**基于LoRA的Qwen2.5-3B模型中文成语完形填空任务微调**

摘 要

本项目将Qwen2.5-3B大语言模型应用于中文成语完形填空任务，采用LoRA (Low-Rank Adaptation)方法对参数进行高效微调，在ChID数据集上对模型进行了训练。系统设计采用了DeepSpeed分布式训练框架，结合8比特量化技术降低计算资源需求。

实验结果表明，经过约16小时的训练，模型在测试集上达到了99.83%的惊人高准确率。怀疑Qwen2.5-3B模型在预训练阶段已经接触过大量的中文成语数据，包括但不限于ChID数据集中的内容，使得模型在微调前就具备了强大的成语理解能力。为确保模型的泛化能力，本项目决定根据2021年的各地高考语文试题自行构建评测数据集，共构建22条。模型同样展现出优异表现，这一结果反映出Qwen2.5-3B模型在预训练阶段可能已具备较强的成语理解能力。

最后，本项目指出遮蔽掩码式的成语填空是封闭域任务，或许更多考察模型的知识提取而非真正的理解能力。为此，研究建议未来可通过开放式成语填空、成语猜谜等多维度评测方式，更全面地验证模型对成语的理解水平。

本项目的实践经验和技术方案，为大语言模型在特定中文自然语言处理任务上的应用提供了一定的参考，代码发布在<https://github.com/01xianyi/Qwen2.5-3B-ChID->

目录

[一、 引言 2](#_Toc187717315)

[1. 研究背景 2](#_Toc187717316)

[2. 研究意义 2](#_Toc187717317)

[3. 研究目标与创新点 3](#_Toc187717318)

[二、 相关工作 3](#_Toc187717319)

[1. 中文成语填空任务研究 3](#_Toc187717320)

[1.1 成语填空任务的定义 3](#_Toc187717321)

[1.2 数据集介绍 4](#_Toc187717322)

[1.3 成语填空任务方法综述 4](#_Toc187717323)

[2. 大语言模型微调方法 5](#_Toc187717324)

[三、 数据集与模型选择 5](#_Toc187717325)

[1. 数据集选择 5](#_Toc187717326)

[2. 数据预处理 6](#_Toc187717327)

[2.1 数据质量检查与分析 6](#_Toc187717328)

[2.2 数据格式转换 6](#_Toc187717329)

[3. 模型选择 7](#_Toc187717330)

[四、 系统设计与实现 8](#_Toc187717331)

[1. 实验环境 8](#_Toc187717332)

[2. 系统整体架构 8](#_Toc187717333)

[2.1 DeepSpeed 分布式训练方案 9](#_Toc187717334)

[2.2 LoRA 参数高效微调 10](#_Toc187717335)

[3. 训练流程实现 10](#_Toc187717336)

[3.1 训练器设计与实现 10](#_Toc187717337)

[3.2 损失函数设计与实现 11](#_Toc187717338)

[3.3 训练策略 12](#_Toc187717339)

[五、 实验过程及结果分析 12](#_Toc187717340)

[1. 训练过程分析 12](#_Toc187717341)

[1.1 训练损失演变 12](#_Toc187717342)

[1.2 梯度范数变化 13](#_Toc187717343)

[2. 性能分析 13](#_Toc187717344)

# 引言

## 研究背景

成语作为中国特有的语言文化现象，是中华文化的精髓和重要组成部分。这些由固定字数（通常为四字）组成的短语，不仅在日常交际中被广泛使用，更承载着丰富的历史文化内涵。然而，成语的理解和运用具有其独特的挑战性：首先，许多成语具有非组合性（non-compositionality）特征，其整体含义往往无法从字面意思直接推导；其次，大量成语源自历史典故，包含深层的隐喻义；再次，成语之间存在大量近义现象，在语境中的准确使用要求对其细微差别的把握。

近年来，国内外大语言模型发展迅速，以ChatGPT、Claude等为代表的模型展现出强大的语言理解能力。国内以Qwen、百川、书生等为代表的中文大语言模型的出现，为解决中文特定的语言理解任务提供了新的技术方案。Qwen2.5-3B作为开源的中文大语言模型，具有较好的性能表现和易于部署的特点，值得在成语理解任务中进行尝试与探索。

## 研究意义

本研究通过在成语填空任务上应用和验证Qwen2.5-3B模型的性能，探索了大语言模型在处理中文特定语言现象方面的能力。通过对模型在ChID数据集上的完整训练和测试，为研究中文大语言模型在成语理解任务上的表现提供了具体的实验数据和参考基准。这些实验结果和发现可以为后续相关研究提供有价值的参考。

