arXiv:2602.19169v1 [cs.LG] 2 Dec 2025

虚拟参数锐化：用于推理时推理增强的动态低秩扰动

Saba Kublashvili独立研究者GitHub：
Saba-Kublashvili/vps-virtual-parameter-synthesis

摘要

我们提出了**虚拟参数锐化（VPS）**，这是一种推理时技术，通过动态的、激活条件化的低秩扰动来增强冻结的 Transformer 线性层。与 LoRA 等参数高效微调方法不同，后者学习静态低秩适配器，而 VPS 根据批次激活统计信息和可选的梯度信号即时构造扰动因子，从而无需持久化参数更新即可实现测试时自适应。扰动形式为 $\Delta W = \gamma \cdot W^\top V U^\top W$，其中选择矩阵 U 和 V 通过稀疏激活引导选择或 Sylvester 耦合回归构造。我们对扰动的谱性质进行了理论分析，并介绍了一种自适应策略系统，该系统根据激活能量和词元级熵调节扰动幅度。对于具有真实标注监督的任务，该系统结合多目标验证与迭代细化。我们给出了完整的算法框架，分析了其数学基础，并探讨了激活条件计算增强大语言模型推理能力的机制。实现和实验代码可在以下地址获取：
h
t
t
p
s://
github.com/Saba-Kublashvili/vps-virtual-parameter-synthesis。

关键词：推理时自适应、低秩扰动、推理增强、大语言模型、动态计算

1 引言

大语言模型（LLM）在各种推理任务中展现出卓越的能力 [Brown 等人，2020年，Wei 等人，2022年，Kojima 等人， 2022]。然而，要有效部署这些模型，通常需要针对特定任务进行适配，传统上通过微调或提示工程来实现

。LoRA 等参数高效微调方法 [Hu 等人， 2021] 以及适配器 [Houlsby 等人， 2019] 已成为全量微调的实用替代方案，它们在冻结的基础权重上学习紧凑的修改。这些方法虽然有效，但仍需在训练阶段使用特定任务的数据，并会产生静态的参数修改。

另一个互补的研究方向探索推理时适配，即在生成过程中动态修改模型行为，而无需持久化的参数变更。测试时训练 [Sun 等人、 2020]，上下文学习 [Brown 等人以及 2020]，自一致性解码 [Wang 等人， 2023] 代表了这一范式的不同方法。推理时方法的吸引力在于其灵活性：同一个基础模型可以针对不同输入表现出不同的行为，而无需维护单独的参数检查点。

在本文中，我们提出了**虚拟参数锐化（VPS）**，这是一种推理时技术，通过利用激活统计信息构建的动态低秩扰动来增强 Transformer 线性层。VPS 的关键区别特征包括：

1. **依赖权重的扰动结构**：不同于加性 LoRA 适配器（$W + BA$），VPS 构建出形如 $\Delta W \propto WUV^\top W$ 的扰动，确保修改遵循现有权重空间的几何结构。2. **激活引导的稀疏性**：选择器矩阵 U 和 V 通过识别高激活的输入和输出维度构建，将扰动容量集中于与任务相关的特征子空间。3. **自适应策略系统**：扰动幅度和秩根据激活能量与词元级熵进行调节，从而提供依赖输入的计算深度。4. **迭代验证优化**：对于具有真实标签监督的任务，多目标验证器向策略系统提供反馈，使其能够在多个推理轮次中不断改进。

本文其余部分安排如下。第2节回顾参数高效自适应和推理时方法的相关工作。第3 节介绍完整的 VPS 框架，包括数学表述、构建算法和自适应策略系统。第4节对谱性质和扰动界进行理论分析。第5 节介绍实现细节以及与 Transformer 架构的集成。第6 节介绍实验框架。第7节讨论相关机制和局限性。第8 节总结全文。

2 相关工作

2.1 参数高效微调

参数高效微调（PEFT）方法只修改模型的一小部分参数，同时冻结大多数参数。适配器 [Houlsby 等人，2019年，Pfeiffer 等人， 2020] 会在 Transformer 层之间插入小型可训练模块。LoRA [Hu 等人， 2021] 将权重更新参数化为低秩矩阵 $\Delta W = BA$，其中 $B \in \mathbb{R}^{d_{out} \times r}$ 且 $A \in \mathbb{R}^{r \times d_{in}}$，并满足$r \ll \min(d_{in}, d_{out})$。后续工作通过自适应秩分配 [Zhang 等人， 2023], 量化感知训练 [Dettmers 等人，2023], 以及动态组合 [Liu 等人， 2024]扩展了 LoRA。

VPS 与这些方法存在根本差异：PEFT 方法在训练阶段学习静态参数修改，而 VPS 则在推理时动态构造扰动，不依赖学习得到的参数。扰动结构也有所不同——VPS 使用依赖权重的构造方式，而非彼此独立的加性因子。

2.2 测试时自适应

测试时自适应（TTA）利用测试输入本身的信息，在推理过程中改变模型行为。 Sun 等人 [2020]提出更新批归一化统计量，并在测试样本上最小化熵。 Wang 等人 [2021] 进一步针对领域偏移发展了熵最小化方法。在语言模型领域，Bansal 等人 [2023] 探索了在测试输入本身上进行测试时训练。

