用于大语言模型持续适应的记忆库压缩

Thomas Katraouras
tkatraouras@uth.gr 塞萨利大学 沃洛斯，希腊

Dimitrios Rafailidis
draf@uth.gr 塞萨利大学 沃洛斯，希腊

arXiv:2601.00756v1 [cs.LG] 2 Jan 2026

摘要

大语言模型（LLM）已成为众多日常应用的主流技术。然而，随着数据不断演变，其知识很快就会过时。持续学习旨在用新信息更新大语言模型，同时不抹除此前获得的知识。尽管全量微调等方法能够整合新数据，但其计算成本高昂，并且容易导致灾难性遗忘，即原有知识被覆盖。基于记忆的方法通过为大语言模型配备记忆库来应对这一问题；记忆库是一种用于存储信息以供未来使用的外部记忆模块。然而，这些方法面临一个关键限制：在现实场景中，当大规模数据流不断到来时，记忆库会持续增长。本文提出 MBC，这是一种在在线适应学习过程中通过码本优化策略压缩记忆库的模型。为确保学习稳定性，我们还引入了一种在线重置机制，以防止码本坍塌。此外，我们在大语言模型的注意力层中采用键值低秩适配，从而高效利用压缩后的记忆表示。在基准问答数据集上的实验表明，与最具竞争力的基线方法相比，MBC 可将记忆库大小降低至 0.3%，同时在在线适应学习过程中保持较高的保持准确率。我们的代码已公开于 https://github.com/Thomkat/MBC。

ACM计算分类体系概念

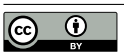
•**计算方法** → **自然语言生成**；**机器学习**；**人工智能**；**自然语言处理**；**在线学习设置**。

关键词

大语言模型、持续学习、记忆库、记忆压缩、问答、记忆增强大语言模型

ACM参考文献格式：

Thomas Katraouras 和 Dimitrios Rafailidis。2026 年。面向大语言模型持续适应的记忆库存储库压缩。载于第*41*届*ACM/SIGAPP*应用计算研讨会（*SAC '26 27，2026*年，*238*），*3*月——希腊塞萨洛尼基。ACM，纽约，纽约州，美国，第若干页。https://doi.or g/10.1145/3748522.3779742



本作品根据知识共享署名4.0国际许可协议授权。*SAC '26*，塞萨洛尼基，希腊
© 2026版权所有者/作者所有。ACM ISBN 979-8-4007-2294-3/2026/03
https://doi.org/10.1145/3748522.3779742

1 引言

大语言模型（LLM） [20, 33] 在包括机器翻译 [38], 、摘要生成 [35], 、问答 [31], 和高级推理 [44]在内的广泛自然语言处理任务中展现出强大的性能。如今，它们已广泛应用于搜索引擎 [47] 和个人助理 [4]等众多应用。然而，这些模型的一个主要局限是其静态性 [10]。模型训练完成后，其参数仅反映训练集中的数据，因而难以轻松纳入新知识。这导致了知识截止问题：随着新信息不断出现，模型内部的知识会逐渐过时 [20, 33]。

为解决这一局限，人们提出了检索增强生成（RAG）策略 [14, 40]。在推理时，冻结的大语言模型会使用检索器从外部语料库中获取相关段落，使模型无需重新训练即可获取最新信息 [14]。然而，RAG 方法面临若干挑战。它们依赖最近邻搜索，这会增加计算开销和延迟 [40]。检索质量会影响大语言模型的性能，而检索中的错误会直接传递给生成器 [46]。此外，检索器通常需要进行领域特定调优，并且可能难以跨领域泛化 [19]。此外，检索到的段落通常会与查询拼接在输入上下文窗口中，这限制了模型充分利用信息的能力；当拼接后的长度超出模型容量时，还会引发问题 [14]。这些问题限制了 RAG 在流式环境中进行长期适应的可扩展性，也反映了真实世界的情形。

为解决大语言模型的长期适应问题，研究者提出了持续学习方法 [30, 43]。在这种设置下，模型会随着新数据的到来而更新。最简单的方法是全量微调，也称为均匀微调 [9]，因为所有词元的权重相同；该方法利用新数据更新所有参数。虽然这种方法对小型模型有效，但用于大型大语言模型时计算开销很大，而且容易导致灾难性遗忘 [18], : 模型针对新信息进行优化时，先前已学知识的表现会下降。参数高效微调（PEFT）方法 [41]，例如适配器 [7, 22], 、前缀调优 [15], 和 LoRA [8]，通过引入小型可训练模块，同时冻结大部分参数，来降低计算成本。这些方法降低了训练开销，但在部署时仍需要基于梯度的更新，因此不适用于流式场景。研究者还提出了其他策略，以实现选择性且稳定的更新，例如将更新限制在预先定义的显著片段中 [5], 、通过元学习获取词元重要性权重 [9], ，或交替使用过去的样本和新样本

通过重放 [2, 28]。尽管有这些改进，根本性的局限依然存在，即需要反复优化、计算开销和延迟开销高昂，以及仍易发生灾难性遗忘 [30]。

大语言模型持续学习的一个有前景的方向是记忆增强 [6, 21, 39]。与从外部语料库检索原始文本不同，信息会直接存储在结构化的记忆模块中，并且该模块也可以动态更新。在推理时，模型可以利用这些已存储的表示来调整自身行为并纳入新信息。这避免了反复进行梯度更新，并在存储的知识与模型的计算之间建立了直接联系 [6]。然而，随着处理的文档越来越多，记忆增强也带来了挑战：记忆库会不断增长。这会增加存储成本并减慢推理速度，因为模型必须关注日益庞大的上下文集合 [45]。在不干扰模型原有行为的情况下扩展记忆仍然很困难，因此，记忆增强策略需要重新训练或微调才能持续有效。

最近，为了克服这一缺陷，研究者提出了记忆增强框架，将每篇文档学习到的调制参数存储在外部记忆库中 [32]。这些方法保持基础模型冻结，使模型以整个记忆库而非单个检索到的文档为条件，并在适应过程中避免进一步微调，同时缓解灾难性遗忘。然而，在文档流达到数十万甚至数百万条目的现实场景中，记忆库会变得非常庞大，难以管理。这凸显出可扩展性是记忆增强系统中尚待解决的问题，同时也表明需要在适应能力、效率和稳定性之间取得平衡。

本文提出 MBC，这是一种在保持下游问答（QA）任务高性能的同时压缩记忆库的模型。具体而言，我们作出以下贡献：

- 我们提出一种基于码本优化策略的记忆库压缩方法，以码本索引取代完整的文档表示进行存储。此外，我们还引入在线重置策略，以防止码本坍塌，并确保码本利用率均衡且训练稳定。

- 我们采用仅针对模型注意力层的键值低秩适配。通过这种方式，无需全量微调即可提升所提模型在新数据到来时的适应能力。

我们在基准问答数据集上进行了实验，并将 MBC模型与基线方法进行比较。结果表明，与基线策略的初始存储库大小相比，MBC 可将存储库大小显著缩减至 0.3%，同时提高问答准确率。此外，在具有挑战性的在线适应场景中评估灾难性遗忘时，我们的模型仍能保持较高的保留准确率。

本文其余部分结构如下：第2节对大语言模型中的在线学习问题进行形式化表述。第3节概述MBC模型，第4节提供实验评估。最后，第5节总结全文，概述主要发现并讨论未来可能的研究方向。

