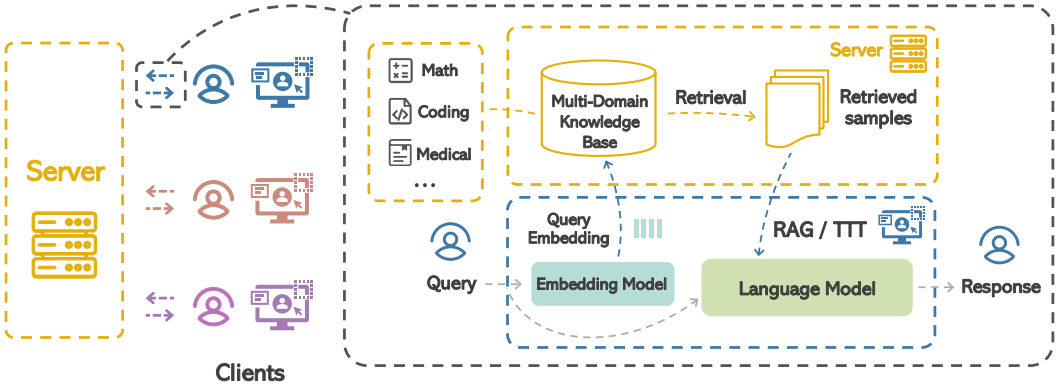

Figure 3: 检索和测试时计算阶段概览。当奖励模型确定 LLM 响应未经适配不符合预期时，会从远程多域知识库中检索相关样本。通过利用先进的测试时计算策略（如 RAG 或 TTT），RTTC 在客户端设备上提升模型性能。

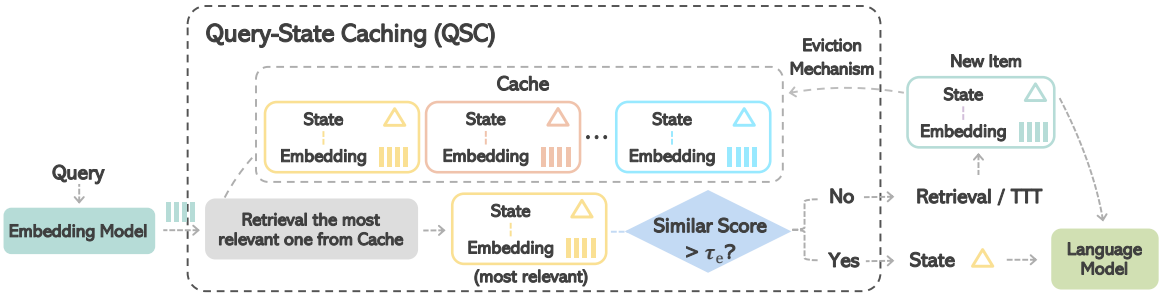

Figure 4: 多轮缓存与高效测试时计算的查询-状态缓存策略 (QSC) 的概述。QSC 兼容 RAG 和 TTT：对于 RAG，缓存状态由检索到的样本组成；对于 TTT，缓存状态存储微调的模型状态（例如，LoRA 适配器 (Hu et al., 2022))。通过管理历史查询嵌入和其关联状态，QSC 加速了测试时计算成本（检索和微调）。