VPS 同样采用测试时范式，但它通过激活条件扰动发挥作用，而非显式进行参数更新。这样便无需针对每个输入都通过整个模型执行反向传播，从而避免了相应的计算开销。

2.3 动态与条件计算

动态网络根据输入特征调节计算。混合专家（MoE）架构 [Shazeer 等人, 2017年, Fedus 等人,2022]将 token 路由到不同的专家子网络。早退机制[Schwartz 等人, 2020年, Schuster 等人， 2022] 允许可变的计算深度。超网络 [Ha 等人, 2016] 根据辅助输入生成网络权重。

VPS 实现了一种条件计算形式：应用于每个线性层的扰动取决于当前批次的激活统计信息。这使模型能够针对每个输入进行自适应，而无需采用 MoE 的离散路由决策，也不会产生超网络的计算开销。

2.4 自洽性与验证

Wang 等人 [2023] 证明，对多条推理路径进行采样并通过多数投票进行聚合，可以提升推理任务的表现。后续工作探索了更复杂的聚合方式 [Li 等人，2023]以及迭代细化 [Madaan 等人，2023年，Shinn 等人， 2023]。VPS 纳入了一种验证机制，可在推理迭代期间提供反馈，通过扰动调整而非显式重新提示来实现一种迭代细化形式。

3 方法

3.1 问题设定

考虑一个具有冻结参数的预训练 Transformer 模型。对于权重矩阵为 $W \in \mathbb{R}^{d_{out} \times d_{in}}$ 的线性层，标准前向传播针对输入 $x \in \mathbb{R}^{d_{in}}$ 计算 $y = Wx$ 。我们的目标是构造动态扰动 $\Delta : \mathbb{R}^{d_{in}} \to \mathbb{R}^{d_{out}}$ ，使得修改后的计算 $\tilde{y} = Wx + \Delta(x)$ 在提升任务性能的同时：

1. 保持计算效率（关于 $d_{in} \cdot d_{out}$ 为次线性）2. 保持稳定性（扰动幅度有界）3. 适应输入特征（激活条件化）

3.2 VPS 前向传播

线性层的 VPS 增强前向传播为：

$$\tilde{y} = Wx + \gamma \cdot (xA)B^\top \tag{1}$$

其中：

- $A \in \mathbb{R}^{d_{in} \times r}$ 和 $B \in \mathbb{R}^{d_{out} \times r}$ 是动态构造的低秩因子
- $\gamma \in [0, 1]$ 是缩放系数（自适应或固定）
- $r \ll \min(d_{in}, d_{out})$ 是扰动秩

因子 A 和 B 由选择器矩阵 $U \in \mathbb{R}^{d_{in} \times r}$ 和$V \in \mathbb{R}^{d_{out} \times r}$ 通过以下方式推导得出：

$$A = W^\top V, \quad B = WU \tag{2}$$

将（2）代入（1）并重新整理：

$$\begin{aligned}
\tilde{y} &= Wx + \gamma \cdot x(W^\top V)(WU)^\top \\
&= Wx + \gamma \cdot xW^\top VU^\top W^\top \\
&= \left(W + \gamma \cdot (WUV^\top W)^\top\right) x \\
&= \left(W + \gamma \cdot W^\top VU^\top W\right)^\top x
\end{aligned} \tag{3}$$

注 3.1 (权重依赖结构)。 有效扰动 $\Delta W = \gamma \cdot W^\top V U^\top W$ 是冻结权重 W 的函数。这与 *LoRA* 的加性结构 $\Delta W = BA$ 有根本区别，在该结构中， B、 A 独立于 W。*VPS* 扰动作用于 W 的列空间和行空间，修改现有权重结构，而非添加正交分量。

3.3 选择器构造：构建算法

核心问题是如何构造 U 和 V。我们提出了三种复杂程度逐步提升的构建算法。

3.3.1 SK 构建器：通过激活幅值构建稀疏选择器

SK（稀疏-k）构建器通过识别高激活维度来构造独热选择器矩阵。

算法 1 SK 构建器

要求：批次激活 $X \in \mathbb{R}^{N \times d_{in}}$、线性层 $W \in \mathbb{R}^{d_{out} \times d_{in}}$、参数 k, r1：计算输入激活得分： $s_j^{(in)} = \frac{1}{N} \sum_{i=1}^{N} |X_{ij}|$ ，针对 $j \in [d_{in}]$2：通过基础层计算输出： $H = XW^\top \in \mathbb{R}^{N \times d_{out}}$3：计算输出激活得分： $s_\ell^{(out)} = \frac{1}{N} \sum_{i=1}^{N} |H_{i\ell}|$ ，针对 $\ell \in [d_{out}]$4：选择 Top-k 个输入索引： $\mathcal{I}_{in} =$ top-k($s^{(in)}, k$) 5：选择 Top-k 个输出索引： $\mathcal{I}_{out} =$ top-k($s^{(out)}, k$) 6：将 $U \in \mathbb{R}^{d_{in} \times r}$ 和 $V \in \mathbb{R}^{d_{out} \times r}$ 初始化为零矩阵 7：对于 $c = 1$ 到 r ，执行 8：$U_{\mathcal{I}_{in}[c],c} \leftarrow 1$9：$V_{\mathcal{I}_{out}[c],c} \leftarrow 1$10：结束循环 11：返回 U, V