2 大语言模型的在线适应

设 f_{θ_b} 为一个经过预训练但已过时的语言模型，其参数为 θ_b。在线适应期间，f_{θ_b} 通过一个新文档流持续更新，该文档流记为 $D^{test} := \{d_i\}$。适应过程会生成更新后的模型 $f_{\tilde{\theta}_b}$ [9]。随后，在一组查询 $Q^{test} := \{q_i\}$ 及其对应标签 $Y^{test} := \{y_i\}$ 上评估该适应后的模型。假设每个查询—标签对 (q_i, y_i) 都是从以相应文档 d_i 为条件的分布中采样得到的，即 $(q_i, y_i) \sim p(q, y \mid d_i)$。例如，在问答场景中，$q_i$ 可能表示一个关于 d_i 所含信息的问题，而 y_i 则是正确答案 [32]。使用 D^{test} 进行适应时，模型无法访问相关查询 Q^{test}。因此，更新过程必须与查询无关。为此，我们假设可以访问一个辅助训练集 D^{train}，其中包含相关查询 Q^{train} 及标签 Y^{train}，其定义方式与 (Q^{test}, Y^{test}) 相同。该辅助集提供了查询与文档之间关系的示例，并引导模型在更新参数的同时保留既有知识、提升对未来查询 [9]的处理能力。涉及 $(D^{train}, Q^{train}, Y^{train})$ 的训练步骤称为学习阶段。随后，在无法访问查询的情况下，使用测试流 D^{test} 更新 f_{θ_b} 的过程称为在线适应阶段 [32]。

3 提出模型

3.1 聚合上下文记忆

所提出的模型包含三个核心组件：(i) 一个用于编码文档的摊销网络，(ii) 一个用于存储编码信息的记忆库，以及 (iii) 一个用于综合存储信息以回答给定查询的聚合网络。

摊销网络。 摊销网络负责将每篇文档映射为紧凑的潜在表示，以便高效存储和检索 [32]。形式上，该网络记为 $g_{\theta_{amort}}$，由 T5编码器-解码器模型实现 [24]。给定一篇文档 d_i，摊销网络会生成连续的潜在表示 $\phi_i := g_{\theta_{\text{amort}}}(d_i) \in \mathbb{R}^{T \times D}$，其中 T 是该表示中的词元数量，D 是基础模型的隐藏维度。

记忆库。 从文档流中生成的上下文向量 ϕ_i 存储在外部记忆库 $\mathrm{M} := \{\phi_i \mid d_i \in D^{test}\}$ [32] 中。该记忆库为基础大语言模型提供不断扩充的知识库。

聚合网络。 当查询 q_i 在测试时被提出时，模型必须从记忆库中检索相关信息。聚合网络（记为 h_ψ）经过训练，可动态执行此功能。它以整个记忆库 M 和当前查询 $g_{\theta_{input}}(q_i)$ 的编码表示作为输入，其中 θ_{input} 使用与 θ_{amort} 相同的架构。该网络对 M 的排列顺序具有置换不变性。通过交叉注意力机制 [11, 36], ，它将存储的上下文向量合成为单个查询特定调制 $\phi_i^* := h_\psi(g_{\theta_{input}}(q_i),\ \mathcal{M})$。该调制 ϕ_i^* 充当一组软

提示，通过 P-tuning v2 [17]作为可学习前缀注入基础大语言模型每个自注意力层的键值矩阵中。形式上，经过调制的基础大语言模型可表示为$f_{\theta_b}^{\phi_i^*}(q_i)$：$= f_{\theta_b}(q_i; \{K'_\ell,\ V_{\prime\ell}\}_{\ell=1}^{L})$，其中 K'_ℓ，$V_{\prime\ell}$表示第 ℓ层在添加 ϕ_i^*后修改的键矩阵和值矩阵。为在推理时高效处理大型记忆库，采用分层调制聚合策略。具体而言，上下文集首先被划分为更小的子组，每个子组分别进行聚合，然后递归合并所得表示，直至获得单个最终调制。该分治过程将内存复杂度降低至 O(MT)，其中 M 是超参数，T 表示词元数量，即使存储文档数量增加 [32]，也能确保可扩展性。

3.2 记忆库压缩的码本优化

存储连续上下文向量 $\{\phi_i\}$ 的一个关键局限在于，随着文档流不断增长，记忆库 M 可能变得非常庞大。为解决这一问题，我们引入一种采用向量量化变分自编码器（VQ-VAE）风格的 [34] 量化模块来压缩记忆。我们不再存储高维连续向量 ϕ_i，而是将每个上下文向量映射到学习得到的有限码本 $E \in \mathbb{R}^{N_c \times D}$ 中与之最近的条目，其中 N_c 表示码向量的数量：c_i:= argmin $\|\phi_i - E_j\|_2^2$。选中的j码本向量记为 $\hat{}\ \phi_i^{\text{hard}}$:$= E_{c_i}$，只存储整数索引 c_i，从而得到压缩记忆库表示 $\text{M}^{\text{VQ}} := \{c_i\}$。由于 $\hat{}\ \phi_i^{\text{hard}}$是离散且不可微的，在前向传播中，我们使用直通估计器（STE） [1] 来定义 $\hat{}\ \phi_i$：$= \phi_i + \text{sg}[\hat{}\ \phi_i^{\text{hard}} - \phi_i]$，其中 sg[·] 表示停止梯度算子。因此，$\hat{}\ \phi_i$ 取值为 $\hat{}\ \phi_i^{\text{hard}}$，同时仍允许梯度回传至 ϕ_i。在后续记法中，$\hat{}\ \phi_i$表示可微的前向传播表示，而 $\hat{}\ \phi_i^{\text{hard}}$仅用于向量量化损失。由此，记忆库的有效存储内容缩减为索引集合及码本 E。码本本身在端到端训练过程中进行优化。推理时，利用存储的索引检索对应的量化向量 $\{\phi_i\}$，随后将其与查询表示一同聚合，以生成调制 $\hat{}\ \phi_i^*$。

3.3 在线码本重置

为防止利用不足和码本崩溃（即只有一小部分码字被反复使用），训练期间会根据代码使用率的指数移动平均（EMA）更新码本。对于大小为 K的小批次，令

$$n_j := \sum_{i=1}^{K} \mathbf{1}[c_i = j], \quad u_j := \gamma\, u_j + (1-\gamma)\, n_j \quad \forall j \in \{1, \ldots, N_c\} \tag{1}$$

其中 u_j 跟踪平滑后的使用情况，$\gamma \in (0,1)$ 是衰减率超参数。使用量低于超参数阈值 ϵ 的码字会被标记为非活跃：I_{dead}:= $\{\ j \mid u_j < \epsilon\}$。当 $I_{\text{dead}} \neq \emptyset$ 时，从当前批次中采样至多 个不同的编码器输出（$\{\phi_{s(j)}\}$），以重新初始化相应的码本向量：

$$E_j := \phi_{s(j)} \quad \forall j \in I_{\text{dead}}, \qquad u_j := \bar{u} := \frac{1}{N_c} \sum_{\ell=1}^{N_c} u_\ell \tag{2}$$

其中，s（j）表示从批次中均匀随机且不放回地采样的索引，$\bar{u}$表示所有码字的平均使用量，以确保重新初始化的条目仍保留非忽略不计的先验使用量估计。该过程仅在训练期间应用（不计算梯度），能够维持码本多样性并防止码本崩溃，我们将在第4.4.4节通过实验予以展示。