从实践应用的角度来看，本研究的成果也有一定价值。首先，模型可以直接应用于教育领域，用于开发中文教育辅助工具，比如帮助学习者进行成语练习和测试，提升学习效果。其次，本研究提供了一个基于Qwen2.5-3B模型完成特定NLP任务的完整实现范例，为开发类似的中文文本理解应用提供了一定实践经验和技术参考。

此外，本研究也面临着一些重要的挑战需要解决。一方面，成语理解本身具有高度的复杂性，其往往涉及历史文化背景、比喻义等多层面的含义，这要求模型能具备深层的语义理解能力。另一方面，成语使用的上下文依赖性强，模型需要准确捕捉上下文信息才能做出正确判断。最后是计算资源的限制问题，大语言模型的训练和优化需要大量计算资源，如何在有限资源条件下实现模型的最优效果具有难度。

## 研究目标与创新点

本研究的主要目标是利用Qwen2.5-3B模型解决中文成语完形填空任务，并通过采用LoRA（Low-Rank Adaptation）方法进行高效微调，以降低计算资源需求。具体创新点包括：

1. **应用Qwen2.5-3B模型于中文成语填空任务**：探索大型开源中文语言模型在特定语言现象理解中的潜力，为相关研究提供新思路。
2. **采用LoRA进行高效微调**：通过Parameter-Efficient Fine-tuning (PEFT) 的一种具体实现方法，即LoRA，在不大幅增加模型参数的情况下，实现高效的模型微调，提高训练效率和效果。
3. **提出成语理解任务中的优化策略**：基于实验结果，提出针对成语完形填空任务的具体优化措施，提升模型在此类任务上的表现。

期望通过上述目标和创新点的实现，本项目的实践能够具有一定价值。相关代码已发布在<https://github.com/01xianyi/Qwen2.5-3B-ChID-/tree/master>

# 相关工作

本章节将回顾中文成语填空任务的研究进展、大语言模型的微调方法，特别是参数高效微调（PEFT）及其代表性方法LoRA（Low-Rank Adaptation）的相关研究，为本研究提供理论支持和技术参考。

## 中文成语填空任务研究

### 成语填空任务的定义

成语填空任务是自然语言处理（NLP）中的一种理解与生成任务，旨在在给定的上下文中填入最恰当的成语。该任务要求模型能够理解上下文的语境和语义，通常分为以下两种形式：

|  |  |  |  |
| --- | --- | --- | --- |
| 任务类型 | 任务形式 | 实现方法 | 常用模型结构 |
| 遮蔽掩码式 | 将文本中的成语替换为特殊标记`[MASK]`。 ·从提供的多个候选成语中选择正确答案（如ChID数据集中提供7个候选项）。 | 输入：被掩码的文本序列。 输出：候选项的概率分布，选择概率最高的成语作为填空结果。 | ·基于BERT类预训练模型的掩码语言模型（MLM）。 ·基于LSTM的双向编码器。 ·带有注意力机制的编码器-分类器架构。 |
| 生成式 | 给定上下文，直接生成合适的成语。  ·不提供候选选项，可能生成多个合理答案。 | 输入：包含提示或占位符的文本。 输出：直接生成成语作为填空结果。 | ·编码器-解码器（Encoder-Decoder）架构。 ·自回归语言模型，如GPT系列。 ·大型预训练语言模型。 |