原理：通过选择激活幅值较高的维度，SK 构建器将扰动容量集中用于当前输入中最活跃的特征子空间。激活接近于零的维度对输出的贡献极小；对这些维度施加扰动会浪费容量。

复杂度：SK 构建器在计算激活时需要 $O(Nd_{in} + Nd_{out})$，在进行 top-k 选择时需要 $O(d_{in} \log k + d_{out} \log\ k)$，其复杂度随批次大小和层维度线性增长。

3.3.2 SC 构建器：Sylvester 耦合细化

SC（Sylvester-Coupled，Sylvester 耦合）构建器通过求解岭回归来改进 SK 选择，该回归将输入激活与输出激活耦合起来

选定子空间。

算法 2 SC 构建器

需要： 批次激活 X、线性层 W、参数 k, r、正则化-项 α

从 SK 构建器（算法 1）获取初始值 U, V
提取选定的输入激活： $X_A = X[:, \mathcal{I}_{in}] \in \mathbb{R}^{N \times r}$
提取选定的输出激活： $Y = H[:, \mathcal{I}_{out}] \in \mathbb{R}^{N \times r}$
计算格拉姆矩阵： $G = X_A^\top X_A \in \mathbb{R}^{r \times r}$
计算交叉协方差： $C = X_A^\top Y \in \mathbb{R}^{r \times r}$
求解岭回归方程： $T = (G + \alpha I)^{-1} C \in \mathbb{R}^{r \times r}$
更新输出选择器： $\tilde{V} = V T^\top$
列归一化： $\tilde{V}_{:,c} \leftarrow \tilde{V}_{:,c} / \|\tilde{V}_{:,c}\|_2$ 对于每个 c
返回 U , $\tilde{V}$

原理：岭回归会找到一个线性变换 T，以便根据选定的输入激活最佳地预测选定的输出激活：

$$T^* = \arg\min_T \|X_A T - Y\|_F^2 + \alpha \|T\|_F^2 \tag{4}$$

解 $T^* = (X_A^\top X_A + \alpha I)^{-1} X_A^\top Y$刻画了选定子空间内的局部输入输出耦合。将 $T^\top$ 应用于 V 会旋转输出选择器，使其与可从输入预测的方向对齐，从而可能提高扰动的相关性。

复杂度：额外开销为 $O(Nr^2 + r^3)$，用于构建格拉姆矩阵并求解 $r \times r$ 线性方程组。对于较小的 r （通常为2-8），该开销可忽略不计。

3.3.3 混合构造器

混合构建器根据可用信息在 SK 和 SC 之间进行自适应选择：

$$\text{Builder}(X, H, W, g) = \begin{cases} \text{SC}(X, H, W) & \text{if } g \neq \emptyset \\ \text{SK}(X, H, W) & \text{otherwise} \end{cases} \tag{5}$$

其中， g 表示梯度信息（若可从验证损失中获取）。当梯度信号表明处于优化上下文时，将采用更复杂的 SC 耦合。

3.4 谱范数控制

为防止扰动导致前向传播不稳定，我们对构造的因子进行逐分量谱裁剪。

定义 3.2（逐分量裁剪）。 对于因子 $A \in \mathbb{R}^{d_{in} \times r}$ 和$B \in \mathbb{R}^{d_{out} \times r}$ 以及阈值 $\tau > 0$，定义裁剪操作：

$$Clip_\tau(A, B) = (\tilde{A}, \tilde{B}) \tag{6}$$

其中对于每一列 $c \in [r]$:

$$\sigma_c = \|A_{:,c}\|_2 \cdot \|B_{:,c}\|_2, \quad s_c = \max\left(1, \frac{\sigma_c}{\tau}\right) \tag{7}$$

$$\tilde{A}_{:,c} = \frac{A_{:,c}}{\sqrt{s_c}}, \quad \tilde{B}_{:,c} = \frac{B_{:,c}}{\sqrt{s_c}} \tag{8}$$

˜ ˜这确保$\|A_{:,c}\|_2 \cdot \|B_{:,c}\|_2 \leq \tau$ 对所有 c 成立，从而约束扰动每个秩一分量的谱范数。

3.5 自适应策略系统

策略系统根据依赖于输入的统计量调节扰动超参数。

3.5.1 激活能量

对于隐藏激活 $H = XW^\top \in \mathbb{R}^{N \times d_{out}}$，定义批次能量：

$$E = \frac{1}{N \cdot d_{out}} \sum_{i,j} H_{ij}^2 = \frac{\|H\|_F^2}{N \cdot d_{out}} \tag{9}$$