3.4 用于调制适配的键值低秩适配

$\hat{}$ 调制 ϕ_i^*被注入基础大语言模型每个自注意力层的键值对中。不同于 [32], 中所述研究将基础大语言模型完全冻结的做法，我们专门在键和值投影中引入轻量级低秩适配（LoRA） [8] 模块（KV-LoRA）。这里，θ_b 表示预训练模型 f_{θ_b}的冻结参数。此外，还添加了一小组可训练参数 $\theta_{\text{KV-LoRA}}$，用于对调制后的键矩阵和值矩阵进行低秩更新：

$$K''_\ell := K'_\ell + A_{K,\ell} B_{K,\ell}, \qquad V''_\ell := V'_\ell + A_{V,\ell} B_{V,\ell} \tag{3}$$

其中 $A_{K、\ell}$、$A_{V、\ell} \in \mathbb{R}^{D \times r}$和 $B_{K、\ell}$、$B_{V、\ell} \in \mathbb{R}^{r \times D}$是秩为 $r \ll D$ 的低秩因子。在实践中，更新会按因子 α/r进行缩放，并通过丢弃进行正则化。KV-LoRA 仅应用于最后 n_{lora} 个 Transformer 层，在计算效率与适配能力之间取得平衡。r、α、n_{lora}以及丢弃概率 ρ 是超参数。因此，适配后的模型具有参数.θ_b：$= \theta_b \cup \theta_{\text{KV-LoRA}}$，这保留了预训练知识，使注意力机制能够更有效地利用调制 $\hat{}\ \phi_i^*$。形式化地，调制后的基础大语言模型表示为：

$$f_{\tilde{\theta}_b}^{\hat{\phi}_i^*}(q_i) := f_{\hat{\theta}_b}(q_i; \{K''_\ell, V''_\ell\}_{\ell=1}^{L}) \tag{4}$$

3.5 端到端训练目标

整个架构采用端到端方式进行训练，包括摊销网络和聚合网络、VQ 码本以及 KV-LoRA 模块。主要目标是问答（QA）损失 L^{QA}，其定义为：在给定查询 q_i、相应文档 d_i及其调制 ϕ_i^*的条件下，预测目标序列 y_i的负对数似然。答案生成由基础大语言模型 $f_{\hat{\theta}_b}^{\phi_i^*}(q_i)$完成· ˆ：$\text{L}^{\text{QA}}$:$= -\log p_{\hat{\theta}_b}^{\phi_i^*}($ $y_i \mid q_i$、d_i)。为实现紧凑的离散记忆表示，使用向量量化损失 L^{VQ}。给定一个连续上下文向量 $\phi_i \in \mathbb{R}^{T \times D}$，其最近码本条目表示为 $\hat{}\ \phi_i^{\text{hard}}$:$= E_{c_i}$。量化损失使用 $\hat{}\ \phi_i^{\text{hard}}$进行定义，以确保码本和编码器都得到适当的更-新：

$$\mathcal{L}_{\text{VQ}} := \|\text{sg}[\phi_i] - \hat{\phi}_i^{\text{hard}}\|_2^2 + \beta\, \|\phi_i - \text{sg}[\hat{\phi}_i^{\text{hard}}]\|_2^2 \tag{5}$$

其中，$\beta > 0$ 是承诺成本超参数。第一项更新选定的码本条目 E_{c_i}，而第二项则促使编码器输出 ϕ_i 保持接近其量化分配结果。最终目标是加权组合：

$\mathrm{L}_{\text{total}} = \mathrm{L}_{\text{QA}} + \lambda_{\text{VQ}}\,\mathrm{L}^{\text{VQ}}$，其中 λ_{VQ} 是用于平衡量化损失影响的超参数。在训练集期间，摊销网络（θ_{amort}）、输入编码器（θ_{input}）、聚合网络（ψ）、KV-LoRA 模块（$\theta_{\text{KV-LoRA}}$）和码本（E）的参数都进行端到端优化，而基础模型参数 θ_b 保持冻结：

$$\min_{\theta_{\text{amort}},\theta_{\text{input}},\psi,\theta_{\text{KV-LoRA}},E} \frac{1}{K}\sum_{i=1}^{K}\left[\mathcal{L}_{\text{QA}}(q_i,d_i,y_i) + \lambda_{\text{VQ}}\,\mathcal{L}_{\text{VQ}}(\phi_i,\hat{\phi}_i^{\text{hard}})\right] \tag{6}$$

其中 K 为批次大小。使用 Adam [12] 优化器进行优化。算法 1 展示了 MBC 端到端优化算法的概览。

3.6 MBC 的在线适应

训练完成后，进入在线适应阶段。此阶段无需进行基于梯度的更新，完全通过前向传播运行。该过程由两个部分组成：

记忆化。 对于测试流 D^{test} 中到达的每个新文档 d_i，摊销网络 $g_{\theta_{\text{amort}}}$ 将其 d_i 编码为上下文向量 ϕ_i，随后将其量化为 E中的最近码本条目。所得离散代码 c_i 存储在压缩记忆库 M^{VQ}中。

推理。 当收到查询 q_i时，模型使用记忆库中存储的代码 $\{c_j\}$，并从码本 E 中检索相应的量化上下文向量 $\{\hat{\phi}_j\}$。这些向量与查询表示 $g_{\theta_{\text{input}}}(q_i)$ 一起由 h_ψ 聚合，以生成查询特定调制 $\hat{\phi}_i^*$。该调制对 KV-LoRA增强的基础大语言模型 $f_{\theta_b}(q_i)$ 进行条件控制，随后由该模型生成最终答案。在线适应与评估过程见算法2。

4 实验评估

4.1 数据集

遵循 [9, 32]，，我们在三个问答数据集上评估考察模型：

StreamingQA [16]。 其中包含由标注人员创建或使用语言模型生成的问题。这些问题基于带有时间戳的英文 WMT 新闻文章（2007–2020 年），这些文章也包含在数据集中。遵循以往的设置，我们使用 21K 个训练集问题、1.7K 个验证集问题和 5K 个测试集问题，同时使用数量相同的文档。对于问答预训练基线，使用 40K 个训练集问题和 4K 个验证集问题。

SQuAD [25]。 斯坦福问答数据集（SQuAD）包含通过众包收集的维基百科问题，答案是文章中的文本片段。按照以往的设置，我们使用 39.9K 个训练集问题、5.6K 个验证集问题和 10.6K 个测试集问题，分别对应 8.6K 篇训练集文档、1.2K 篇验证集文档和 2.1K 篇测试集文档。对于问答预训练基线方法，使用 40K 个训练集问题和 2.1K 个验证集问题。

ArchivalQA [37]。 该数据集由《纽约时报》标注语料库中的文章构建而成， [26]问题由语言模型生成。答案是文章中的文本片段。沿用此前的

算法 1： MBC 端到端优化

输入： 摊销参数 θ_{amort}、输入编码器参数 θ_{input}、基础大语言模型参数 θ_b、聚合参数 ψ、KV-LoRA 参数 $\theta_{\text{KV-LoRA}}$，隐藏维度 D、训练语料库 D^{train}、学习率 η、训练轮次 m、批次大小 K、向量量化承诺代价 β_{commit}、向量量化权重 λ_{VQ}，码本大小 N_c，重置阈值 ϵ，指数移动平均衰减率 γ

输出： $\theta_{\text{amort}}, \theta_{\text{input}}, \psi, \theta_{\text{KV-LoRA}}, E$

// 初始化码本和使用率指数移动平均

$E_j \sim \mathcal{U}(-\frac{1}{N_c}, \frac{1}{N_c})\ \forall j \in \{1,\dots,N_c\}$，　$u_j \leftarrow 0\ \forall j$