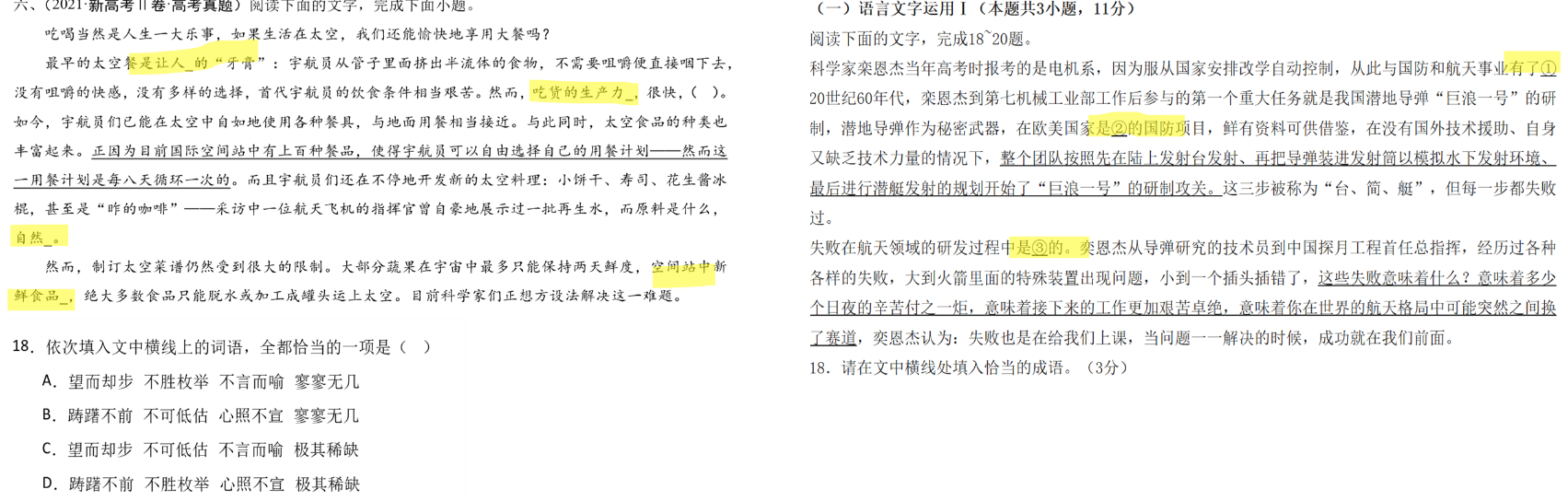


图 1遮蔽掩码式与生成式示例

### 数据集介绍

为了推动成语填空任务的研究，学术界和工业界开发了多个相关数据集。其中，ChID（Chinese Idiom Dataset）是目前较为广泛使用的数据集之一。ChID数据集包含大量包含成语的句子，并标注了正确的成语答案，适用于训练和评估模型在成语填空任务上的表现。此外，其他数据集如CCL（Chinese Corpus of Learning）和SIGHAN（Chinese Language Processing Benchmark）也提供了相关的语料资源，尽管其主要目标不完全是成语填空，但同样对模型的中文理解能力具有重要的参考价值。

### 成语填空任务方法综述

针对成语填空任务，研究者提出了多种方法，包括基于规则的方法、统计模型、深度学习方法以及大语言模型微调技术。

早期研究多采用基于规则的匹配方法，通过预定义的成语库和句法规则进行填空。该方法实现简单，但在处理复杂语境和新颖成语时表现欠佳，缺乏灵活性和适应性。

随后，研究者引入统计模型，如条件随机场（CRF）和隐马尔可夫模型（HMM），通过统计特征进行成语预测。这些模型在一定程度上提升了性能，但仍难以捕捉成语的深层语义关系，表现有限。

近年来，深度学习，尤其是预训练语言模型（如BERT、RoBERTa），在成语填空任务中取得了显著进展。这些模型通过大规模语料预训练，能够捕捉复杂的语言模式和语义信息。然而，在处理非组合性强、文化背景深厚的成语时，仍存在一定挑战。例如，清华大学CoAI实验室在2019年ACL发表的论文中，构建了大规模成语填空数据集ChID，并分析了成语表示方法和候选项设计对任务难度的影响。

随着大语言模型（如GPT-3、Llama、Qwen等）的兴起，研究者开始尝试将这些模型微调用于成语填空任务。通过在特定任务上进行微调，模型能够更好地适应成语的语义和上下文需求。虽然能够显著提升模型性能，但大规模模型的训练和微调往往需要大量的计算资源和存储空间，限制了其广泛应用。近期的研究综述了大语言模型微调技术的发展，强调了参数高效微调方法（如LoRA、QLoRA等）在降低计算和存储成本方面的优势。

成语填空任务的方法经历了从基于规则、统计模型到深度学习和大语言模型微调的演进。每种方法各有优缺点，研究者们不断探索，以期在处理复杂语境和具有深厚文化背景的成语时取得更优的表现。

## 大语言模型微调方法

大语言模型在自然语言处理（NLP）任务中表现优异，但由于其庞大的参数规模和高昂的计算成本，使得微调过程变得资源密集且复杂。因此，研究者提出了多种高效微调方法，以降低微调的计算与存储成本，同时尽量保持模型性能。

传统微调方法通常采用全参数更新的策略，即对预训练模型的所有参数进行梯度下降，以适配特定任务。这种方法尽管简单直接，但在处理如GPT-3等具有海量参数的大模型时，存在高计算成本、存储开销大、过拟合风险过高等缺陷。故有学者提出了参数高效微调（Parameter-Efficient Fine-Tuning， PEFT）方法，通过调整少量模型参数或在模型中插入少量可训练参数，而在显著降低计算和存储成本的同时，充分利用预训练模型的知识以适应新任务。其核心思想是通过最小化参数调整来优化资源利用，并缓解全量微调中的过拟合问题。