能量通过饱和非线性映射为缩放因子：

$$\sigma = 1 - e^{-E} \tag{10}$$

这会将 $E \in [0, \infty)$ 映射为 $\sigma \in [0, 1)$：低能量对应较小的 σ(最小扰动)，高能量对应 $\sigma \approx 1$ (完整扰动)。

3.5.2 词元熵

对于生成步骤中的输出逻辑值 $z \in \mathbb{R}^V$（其中 V 是词汇表大小），定义词元熵：

$$\mathcal{H}(z) = -\sum_{v=1}^{V} p_v \log p_v, \quad p = \text{softmax}(z) \tag{11}$$

调整缩放因子以纳入熵：

$$\sigma \leftarrow \max\left(\sigma, \min\left(1, \frac{\mathcal{H}(z)}{3}\right)\right) \tag{12}$$

原理：高熵意味着模型不确定性较高——概率质量分散在许多词元上。在不确定状态下，增大扰动幅度可能有助于模型区分候选项，这类似于提高模拟退火中的温度，以逃离局部最优。

3.5.3 自适应超参数

给定边界 $[r_{lo}, r_{hi}]$、$[\gamma_{lo}, \gamma_{hi}]$ 等，超参数通过插值确定：

$$r = r_{lo} + \lfloor (r_{hi} - r_{lo}) \cdot \sigma \rfloor \tag{13}$$

$$\gamma = \gamma_{lo} + (\gamma_{hi} - \gamma_{lo}) \cdot \sigma \tag{14}$$

这提供了平滑且依赖输入的超参数调度，无需进行离散模式切换。

3.5.4 改进历史

该策略会跟踪近期扰动是否改善了验证指标：

$$\text{improved}_t = \mathbf{1}[\mathcal{L}_t < \mathcal{L}_{t-1}] \tag{15}$$

近期改进所占的比例会影响后续缩放：

$$\rho = \frac{1}{|\mathcal{W}|} \sum_{t \in \mathcal{W}} \text{improved}_t \tag{16}$$

其中，$\mathcal{W}$ 表示近期迭代次数的滑动窗口。较高的 ρ会增大缩放因子；较低的 ρ 则会减小缩放因子，从而实现一种简单的在线自适应。

3.6 多目标验证

对于具有真实值监督的任务，我们采用复合验证器，计算多个目标的加权损失。

定义 3.3 (复合验证损失)。 给定预测文本 $\hat{y}$、真实值 y^*和权重 $\{w_i\}$：

$$\mathcal{L}(\hat{y}, y^*) = \sum_i w_i \cdot \ell_i(\hat{y}, y^*) \tag{17}$$

实现的目标如下：

数值损失：使用模式匹配从 $\hat{y}$ 和 y^* 中提取数值，并计算平方误差：

$$\ell_{num} = (\text{extract}(\hat{y}) - \text{extract}(y^*))^2 \tag{18}$$

单位一致性损失：使用量纲分析解析带单位的量，并对不一致的量纲进行惩罚：

$$\ell_{unit} = \mathbf{1}[\dim(\hat{y}) \neq \dim(y^*)] \tag{19}$$

代数形式损失：对表达式进行符号解析，检查结构等价性：

$$\ell_{alg} = \mathbf{1}[\text{simplify}(\hat{y} - y^*) \neq 0] \tag{20}$$

自一致性损失：以温度 $T > 0$采样 n 个回答，计算方差：

$$\ell_{sc} = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2/(\bar{y}^2 + \epsilon) \tag{21}$$

验证损失为策略更新提供信号，并支持在多个推理轮次中进行迭代细化。

3.7 迭代细化循环

算法 3 展示了包含迭代细化的完整 VPS推理流程。

Algorithm 3 VPS 推nf理　　并进 erative $\text{Refi}_{\text{nemen}}$ t

要求：提示词 p，真实值 y^* （可选），迭代次数 T，模型 $\mathcal{M}$1：初始化策略状态， $\mathcal{L}_{prev} \leftarrow \infty$2：$\hat{}y_0 \leftarrow$ 生成（$\mathcal{M}$， p）{基线生成} 3：如果 $y^* = \emptyset$ 或 $T < 2$ 则 4：返回 $\hat{}y_0$5：结束 if 6：对于 $t = 1$ 到 $T - 1$ 执行 7：清空梯度缓冲区和 L-BFGS记忆 8：执行前向传播以计算词元熵 $\mathcal{H}$9：使用熵更新策略：Policy.set_entropy($\mathcal{H}$) 10：生成预测：$\hat{}y_t \leftarrow$ 生成（$\mathcal{M}$， p） 11：计算验证损失： $\mathcal{L}_t \leftarrow$ Verify($\hat{}y_t$， y^*) 12：根据结果更新策略：Policy.update($\mathcal{L}_t < \mathcal{L}_{prev}$) 13：$\mathcal{L}_{prev} \leftarrow \mathcal{L}_t$14：可选：为 SC 构建器计算梯度替代量 15：结束 for 16：返回 $\hat{}y_{T-1}$