对于训练轮次 $= 1 \to m$ **执行**

　样本文档 $\{d_1、\dots、d_K\} \subset D^{\text{train}}$

　示例问答对 $(q_i, y_i) \sim p(q, y \mid d_i)$ 对于 $i = 1, \dots, K$

　对于 $i = 1 \to K$ **执行**

　　$\phi_i \leftarrow g_{\theta_{\text{amort}}}(d_i)$；　// 编码文档

　　// 向量量化（最近码字）

　　$c_i \leftarrow$ 参数最小值$_{j\in\{1,\dots,N_c\}} \|\phi_i - E_j\|_2^2$，　$\hat{\phi}_i^{\text{hard}} \leftarrow E_{c_i}$

　　$\hat{\phi}_i \leftarrow \phi_i + \text{sg}[\hat{\phi}_i^{\text{hard}} - \phi_i]$；// 直通估计器

　// 更新码字使用率的指数移动平均，并重置死码

　$n_j \leftarrow \sum_{i=1}^{K} \mathbf{1}[c_i = j]\ \forall j$，　$u_j \leftarrow \gamma\, u_j + (1-\gamma)\ n_j\ \forall j$

　$I_{\text{dead}} \leftarrow \{\ j \in \{1,\dots,N_c\}\ \mid u_j < \epsilon\}$

　如果 $|I_{\text{dead}}| > 0$ **那么**

　　// 用随机批次样本替换死码

　　$S \sim Unif\left(\binom{\{1,\dots,K\}}{|S|}\right)$，　$|S| =$ 最小值$(|I_{\text{dead}}|,\ K)$

　　$E_j \leftarrow \phi_{s(j)}\quad \forall j \in I_{\text{dead}}$，最多 $|S|$

　　$u_j \leftarrow \frac{1}{N_c}\sum_{\ell=1}^{N_c} u_\ell\quad \forall j \in I_{\text{dead}}$；　// 重置用量

　// 将量化后的上下文与查询聚合

　$\hat{\phi}_i^* \leftarrow h_\psi(g_{\theta_{\text{input}}}(q_i),\ \{\hat{\phi}_i\}_{i=1}^{K})$

　// 通过调制基础大语言模型计算问答损失

　$\mathrm{L}_{\text{QA}} \leftarrow -\frac{1}{K}\sum_{i=1}^{K}$交叉熵$\left(f^{\hat{\phi}_i^*}_{\theta_b}(q_i),\ y_i\right)$

　// VQ 损失（码本项和承诺项）

　$\mathrm{L}_{\text{codebook}} \leftarrow -\frac{1}{K}\sum_{i=1}^{K} \|\text{sg}[\phi_i] - \hat{\phi}_i^{\text{hard}}\|_2^2$

　$\mathrm{L}_{\text{commit}} \leftarrow \frac{1}{K}\sum_{i=1}^{K} \left\|\phi_i - \text{sg}[\hat{\phi}_i^{\text{hard}}]\right\|_2^2$

　$\mathrm{L}_{\text{VQ}} \leftarrow \mathrm{L}_{\text{codebook}} + \beta_{\text{commit}} \cdot \mathrm{L}_{\text{commit}}$

　$\mathrm{L}_{\text{total}} \leftarrow \mathrm{L}_{\text{QA}} + \lambda_{\text{VQ}}\,\mathrm{L}^{\text{VQ}}$；　// 总目标函数

　// 梯度更新（基础模型 θ_b 冻结）

　$\theta_{\text{amort}} \leftarrow \theta_{\text{amort}} - \eta\nabla_{\theta_{\text{摊销}}}\mathrm{L}_{\text{total}}$

　$\theta_{\text{input}} \leftarrow \theta_{\text{input}} - \eta\nabla_{\theta_{\text{input}}}\mathrm{L}_{\text{total}}$

　$\psi \leftarrow \psi - \eta\nabla_{\psi}\mathrm{L}_{\text{total}}$

　$\theta_{\text{KV-LoRA}} \leftarrow \theta_{\text{KV-LoRA}} - \eta\nabla_{\theta_{\text{KV-LoRA}}}\mathrm{L}_{\text{total}}$

　$E \leftarrow E - \eta\nabla_{E}\mathrm{L}_{\text{total}}$

算法2： MBC在线适配

输入： 测试文档流 D^{test}，测试问答集 $\{(q_i, y_i)\}_{i=1}^{I}$，摊销参数 θ_{amort}，输入编码器参数 θ_{input}，基础大语言模型参数，包含 KV-LoRA $\tilde{\theta}_b$，聚合参数 ψ，学习得到的码本 E

输出： $\{(q_i, y_i)\}$上的精确匹配和F1

$\mathcal{M}_{\text{VQ}} \leftarrow \emptyset$；　// 初始化压缩记忆库

for $d_k \in D^{\text{test}}$ **do**

　$\phi_k \leftarrow g_{\theta_{\text{amort}}}(d_k)$；　// 编码文档

　$c_k \leftarrow \arg\min_j \|\phi_k - E_j\|_2^2$；　// 量化

　$\mathrm{M}_{\text{VQ}} \leftarrow \mathrm{M}_{\text{VQ}} \cup \{c_k\}$；　// 将文档保存到存储库

for $i = 1 \to I$ **do**

　$\hat{\phi}_i^* \leftarrow h_\psi(g_{\theta_{\text{input}}}(q_i), \{E_{c_j}\}_{c_j \in \mathcal{M}^{\text{VQ}}})$；// 使用查询聚合记忆

　查询 $\hat{y}_i \leftarrow f^{\hat{\phi}_i^*}_{\theta_b}(q_i)$；　// 预测答案

// 最终评估；norm()：转为小写，去除标点符号，移除冠词，合并空白字符

精确匹配 $\leftarrow \frac{1}{I}\sum_{i=1}^{I} \mathbf{1}[\text{norm}(y_i) = \text{norm}(\hat{y}_i)]$；　// 精确匹配

F1 $\leftarrow \frac{1}{I}\sum_{i=1}^{I} \text{F1}_{\text{token}}(y_i, \hat{y}_i)$；　// 词元级 F1

return (精确匹配, F1)

设置，我们分别使用 21.7K 个训练集问题、5.3K 个验证集问题和 8.7K 个测试集问题，以及 12.8K 个训练集文档、3.0K 个验证集文档和 5.0K 个测试集文档。对于问答预训练基线方法，使用 12.4K 个训练集问题和 3K 个验证集问题。

4.2 评估协议

为确保基线方法之间的公平比较，我们遵循先前工作的训练配置 [9, 32]。对于每个数据集，模型使用从测试流 D^{test}中采样的 1,665 篇文档进行适应，随后在从同一批文档中抽取的问答对上评估其性能。我们报告精确匹配（EM）和词元级 F1 分数作为评估指标。

EM 分数衡量经过规范化（转为小写、标点处理、冠词移除以及将多个空格合并为一个空格）后与真实答案完全匹配的预测所占比例：

$$\text{EM} = \frac{1}{I}\sum_{i=1}^{I} \mathbf{1}\big[\text{norm}(\hat{y}_i) = \text{norm}(y_i)\big] \tag{7}$$

其中，I 是问答对的数量，$\hat{y}_i$ 是预测答案，y_i 是真实答案。

词元级 F1 分数用于衡量精确率和召回率在词元级别上的调和平均值：

$$\text{Precision}_i = \frac{|\,\text{tok}(\hat{y}_i) \cap \text{tok}(y_i)\,|}{|\text{tok}(\hat{y}_i)|}, \qquad \text{Recall}_i = \frac{|\,\text{tok}(\hat{y}_i) \cap \text{tok}(y_i)\,|}{|\text{tok}(y_i)|} \tag{8}$$

$$\text{F1}_i = \frac{2 \cdot \text{Precision}_i \cdot \text{Recall}_i}{\text{Precision}_i + \text{Recall}_i}, \qquad \text{F1} = \frac{1}{I}\sum_{i=1}^{I} \text{F1}_i \tag{9}$$