LoRA（Low-Rank Adaptation）是一种典型的参数高效微调方法。其主要原理是在预训练模型的权重矩阵中引入低秩矩阵，通过分解权重矩阵实现参数高效调整。LoRA先在模型的关键层（如注意力机制的投影层）中，分解权重矩阵为两个低秩矩阵的乘积。随后仅对这些低秩矩阵的参数进行训练，而其他预训练参数保持冻结。

故相比全量微调，LoRA仅需训练少量参数，从而显著降低计算资源需求和存储开销。并且适用于多种预训练模型架构，可广泛应用于不同的NLP任务。而且LoRA实现简单，可以无缝集成到现有的训练框架中，降低开发复杂性。在多数任务中，LoRA能够在保持甚至提升模型性能的同时，实现参数利用率的最大化。

受到以上相关工作的启发，本项目决定使用大模型+LoRA微调的方法针对某一中文成语数据集进行训练。

# 数据集与模型选择

## 数据集选择

由于其他数据集（如CCL和SIGHAN）虽然提供了相关中文语料资源，对模型的中文理解能力有重要参考价值，但它们的目标并非专注于成语填空任务。故本研究仅选择ChID数据集进行实验。

ChID是一个针对中文成语完形填空任务的大规模数据集，旨在评估模型对成语理解和应用的能力。该数据集包含来自新闻、小说和散文等领域的581，000个段落，并包含729，000个成语填空样本，包含大量不同类型和难度的成语，覆盖广泛的语境和应用场景。

数据集使用#idiom#替换段落中的成语，并提供了7个候选成语（包括1个正确答案、3个相似成语和3个随机成语），从而要求模型根据上下文选择最适合的成语。

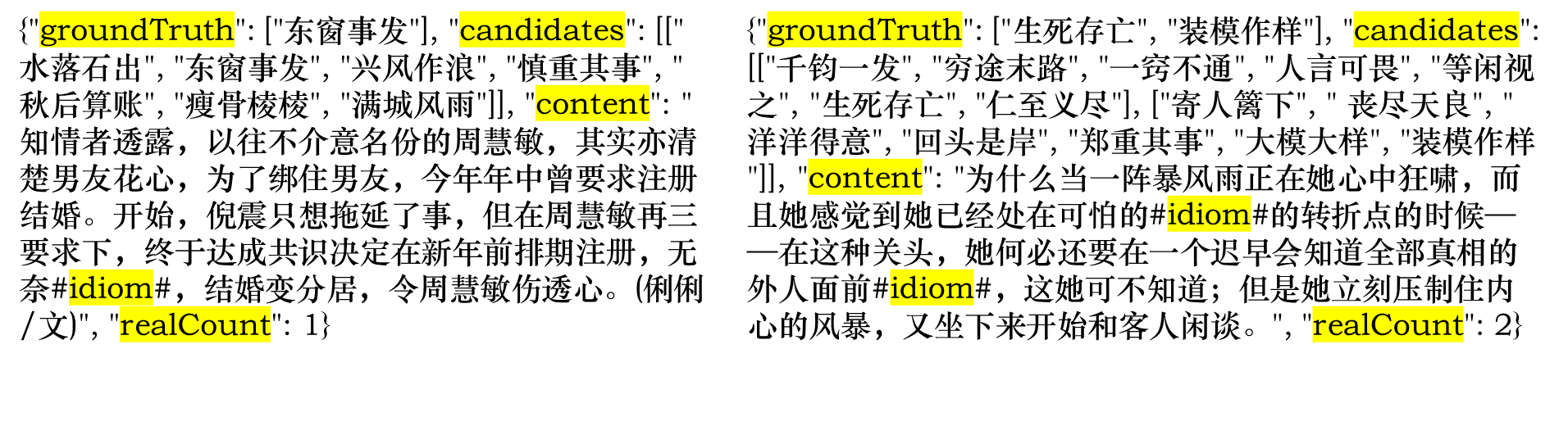


图 2ChID数据集展示

## 数据预处理

确保模型能够充分利用ChID数据集并适应其特性，我们对数据集进行了以下预处理操作。

### 数据质量检查与分析

在处理之前，我们对数据集的整体质量进行了初步检查，统计并验证了数据的结构和内容完整性，未发现数据质量问题。

表 1数据统计信息

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 基础统计 | |  | 成语统计 | |  |
| 总样本数 | 648920 |  | 不同成语总数 | 3848 |  |
| 平均文本长度 | 355.18 |  | 如日中天 | 4823 |  |
| 最短文本长度 | 186 |  | 失魂落魄 | 4581 |  |
| 最长文本长度 | 770 |  | 独树一帜 | 4182 |  |
| 平均填空数量 | 3 |  | 层出不穷 | 4025 |  |