3.8 注意力中的 Q/K 耦合

在自注意力层中，查询和键投影通过注意力分数计算相互作用：

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V \tag{22}$$

其中 $Q = XW_Q$ 和 $K = XW_K$。

对 W_Q 和 W_K 进行独立扰动可能会破坏查询和键表示之间的对齐。VPS 可选择配对查询和键投影层：

$$\text{VPS}_{W_Q}.\text{peer} = \text{VPS}_{W_K}, \quad \text{VPS}_{W_K}.\text{peer} = \text{VPS}_{W_Q} \tag{23}$$

这种配对支持协同选择器构造，确保对 Q和 K 的扰动能够保留它们之间的几何关系。

4 理论分析

4.1 扰动界

命题 4.1（谱范数界）。设 $\Delta = AB^\top$ 其中 $A \in \mathbb{R}^{d_{in}\times r}$，$B \in \mathbb{R}^{d_{out}\times r}$，且 $\mathit{Clip}_\tau(A,\ B) = (\tilde{A},\ \tilde{B})$ 采用第3.4节中的裁剪操作。则：

$$\|\tilde{A}\tilde{B}^\top\|_2 \le \sqrt{r}\cdot\tau \tag{24}$$

证明。该扰动可以写成秩1项之和：

$$\tilde{A}\tilde{B}^\top = \sum_{c=1}^{r} \tilde{A}_{:,c}\tilde{B}_{:,c}^\top \tag{25}$$

根据谱范数的三角不等式：

$$\|\tilde{A}\tilde{B}^\top\|_2 \le \sum_{c=1}^{r} \|\tilde{A}_{:,c}\|_2\|\tilde{B}_{:,c}\|_2 \le r\cdot\tau \tag{26}$$

利用弗罗贝尼乌斯范数关系可以得到更紧的界：

$$\|\tilde{A}\tilde{B}^\top\|_2 \le \|\tilde{A}\tilde{B}^\top\|_F = \sqrt{\sum_{c,c'} (\tilde{A}_{:,c}^\top\tilde{A}_{:,c'})(\tilde{B}_{:,c}^\top\tilde{B}_{:,c'})} \tag{27}$$

对于正交列或近正交列（稀疏 SK 构造会鼓励这种情况），交叉项消失，从而得到：

$$\|\tilde{A}\tilde{B}^\top\|_F \approx \sqrt{\sum_{c=1}^{r} (\|\tilde{A}_{:,c}\|_2\|\tilde{B}_{:,c}\|_2)^2} \le \sqrt{r}\cdot\tau \tag{28}$$

□

推论 4.2. 扰动输出满足：

$$\|\tilde{y}-y\|_2 = \gamma\|(xA)B^\top\|_2 \le \gamma\|x\|_2\|A\|_2\|B\|_2 \le \gamma\sqrt{r}\tau\|x\|_2 \tag{29}$$

这表明，扰动幅度由 γ、r 和 τ 控制，它们都是可调超参数。

4.2 有效秩与表达能力

命题 4.3 (选择器秩)。 设 U、 V 由 *SK* 构建器通过 *top-*k 选择构造，目标秩为 $r \leq k$。若所选索引互不相同（没有重复的激活值），则 $\mathit{rank}(U) = \mathit{rank}(V) = r$。

证明。 根据构造， U 和 V 恰好各有 r 个非零元素，且每个元素均为 1，分别位于不同的行。因此，各列线性无关，秩为 r。

□

有效扰动 $\Delta W = \gamma W^\top V U^\top W$的秩至多为 r，但其对输入空间的作用取决于选定维度与权重矩阵结构之间的对齐情况。

4.3 与投影算子的联系

当 U 和 V 是稀疏独热选择器时，定义投影矩阵：

$$\Pi_U = UU^\top \in \mathbb{R}^{d_{in} \times d_{in}}, \quad \Pi_V = VV^\top \in \mathbb{R}^{d_{out} \times d_{out}} \tag{30}$$

这些是仅在选定索引处取值为 1 的对角矩阵。扰动可改写为：

$$\Delta W = \gamma W^\top \Pi_V \Pi_U W = \gamma (P_V W)(P_U W)^\top \tag{31}$$

其中 P_V, P_U 为相应的选择算子。这表明，VPS 会选择性地放大特定输入与输出特征维度之间的交互，并由现有的权重结构加以介导。

5 实现

5.1 VPSLinear 模块

核心实现封装了一个标准的 nn.Linear 模块：

```
class VPSLinear ( nn . Module ): def __init__ ( self , base : nn . Linear , cfg ): self . base =base # 冻结权重self . builder =make_builder ( cfg . builder , cfg ) self . policy =VPSPolicy ( cfg ) def forward ( self , x ): y_base =self . base (x ) x2d , h2d =展平 ( x ) , flatten ( y_base . 分离 ())# 构造选择器U , V =self . builder ( x2d , h2d , self . base )
```

```
        # 推导低秩因子A =self . base . weight . t () @ V #
    ( 输入 , 秩 )B =self . base . weight @ U # ( 输出 , 秩 ) #
    谱裁剪 A , B = 谱 _ 剪裁 (A , B , 陶 )# 计算扰动 delta =
    ( x2d @ A ) @ B . t ()# 应用策略确定的伽马值pol =
    self . policy . decide ( h2d ) return y_base +pol .
    gamma * delta
```