其中，tok( ) 表示答案的词元化表示。

4.3 实验设置

4.3.1 实现细节。我们使用四种骨干大语言模型评估 MBC：GPT-2 系列（DistilGPT2 [27], GPT2-Large [23], GPT2-XL [23]）和 LLaMA-2-7B [33],，其参数量分别为 82M、774M、1.5B 和 7B。摊销网络 $g_{\theta_{\text{amort}}}$ 基于 T5 [24],，DistilGPT2 使用 T5-Small，GPT2-Large 使用 T5-Base，而 GPT2-XL 和 LLaMA-2-7B 使用 T5-Large。输入编码器$g_{\theta_{\text{input}}}$ 在 DistilGPT2 中使用 T5-Small，其余模型使用 T5-Base。摊销网络为 DistilGPT2 输出 $T = 12$ 个词元，其余模型输出 24 个词元。聚合网络 h_ψ 由四个交叉注意力块 [11, 36], 组成，其中$g_{\theta_{\text{input}}}$（$q_i$）提供初始查询，记忆库 MVQ 提供键和值，后续块则将前一块的输出作为输入，并生成 $\hat{\phi}_i^*$。训练使用 Adam [12] 优化器，运行 50 个训练轮次。学习率在总步数的前 1% 内线性预热，随后保持在 10^{-5}不变。每个训练轮次结束后进行验证。DistilGPT2 的批次大小设为 64，其余模型设为 32，并使用梯度累积。对于参数量超过 1B 的模型，反向传播期间以 $\rho_{\text{back}} = 0.75$ 的概率应用丢弃；也就是说，每个批次仅随机选取一部分文档计算梯度，其余文档则使用停止梯度 [32]。LLaMA-2-7B 的模型和摊销网络均采用 4 位量化 [3] 进行训练。码本大小固定为 $N_c = 512$，向量量化承诺代价为 $\beta_{\text{commit}} = 0.25$，权重为$\lambda_{\text{VQ}} = 1.0$。码本重置使用指数移动平均衰减率 $\gamma = 0.99$ 和重置阈值 $\epsilon = 10^{-4}$。对于 KV-LoRA，DistilGPT2 使用 $r = 16$，$\alpha = 32$、$\rho = 0.05$，应用于最后 $n_{\text{lora}} = 6$ 层。GPT2-Large、GPT2-XL 和 LLaMA-2-7B 使用 $r = 32$、$\alpha = 64$、$\rho = 0.05$，应用于最后 $n_{\text{lora}} = 16$ 层。对于 GPT-2 系列，我们在 K和 V之间共享 LoRA 下投影矩阵，这意味着 $A_{K,\ell} = A_{V,\ell}$。所有实验均在单个 NVIDIA A100 80GB GPU 上进行。

4.3.2 考察模型。

- **均匀微调**[1]：一种基线方法，在模型更新过程中，新文档中的所有词元都被同等对待。
- **显著片段**[1] [5]：一种基于启发式的方法，仅针对预先识别出的显著片段中的词元进行微调，忽略其余词元。
- **CaMeLS**[1] [9]：上下文感知元学习损失缩放，它使用经过元训练的网络，在微调期间为词元分配重要性权重，使学习聚焦于信息量最大的内容。
- **MAC**[2] [32]：摊销上下文记忆，一种在线适应框架，它冻结基础模型，并使用元学习网络将文档编码为紧凑的调制表示，存储在记忆库中。聚合模块会检索这些调制表示，并将其与查询结合，而无需在推理期间进行梯度更新。• **MBC**[3]：本文提出的模型。

为确保公平比较，我们重新训练了所有基线方法。对于均匀微调、显著片段和 CaMeLS，我们遵循 [9] 的配置。每个预训练大语言模型首先在问答对上进行微调，以获得任务适配基础模型。在此预训练过程中，内循环批次包含 6 个文档–查询–标签三元组，外循环梯度则在 24 个样例上累积，并拆分为 4 个批次，每批 6 个样例。随后，基础模型进行在线适应，在文档流上更新。我们在验证集上通过超参数搜索，为每种基础模型与策略组合选择学习率，搜索范围为 $\{10^{-4},\ 2.5{\times}10^{-5},\ 6.25{\times}10^{-6},\ 1.625{\times}10^{-6}\}$。均匀方法和显著片段的最佳学习率通常为 1.625×10^{-6}，而 CaMeLS 的最佳学习率为 2.5×10^{-5}。大多数情况下使用 Adam；对于大型模型，则使用 Adafactor [29]。对于 MAC，我们遵循 [32] 的配置。具体而言，摊销网络对 DistilGPT2 使用 T5-Small（12 个输出词元），对 GPT2-Large 使用 T5-Base（24 个输出词元），对 GPT2-XL 和 LLaMA-2-7B 使用 T5-Large（24 个输出词元）；输入编码器对 DistilGPT2 使用 T5-Small，对其余模型使用 T5-Base。聚合网络由四个交叉注意力块组成。模型使用 Adam 训练 50 轮次，预热 1 轮后采用恒定学习率 10^{-5}；DistilGPT2 的批次大小为 64，其余模型为 32，并使用梯度累积。对于参数量超过 1B 的骨干网络，会以 $\rho_{\text{back}} = 0.75$ 的概率应用反向传播随机失活；LLaMA-2-7B 则采用 4 位量化 [3] 进行训练。

4.4 实验结果

4.4.1 问答性能评估。表1报告了问答性能。我们将 MBC 与基线方法进行比较，其中 MAC 最具竞争力。在所有数据集和基础大语言模型上，MBC 都能持续提升精确匹配和 F1。平均而言，

[1] https://github.com/nathanhu0/CaMeLS [2] https://github.com/jihoontack/MAC [3] https://github.com/Thomkat/MBC

表1：不同骨干模型和基线方法在 StreamingQA、SQuAD 和 ArchivalQA 上的精确匹配（EM）和 F1分数。值越高表示问答性能越好。

模型（参数量）	方法	StreamingQA		SQuAD		ArchivalQA	
		精确匹配（↑）	F1（↑）	精确匹配（↑）	F1（↑）	精确匹配（↑）	F1（↑）
DistilGPT2 (82M)	均匀	1.62	2.97	1.34	2.78	4.01	3.69
	显著片段	1.62	4.33	1.31	2.50	4.08	3.98
	CaMeLS	1.86	4.38	1.43	3.06	4.11	5.99
	MAC	3.48	8.11	1.90	5.00	5.99	8.87
	MBC（我们的方法）	**3.96（↑13.8%）**	**8.76（↑8%）**	**2.10（↑10.5%）**	**5.36（↑7.2%）**	**6.61（↑10.4%）**	**9.27（↑4.5%）**
GPT2-Large (774M)	均匀	4.14	8.08	3.37	5.62	8.03	6.63
	显著片段	4.26	8.53	4.38	6.79	9.75	7.23
	CaMeLS	5.48	10.31	4.45	7.60	9.28	9.18
	MAC	6.12	11.44	6.14	9.75	10.95	12.15
	MBC（我们的方法）	**7.43（↑21.4%）**	**12.77（↑11.6%）**	**6.99（↑13.8%）**	**10.88（↑11.6%）**	**12.03（↑9.9%）**	**13.68（↑12.6%）**
GPT2-XL (1.5B)	均匀	5.16	9.14	5.87	7.87	9.89	10.46
	显著片段	5.46	11.32	5.66	8.69	10.44	13.68
	CaMeLS	6.98	11.23	6.17	9.93	11.48	14.01
	MAC	7.14	12.01	6.89	10.12	11.48	15.52
	MBC（我们的方法）	**7.49（↑4.9%）**	**12.77（↑6.3%）**	**7.40（↑7.4%）**	**11.96（↑18.2%）**	**12.34（↑7.5%）**	**15.93（↑2.6%）**
LLaMA-2 (7B)	均匀	11.76	12.53	12.78	16.62	17.89	20.01
	显著片段	12.12	18.65	13.32	18.09	18.45	22.21
	CaMeLS*	N/A	N/A	N/A	N/A	N/A	N/A
	MAC	14.01	20.44	13.33	18.17	19.58	23.89
	MBC（我们的方法）	**16.04（↑14.5%）**	**25.33（↑23.9%）**	**14.93（↑12%）**	**22.15（↑21.9%）**	**22.71（↑16%）**	**28.66（↑19.9%）**