5.2 模型补丁

VPS 通过递归替换目标线性层，应用于现有的 HuggingFace 模型：

```
def 修补_模型_, 使用_VPS（模型, apply_to , cfg ):
    for name , 模型中的子项。named_children ():
        使用_vps 对_模型_进行补丁处理 ( child , 应用_至 , cfg )
        如果 isinstance (child , nn . Linear ):
            如果匹配_目标（名称, 应用_到）:
                设置属性（模型, 名称, VPSLinear（子项, 配置））
    返回模型
```

目标层通过名称模式指定（例如，q_proj、k_proj、up_proj）。

5.3 钩子系统

钩子管理器为构建器和策略捕获激活值与梯度：

```
类 HookManager:
    def attach（self , 模型 ):
        for m in 模型. modules ():
            if isinstance (m , VPSLinear ):
                m . r e g i s t e r _ f o r w a r d _ p r e _ h o o k ( 捕获_输入 )
                m . r e g i s t e r _ f u l l _ b a c k w a r d _ h o o k ( 捕获_梯度 )
```

5.4 计算开销

VPS 的逐层开销包括：

1. 激活评分： $O(N \cdot d_{in} + N \cdot d_{out})$ 2.
Top-k 选择： $O(d_{in} \log k + d_{out} \log\ k)$

3. 因子计算： $O(d_{in} \cdot d_{out} \cdot r)$ 用于 $A = W^\top V$（稀疏）

4. 扰动应用： $O(N \cdot r \cdot d_{out})$

对于典型取值（$r = 2$、 $k = 32$、 $N \leq 1024$），这会使基础线性层计算的开销增加约 5-15%。

6 实验框架

6.1 评估设置

我们使用以下地址提供的实现来评估 VPS：

https: //github.com/Saba-Kublashvili/虚拟参数合成

实验框架包括以下评估：

ARC 挑战赛 [Clark 等人， 2018]：一项多项选择科学推理基准测试。我们采用置信度筛选评估，从更大的样本池中选取最具挑战性的样例（基线置信度边际最低的样例）。

GSM8K [Cobbe 等人， 2021]：需要进行多步算术推理的小学数学文字题。

6.2 基线配置

实验使用 Qwen2.5 系列模型 [Qwen 团队， 2024] 涵盖不同规模（1.5B、3B 参数）。基线配置如下：

- 贪心解码（温度 $= 0$，用于基线）
- 标准 HuggingFace 生成流程
- 无 VPS 扰动（$\gamma = 0$）

6.3 VPS 配置

VPS 配置会对注意力和 MLP 投影应用扰动：

- 目标层：q_proj、k_proj、v_proj、o_proj、up_proj、down_proj、gate_proj
- 秩： $r = 2$ （自适应范围 $[1, 4]$）
- 前-k： $k = 32$
- 伽马： $\gamma = 0.65$ （自适应范围 $[0.3, 0.8]$）
- 谱阈值： $\tau = 0.8$
- 构建器：混合（有梯度时使用 SC，否则使用 SK）
- 迭代次数：3（含验证反馈）

6.4 消融研究

代码库包含全面的消融实验配置：

- 构建器变体：SK、SC、混合
- 伽马值：0.3、0.5、0.7
- 秩值：1、2、4
- Q/K耦合：启用/禁用
- L-BFGS 预处理：启用/禁用
- 自适应与固定超参数

消融结果量化了各个组件对整体性能的贡献。实验脚本位于 scripts/run_ablations.py。

6.5 可复现性

所有实验均可通过提供的 Colab 笔记本和脚本复现。随机种子均为固定值，以确保行为确定。代码仓库包含：

- 所有 VPS 组件的完整源代码
- ARC 挑战赛和 GSM8K 的评估脚本
- 附有超参数说明的配置文件
- 用于交互式实验的 Colab 笔记本

7 讨论

7.1 为什么 VPS 能提升推理能力?

我们假设 VPS 可能通过以下几种机制提升推理性能：

激活条件计算：与静态权重不同，VPS 会根据当前输入的激活模式调节计算。不同的推理问题会激活不同的特征子空间；VPS 基于激活条件化的选择会将扰动集中于当前相关的子空间，从而可能增强任务特定信号通路，同时抑制无关噪声。

隐式集成效应：扰动的动态特性意味着每个输入都会看到略有不同的有效模型。这样无需进行多次前向传播即可实现隐式集成，有望提升模型在分布外推理模式上的稳健性。

作为搜索的迭代精炼：验证循环在输出空间中实现了一种搜索形式。通过根据

验证反馈，VPS 通过连续扰动调整而非离散采样，探索可能输出的空间。

熵感知适应：在高熵（不确定）状态下增大扰动幅度，可能有助于模型区分不同候选项。这类似于强化学习中的探索与利用权衡。

7.2 局限性

缺乏理论保证：尽管这些机制有充分依据，但没有形式化证明表明 VPS 能提升推理能力。在特定基准测试上的实证验证并不能保证泛化能力。

计算开销：动态构建选择器和进行谱裁剪会增加逐层开销。对于对延迟敏感的应用，这种开销可能难以接受。

超参数敏感性：该系统包含许多相互作用的超参数（γ, τ, r, k、构建器选择）。最佳设置可能取决于具体任务，因而需要进行调优。

验证依赖：完整的迭代细化过程需要使用真实标签提供反馈。对于开放式生成，只有基于激活的机制能够发挥作用。

L-BFGS 的应用：临时 L-BFGS 组件以启发式方式应用于规范的优化循环之外。鉴于缩放因子较小，其对性能的贡献尚不明确，且可能微乎其微。

7.3 与相关方法的比较

表 1：VPS 与相关参数修改方法的比较

方法	需要训练	动态	依赖权重
微调	Yes	No	Yes
LoRA	Yes	No	No
适配器	Yes	No	No
提示调优	Yes	No	No
VPS	No	Yes	Yes

VPS 的独特之处在于无需训练阶段，同时能够生成依赖权重、输入条件化的扰动。

8 结论

我们介绍了虚拟参数锐化（VPS），这是一种推理时技术，可通过动态、由激活

条件化的低秩扰动来增强 Transformer 线性层。主要贡献如下：

1. 一种尊重冻结权重几何结构的新型扰动结构 $\Delta W = \gamma W^\top V U^\top W$ 2. 用于构造选择器矩阵的激活引导构造算法（SK、SC、混合） 3. 一种根据激活能量和词元熵调节扰动的自适应策略系统 4. 面向监督任务的迭代细化多目标验证 5. 对谱界和扰动结构的理论分析

VPS 代表着大语言模型动态、输入条件化计算的一次进步，无需进行特定任务训练。完整的实现和实验代码可在以下地址获取：
https://github.com/Saba-Kublashvili/vps-virtual-parameter-synthesis。

未来的工作可以探索：（1）分析激活条件化扰动何时能提升泛化能力的理论问题；（2）与自洽性等其他推理时方法相结合；（3）应用于多模态模型；（4）通过元优化实现更复杂的策略学习。

致谢

作者感谢开源社区提供的基础工具，使本研究得以实现，其中包括 PyTorch、HuggingFace Transformers 以及评测基准的开发者。

参考文献

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A. 等。《语言模型是少样本学习器》。载于神经信息处理系统进展，2020年。

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., 和 Tafjord, O. 你认为自己已经解决了问答问题吗？来试试 ARC，即 AI2推理挑战。*arXiv* 预印本 *arXiv:1803.05457*，2018年。Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plap- pert, M., Tworek, J., Hilton, J., Nakano, R., 等。训练验证器来解决数学应用题。*arXiv* 预印本 *arXiv:2110.14168*，2021年。

Dettmers, T., Pagnoni, A., Holtzman, A., 和 Zettlemoyer, L. QLoRA：高效微调量化大语言模型。载于神经信息处理系统进展，2023年。Fedus, W., Zoph, B., 和 Shazeer, N. Switch Transformer：以简单高效的稀疏性扩展至万亿参数模型。机器学习研究期刊，23(120):1–39，2022年。Ha, D., Dai, A., 和 Le, Q. V. 超网络。*arXiv* 预印本 *arXiv:1609.09106*，2016年。Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., 和 Gelly, S. 自然语言处理的参数高效迁移学习。载于国际机器学习会议，2019年。Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., 和 Chen, W. LoRA：大语言模型的低秩适配。*arXiv* 预印本 *arXiv:2106.09685*，2021年。Kojima, T., Gu, S. S., Reid, M., Matsuo, Y., 和 Iwasawa, Y. 大语言模型是零样本推理器。载于神经信息处理系统进展，2022年。Li, Y., Lin, Z., Zhang, S., Fu, Q., Chen, B., Lou, J.-G., 和 Chen, W. 利用步骤感知验证器提升语言模型的推理能力。载于第*61*届计算语言学协会年会论文集，2023年。Liu, S.-Y., Wang, C.-Y., Yin, H., Molchanov, P., Wang, Y.-C. F., Cheng, K.-T., 和 Chen, M.-H. DoRA：权重分解低秩适配。*arXiv* 预印本 *arXiv:2402.09353*，2024年。Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegreffe, S., Alon, U., Dziri, N., Prabhumoye, S., Yang, Y., 等。自我精炼：通过自我反馈进行迭代精炼。载于神经信息处理系统进展，2023年。Pfeiffer, J., Kamath, A., Rücklé, A., Cho, K., 和 Gurevych, I. Adapter-Fusion：用于迁移学习的非破坏性任务组合。*arXiv* 预印本 *arXiv:2005.00247*，2020年。Qwen 团队。Qwen2.5：一系列基础模型。*arXiv* 预印本 *arXiv:2412.15115*，2024年。Schwartz, R., Stanovsky, G., Swayamdipta, S., Dodge, J., 和 Smith, N. A. 选对工具：匹配模型与实例复杂度。载于第*58*届计算语言学协会年会论文集，2020年。