* 未报告 *LLaMA-2 (7B)* 上的 *CaMeLS* 结果，因为该模型超出了单张 *NVIDIA A100 80GB GPU* 的内存容量。即使批次大小为 *1*，在我们的硬件限制下也无法复现这一基线方法。

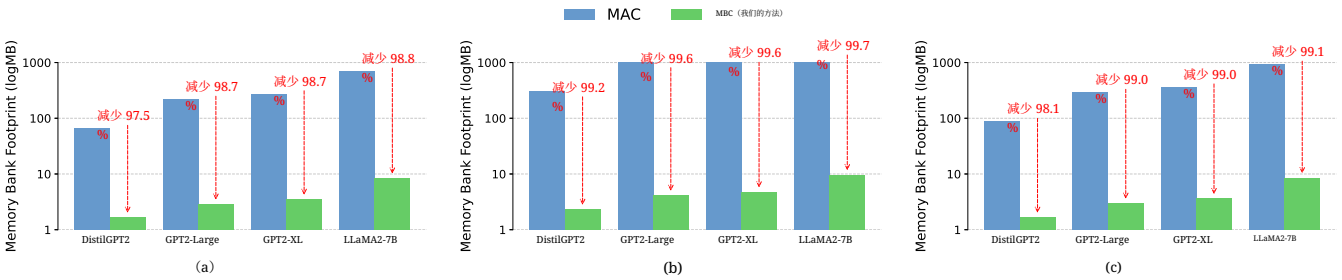


图1：MAC 和 MBC 在 StreamingQA、SQuAD 和 ArchivalQA 上的记忆库占用空间（logMB）。

表2：MAC 和 MBC 的可训练参数（离线）。

方法	DistilGPT2	GPT2-Large	GPT2-XL	LLaMA-2-7B
MAC	197M	927M	1.72B	2.36B
MBC	197.6M（+0.31%）	929.6M（+0.28%）	1.723B（+0.19%）	2.371B（+0.45%）

与 MAC 相比，MBC 的精确匹配提升了 11.84%，F1 提升了 12.99%。性能提升主要源于两项设计。首先，引入 KV-LoRA 使注意力机制能够有效利用调制 $\hat{}\,\phi_i^*$，从而生成准确答案。此外，高效学习得到的码本保留了存储文档的质量，确保压缩机制不会降低性能。

4.4.2 记忆库大小。 图1比较了两种经过研究的记忆增强方法（即 MBC 和 MAC）的记忆库占用空间。对于 MBC，占用空间包括码本和存储在存储库中的索引；对于 MAC，则对应完整的记忆库。在全部三个数据集上，MBC 都显著节省了记忆空间。对于 DistilGPT2，与 MAC 相比，记忆库大小平均减少了 98.27%。对于 GPT2-Large 和 GPT2-XL，平均减少了 99.1%；对于 LLaMA-2-7B，平均减少了 99.2%。这些结果表明，记忆压缩在不同模型规模下始终有效。

为考察 MBC 中码本和 KV-LoRA 的开销，我们比较了两种所研究的记忆增强方法——MAC 和 MBC——的可训练参数数量。这些数据是在离线设置下报告的，也就是说，未考虑

表3：不同基础大语言模型在 StreamingQA、SQuAD 和 ArchivalQA 上的记忆库大小（MB）/ F1保留率（%）。每个条目均报告记忆库占用空间（MB），后跟相应的保留率。

文档数量	DistilGPT2						GPT2-Large					
	StreamingQA		SQuAD		ArchivalQA		StreamingQA		SQuAD		ArchivalQA	
	MAC	MBC	MAC	MBC	MAC	MBC	MAC	MBC	MAC	MBC	MAC	MBC
200	8.21/100	1.52/100	38.1/100	1.60/100	10.91/100	1.53/100	27.36/100	2.54/100	126.99/100	2.7/100	36.36/100	2.56/100
400	16.42/99.5	1.54/99	76.19/99.5	1.70/99	21.82/99.5	1.56/99	54.73/99	2.59/99	253.98/99	2.9/99.5	72.72/99	2.61/99
600	24.63/99	1.56/99	114.29/99	1.80/99	32.72/99.5	1.59/99.5	82.09/99	2.63/99	380.96/98.5	3.1/99.5	109.07/99	2.67/99
800	32.84/99	1.59/99	152.39/99	1.90/98	43.63/99	1.62/99	109.46/98	2.67/99	507.95/98	3.3/99	145.43/98.8	2.73/99
1000	41.04/98	1.61/98	190.48/99	1.99/97.5	54.54/99	1.64/98.5	136.82/97.5	2.71/98.5	634.94/97.5	3.49/98.5	181.79/97.5	2.78/98.8
1200	49.25/98	1.63/98	228.58/98.5	2.09/98	65.45/98.5	1.67/98	164.18/97	2.76/98.5	761.93/97	3.69/98	218.15/97.5	2.84/98.3
1400	57.46/98	1.65/98.5	266.67/98	2.19/98.5	76.35/98.5	1.70/98.5	191.55/96.5	2.8/97	888.91/96.5	3.89/96.5	254.5/96.5	2.89/97.6
1600	65.67/98	1.67/98	304.77/98	2.29/98.5	87.26/98	1.73/98	218.91/97	2.84/97	1015.9/96	4.09/97	290.86/97	2.95/97.2

文档数量	GPT2-XL						LLaMA2-7B					
	StreamingQA		SQuAD		ArchivalQA		StreamingQA		SQuAD		ArchivalQA	
	MAC	MBC	MAC	MBC	MAC	MBC	MAC	MBC	MAC	MBC	MAC	MBC
200	34.2/100	3.16/100	158.73/100	3.32/100	45.45/100	3.18/100	87.56/100	8.04/100	406.36/100	8.2/100	116.34/100	8.06/100
400	68.41/100	3.21/100	317.47/99.5	3.52/99	90.89/100	3.23/99	175.13/100	8.09/99	812.72/99.7	8.4/99.5	232.69/99	8.11/99.5
600	102.61/99	3.25/98.5	476.2/98.8	3.72/99	136.34/99.7	3.29/99	262.69/99.5	8.13/99	1219.08/99.5	8.6/99	349.03/99.8	8.17/99
800	136.82/98	3.29/99	634.94/98.0	3.92/98.7	181.79/99.4	3.35/98.6	350.25/98	8.17/98.5	1625.44/99.5	8.8/98.5	465.38/99.5	8.23/98.5
1000	171.02/97	3.33/98.5	793.67/98.0	4.11/98.8	227.23/98.5	3.4/98.1	437.81/97.4	8.21/98	2031.8/99	8.99/97.5	581.72/98.5	8.28/98.2
1200	205.22/96.7	3.38/98	952.4/97.9	4.31/98	272.68/98	3.46/97.5	525.38/97	8.26/97	2438.16/97	9.19/96.9	698.06/98.2	8.34/98
1400	239.43/96	3.42/97	1111.14/98.0	4.51/98	318.12/98.3	3.51/97.8	612.94/96.5	8.3/96.5	2844.52/96.7	9.39/96.3	814.41/98	8.39/98
1600	273.63/95.5	3.46/96	1269.87/97.5	4.71/97.5	363.57/98	3.58/97.5	700.5/95	8.34/96	3250.88/96.5	9.59/96.5	930.75/97.5	8.45/97.5

在线适应期间存储的文档。如表2所示，所提出的 MBC 模型中码本和 KV-LoRA 引入的额外参数，在所有基础大语言模型中均不到 0.5%。与问答准确率和记忆压缩方面的提升相比，这一开销可以忽略不计。

4.4.3 在线适应期间的知识保留。在本实验中，我们评估模型在继续适应新文档的同时，保留先前已适应文档中知识的能力。按照 [32]，中的在线适应评估协议，我们测量 F1 分数保留率，其定义为：在进一步适应最多 1,400 篇额外文档后，前 200 篇已适应文档的 F1 分数相对下降幅度；额外文档以每次 200 篇的步长递增。较高的保留率表示模型在吸收新信息的同时，仍能保留早期文档中的知识，表明其不易受到灾难性遗忘的影响。表3报告了保留率及相应的记忆库大小。对于所有基础模型和数据集，MBC 都实现了与 MAC 相当的高保留率，表明压缩不会损害模型在在线适应期间保留早期知识的能力。同时，MBC 所需的内存始终少得多。对于相同数量的已适应文档，与 MAC 相比，其记忆库占用空间平均减少了 97.3%。有趣的是，小型模型 DistilGPT2 似乎表现出较高的保留率。然而，这种现象源于其有效利用记忆库的能力有限：它的绝对性能本来就较低，因此保留率显得虚高。与此同时，大型模型 GPT2-XL 和 LLaMA-2-7B 实现了出色的适应性能和较高的保留率，证实 MBC 能够有效扩展至大型基础大语言模型。

4.4.4 码本重置机制的有效性。我们进一步通过比较采用与未采用……的训练过程，评估第3.3节介绍的基于 EMA 的码本重置机制所发挥的作用

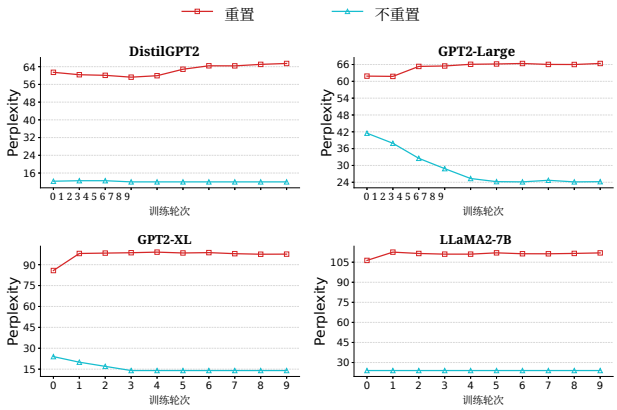


图2：在所有基础大语言模型上，MBC采用和未采用码本重置时，StreamingQA 困惑度随训练轮次的变化。

以及在 MBC 中不进行重置。代码使用率通过困惑度衡量，定义为 PPL $=\exp(-\sum_k \bar{p}_k \log \bar{p}_k)$，其中 $\bar{p}_k = u_k / \sum_j u_j$，而 u_j 是式 1 中定义的经指数移动平均平滑后的使用率。较低且保持平稳的数值表明码本坍塌，这意味着只有一小部分码字被反复使用。图 2展示了四种基础大语言模型在 StreamingQA 上的困惑度曲线。重置时，有效代码使用率在整个训练过程中保持稳定且多样。对于 DistilGPT2，前 10 个训练轮次的困惑度介于 57 到 65 之间，而不进行重置时则会降至接近 12。对于 GPT2-Large，重置可使困惑度维持在 61–66 之间，而不进行重置时则会迅速降至 24。对于 GPT2-XL，重置可使困惑度稳定在 90 以上，而不进行重置时则会降至 14。同样，对于 LLaMA-2-7B，重置可使困惑度维持在 100 以上，而不进行重置时码本则会再次坍塌至 24。这些结果证实，码本重置机制对于防止坍塌并确保

代码使用率均衡至关重要，这有助于所提出的 MBC 方法实现稳定训练和有效适应。

5 结论

在本研究中，我们解决了记忆增强型大语言模型的可扩展性挑战：随着新文档不断处理，记忆库也持续增长。我们提出了 MBC，这是一种压缩记忆库的模型，能够在流式场景中实现大语言模型的高效持续适应。MBC 将基于码本的压缩与在线重置机制相结合，可防止码本崩溃并确保码本利用率均衡。同时，轻量级 KV-LoRA 模块可在注意力机制中实现有针对性的适应，使模型无需全量微调即可高效利用查询-记忆调制。这种设计使 MBC 在提高问答准确率的同时，实现了记忆效率方面的可扩展性。使用问答数据集进行的实验表明，MBC 提高了精确匹配和 F1 分数，同时将记忆库占用空间降低至最具竞争力基线方法的 0.3%。MBC 在在线适应过程中也保持了较高的 F1 分数保留率，从而减少了灾难性遗忘。一个值得探索的未来方向是扩展 MBC，纳入强化信号以自适应地引导记忆使用 [13]，或探索支持联邦持续学习的分布式记忆库 [42]。