Schuster, T., Fisch, A., Gupta, J., Dehghani, M., Bahri, D., Tran, V. Q., Tay, Y., 和 Metzler, D. 自信自适应语言建模。载于神经信息处理系统进展，2022年。Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., 和 Dean, J. 超大规模神经网络：稀疏门控专家混合层。载于国际学习表征会议，2017年。Shinn, N., Cassano, F., Gopinath, A., Narasimhan, K., 和 Yao, S. Reflexion：采用语言强化学习的语言智能体。载于神经信息处理系统进展，2023年。Sun, Y., Wang, X., Liu, Z., Miller, J., Efros, A. A., 和 Hardt, M. 通过自监督测试时训练，实现分布偏移下的泛化。载于国际机器学习会议，2020年。Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., 和 Zhou, D. 自洽性提升语言模型中的思维链推理能力。载于国际学习表征会议，2023年。Wang, D., Shelhamer, E., Liu, S., Olshausen, B., 和 Darrell, T. Tent：通过熵最小化实现完全测试时自适应。载于国际学习表征会议，2021年。Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., 等。思维链提示能够激发大语言模型的推理能力。载于神经信息处理系统进展，2022年。Zhang, Q., Chen, M., Bukharin, A., He, P., Cheng, Y., Chen, W., 和 Zhao, T. AdaLoRA：用于参数高效微调的自适应预算分配。载于国际学习表征会议，2023年。Bansal, H., Grover, A., 等。重新思考规模在上下文学习中的作用：基于可解释性的660亿规模案例研究。*arXiv*预印本 *arXiv:2212.09095*，2023年。

附录 A 算法细节

A.1 VPSLinear 完整前向传播

算法 4 VPSLinear.forward(x)

要求： 输入 $x \in \mathbb{R}^{N \times d_{in}}$，基础层 $W \in \mathbb{R}^{d_{out} \times d_{in}}$

$y_{base} \leftarrow xW^{\top}$ {基础计算}
$x_{2d} \leftarrow$展平(x)， $h_{2d} \leftarrow$展平(y_{base}.分离())
$pol \leftarrow$Policy.decide(h_{2d}) {自适应参数}
U， $V \leftarrow$构建器(x_{2d}， h_{2d}， W) {构建选择器}
$A \leftarrow W^{\top}V$\{(d_{in}， r)\}
$B \leftarrow WU$\{(d_{out}, r)\}
A， $B \leftarrow$ SpectralClip(A， B， τ) {稳定性控制}
$\delta \leftarrow$($x_{2d}A$)$B^{\top}$ {低秩扰动}
如果截断不为 None **则**
    $\delta \leftarrow$ 截断($\delta, -c, c$)
结束 if

返回 $y_{base} + pol.\gamma \cdot \delta$

A.2 自适应策略决策

算法 5 VPSPolicy.decide(h)

要求： 隐藏激活 $h \in \mathbb{R}^{N \times d}$，配置边界

$E \leftarrow \|h\|_F^2/(N \cdot d)$ {批次能量}
$\sigma \leftarrow 1 - e^{-E}$ {饱和非线性}
如果 词元熵 $\mathcal{H}$ 可用**则**
    $\sigma \leftarrow \max(\sigma, \min(1, \mathcal{H}/3))$
结束如果
根据改进历史调整 σ
$r \leftarrow r_{lo} + \lfloor(r_{hi} - r_{lo}) \cdot \sigma\rfloor$
$\gamma \leftarrow \gamma_{lo} +(\gamma_{hi} - \gamma_{lo}) \cdot \sigma$
$k \leftarrow k_{lo} + \lfloor(k_{hi} - k_{lo}) \cdot \sigma\rfloor$

返回 LayerPolicy(r, γ, k,...)

B 配置参数

表2：VPS配置参数

参数	默认值	描述
rank	2	低秩扰动的基础秩
topk	32	SK 选定的特征数量
伽马	0.5	构建器 扰动缩放系数
tau	0.8	谱范数裁剪阈值
构建器	混合	构建器类型：sk、sc 或混合
阶数	1	δ 展开的阶数
qk_耦合	True	在注意力中启用 Q/K 配对
L-BFGS_已启用	True	启用临时 L-BFGS
自适应_秩	True	启用基于能量的秩自适应-
自适应_伽马	True	tion 启用基于能量的伽马自适应-
阿尔法	10^{-3}	化 SC 构建器的岭正则化
秩_边界	$[1, 4]$	自适应秩范围
伽马_边界	$[0.3, 0.8]$	自适应伽马范围
topk_边界	$[16, 64]$	自适应 top-k 范围

C 代码可用性

VPS 的完整实现可在以下地址获取：

https: //github.com/Saba-Kublashvili/虚拟参数合成

该代码仓库包含：

- vps/vpscore/：VPS 核心实现
- vps/scripts/：评估与消融脚本
- vps/configs/：配置文件
- 用于开展可复现实验的 Google Colab 笔记本