参考文献

[1] Bengio、N. Léonard 和 A. Courville。2013年。《通过随机神经元估计或传播梯度以实现条件计算》。(2013年)。arXiv: 1308.3432.[2] A. Chaudhry、M. Rohrbach、M. Elhoseiny、T. Ajanthan、P. K. Dokania、P. H. S. Torr 和 M'A. Ranzato。2019年。《持续学习中的微型情景记忆》。(2019年)。arXiv: 1902.10486。[3] T. Dettmers、A. Pagnoni、A. Holtzman 和 L. Zettlemoyer。2023年。Qlora：量化大语言模型的高效微调。*NeurIPS*，36，10088–10115。[4] T. Gunter 等。2024年。Apple Intelligence Foundation Language Models。(2024年)。arXiv: 2407.21075.[5] K. Guu、K. Lee、Z. Tung、P. Pasupat 和 M. Chang。2020年。《检索增强语言模型预训练》。载于*ICML*。PMLR，3929–3938。[6] Z. He、L. Karlinsky、D. Kim、J.McAuley、D.Krotov 和 R. Feris。2024年。CAMELoT：迈向具有免训练整合式联想记忆的大语言模型。(2024年)。arXiv: 2402.13449。[7] N. Houlsby、A. Giurgiu、S. Jastrzebski、B. Morrone、Q. D. Laroussilhe、A. Ges- mundo、M. Attariyan 和 S. Gelly。2019年。自然语言处理的参数高效迁移学习。载于*ICML*。PMLR，2790–2799。[8] E. J. Hu、Y. Shen、P. Wallis、Z. Allen-Zhu、Y. Li、S. Wang、L. Wang 和 W. Chen。2021年。LoRA：大语言模型的低秩适配。(2021年)。arXiv: 2106.09685.[9] N. Hu、E. Mitchell、C. D. Manning 和 C. Finn。2023年。语言模型的元学习式在线适应。(2023年)。arXiv: 2305.15076。[10] J. Jang、S. Ye、S. Yang、J. Shin、J. Han、G. Kim、S. J. Choi 和 M. Seo。2022年。迈向语言模型的持续知识学习。(2022年)。arXiv: 2110.03215.[11] H. Kim、A. Mnih、J. Schwarz、M. Garnelo、A. Eslami、D. Rosenbaum、O. Vinyals 和 Y. W. Teh。2019年。注意神经过程。(2019年)。arXiv: 1901.05761。[12] D. P. Kingma 和 J. Ba。2017年。Adam：一种随机优化方法。(2017年)。arXiv: 1412.6980。[13] M. Kulkarni、P. Tangarajan、K. Kim 和 A. Trivedi。2024年。用于优化领域聊天机器人检索增强生成的强化学习。(2024年)。arXiv: 2401.06800。[14] P. Lewis 等。2020年。面向知识密集型自然语言处理任务的检索增强生成。载于*NeurIPS*。第33卷。Curran Associates, Inc.，9459–9474。[15] X. L. Li 和 P. Liang。2021年。前缀调优：优化用于生成的连续提示。(2021年)。arXiv: 2101.00190。[16] A. Liska 等。2022年。StreamingQA：问答模型随时间适应新知识的基准测试。载于*ICML*。PMLR，13604–13622。[17] X. Liu、K. Ji、Y. Fu、W. L. Tam、Z. Du、Z. Yang 和 J. Tang。2022年。P-Tuning v2：提示调优在各种规模和任务上均可媲美微调。(2022年)。arXiv: 2110.07602。[18] M. McCloskey 和 N. J. Cohen。1989年。连接主义网络中的灾难性干扰：序列学习问题。载于《学习与动机心理学》。第24卷。Elsevier，109–165。isbn: 978-0-12-543324-2。[19] A. Misrahi、N. Chirkova、M. Louis 和 V. Nikoulina。2025年。面向多领域检索增强生成的大语言模型适配。(2025年)。arXiv: 2504.02411.[20] OpenAI 等。2024年。GPT-4技术报告。(2024年)。arXiv: 2303.08774。[21] S. Park 和 J. Bak。2024年。Memoria：通过受人类启发的记忆架构解决致命遗忘问题。(2024年)。arXiv: 2310.03052。[22] J. Pfeiffer、A. Kamath、A. Rücklé、K. Cho 和 I. Gurevych。2021年。AdapterFusion：用于迁移学习的非破坏性任务组合。(2021年)。arXiv: 2005 .00247。[23] A. Radford、J. Wu、R. Child、D. Luan、D. Amodei 和 I. Sutskever。 [n. d.] 语言模型是无监督多任务学习器。[24] C. Raffel、N. Shazeer、A. Roberts、K. Lee、S. Narang、M. Matena、Y. Zhou、W. Li 和 P. J. Liu。2020年。探索使用统一文本到文本转换器进行迁移学习的极限。JMLR，21，140，1–67。[25] P. Rajpurkar、J. Zhang、K. Lopyrev 和 P. Liang。2016年。SQuAD：用于文本机器阅读理解的100,000+ 个问题。(2016年)。arXiv: 1606.05250。[26] E. Sandhaus。2008年。《纽约时报》标注语料库。语言数据联盟，费城，6，12，e26752。[27] V. Sanh、L. Debut、J. Chaumond 和 T. Wolf。2019年。Distilbert：bert的蒸馏版本，更小、更快、更经济、更轻量。载于NeurIPS EMC^2 Workshop。[28] J. Schwarz、W. Czarnecki、J. Luketina、A. Grabska-Barwinska、Y. W. Teh、R. Pascanu 和 R. Hadsell。2018年。Progress & Compress：一种可扩展的持续学习框架。载于ICML。PMLR，4528–4537。[29] N. Shazeer 和 M. Stern。2018年。Adafactor：以内存开销次线性增长为特点的自适应学习率。载于ICML。PMLR，4596–4604。[30] H. Shi、Z. Xu、H. Wang、W. Qin、W. Wang、Y. Wang、Z. Wang、S. Ebrahimi 和 H. Wang。2025年。大型语言模型的持续学习：综合综述。ACM计算调查，3735633。[31] K. Singhal 等。2025年。借助大语言模型实现专家级医学问答。《自然·医学》，31，3，943–950。[32] J. Tack、J. Kim、E. Mitchell、J. Shin、Y. W. Teh 和 J. R. Schwarz。2024年。基于摊销上下文记忆的语言模型在线适应。NeurIPS，37，130109–130135。[33] H. Touvron 等。2023年。Llama 2：开放式基础模型与微调聊天模型。(2023年)。arXiv: 2307.09288。[34] A. van den Oord、O. Vinyals 和 k. kavukcuoglu。2017年。神经离散表征学习。载于NeurIPS。第30卷。Curran Associates, Inc。[35] D. Van Veen 等。2024年。经过适配的大语言模型在临床文本摘要方面能够胜过医学专家。《自然·医学》，30，4，1134–1142。[36] A. Vaswani、N. Shazeer、N. Parmar、J. Uszkoreit、L. Jones、A. N. Gomez、Ł. Kaiser 和 I. Polosukhin。2017年。注意力就是一切。载于NeurIPS。第30卷。Curran Associates, Inc。[37] J. Wang、A. Jatowt 和 M. Yoshikawa。2022年。ArchivalQA：面向历史新闻集开放域问答的大规模基准数据集。载于SIGIR。ACM，西班牙马德里，3025–3035。isbn: 978-1-4503-8732-3。[38] Y. Wang、J. Zhang、T. Shi、D. Deng、Y. Tian 和 T. Matsumoto。2024年。大语言模型交互式机器翻译的最新进展。IEEE Access，12，179353–179382。[39] Y. Wang 等。2024年。MEMORYLLM：迈向可自我更新的大型语言模型。(2024年)。arXiv: 2402.04624。[40] S. Wu 等。2024年。自然语言处理中的检索增强生成：综述。(2024年)。arXiv: 2407.13193。[41] L. Xu、H. Xie、S-Z. J. Qin、X. Tao 和 F. L. Wang。2023年。预训练语言模型的参数高效微调方法：批判性评述与评估。(2023年)。arXiv: 2312.12148。[42] X. Yang、H. Yu、X. Gao、H. Wang、J. Zhang 和 T. Li。2024年。基于知识融合的联邦持续学习：综述。IEEE Transactions on Knowledge and Data Engineering，36，8，3832–3850。[43] Y. Yang、J. Zhou、X. Ding、T. Huai、S. Liu、Q. Chen、Y. Xie 和 L. He。2025年。基于基础语言模型的持续学习最新进展：综述。ACM计算调查，57，5，1–38。[44] S. Yao、D. Yu、J. Zhao、I. Shafran、T. Griffiths、Y. Cao 和 K. Narasimhan。2023年。思维树：利用大语言模型进行审慎问题求解。载于NeurIPS。第36卷。Curran Associates, Inc.，11809–11822。[45] Z. Zhang、Q. Dai、X. Bo、C. Ma、R. Li、X. Chen、J. Zhu、Z. Dong 和 J-R. Wen。2025年。基于大语言模型的智能体记忆机制综述。ACM信息系统汇刊，3748302。[46] S. Zhao、Y. Shao、Y. Huang、J. Song、Z. Wang、C. Wan 和 L. Ma。2025年。理解检索增强生成系统的设计决策。(2025年)。arXiv: 2411.19463。[47] Y. Zhu 等。2024年。信息检索中的大语言模型：综述。(2024年)。arXiv: 2308.07107。