### 数据格式转换

为了将ChID数据集转换为适合大模型训练的格式，即将任务转换为明确的指令形式，包含具体的任务描述和期望输出，我们有以下处理。

首先，对于每个数据样本，我们根据成语填空的数量生成多个训练样本。具体而言，对于每一个成语空缺位置，我们将其替换为<mask>，而将其他空缺位置用占位符[待填]替换。这一操作确保模型在处理当前填空位置时，能够专注于该位置的上下文信息，而不受其他空缺位置的干扰。随后，我们构建输入文本，包含文段内容、候选成语列表，并要求模型从中选择最合适的成语填入<mask>处。对应的输出则为正确的成语答案。

随后将处理后的数据组织成适合大模型训练的格式。我们将每个处理后的样本构建为一个包含用户输入和助手回复的对话形式，确保数据格式的一致性和可读性。通过批量处理训练集、验证集和测试集等多个数据文件，确保所有数据集采用统一的格式，从而便于大规模模型的高效训练。

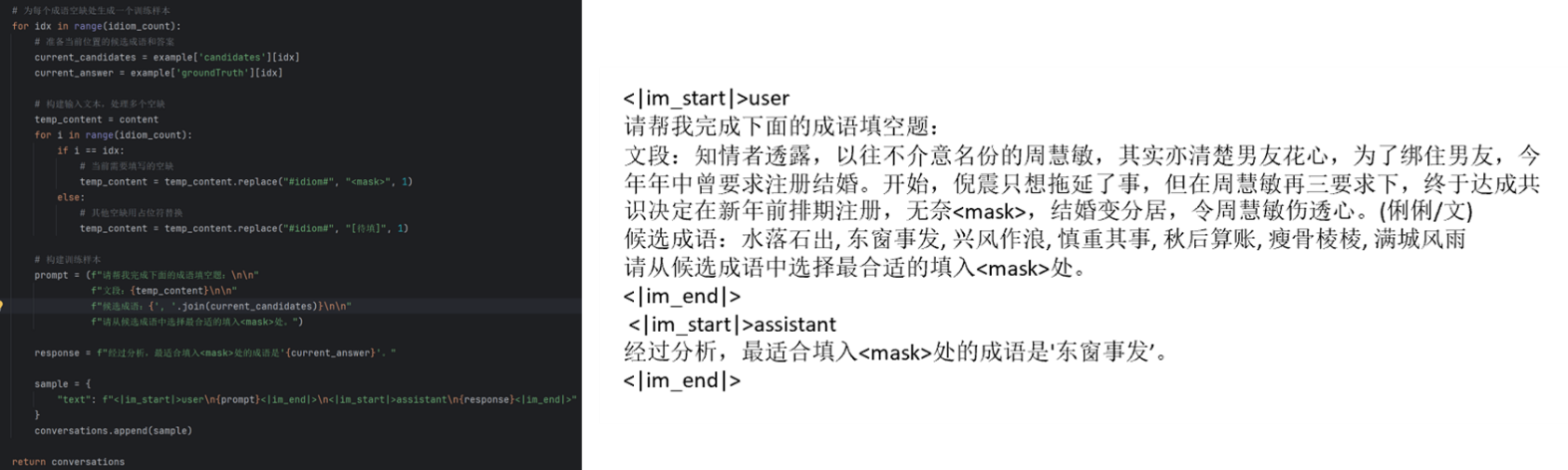


图 3代码及样本示例

## 模型选择

本研究选择Qwen2.5-3B作为基础模型，主要基于其在Qwen2.5-3B CLiB中文大模型能力评测榜单中的优异表现。在5B以下开源大模型的排行榜中，Qwen2.5-3B位居第一，其在多个关键任务中的评分如下：



图 4榜单（<https://github.com/jeinlee1991/chinese-llm-benchmark>）

# 系统设计与实现

本章详细介绍了基于Qwen2.5-3B模型的中文成语补全系统的设计与实现过程。系统采用了DeepSpeed分布式训练框架和LoRA 参数高效微调方法，实现了高效的模型训练与优化。

## 实验环境

本次实验在厚德云平台的云服务器上完成，租用了高性能的显卡资源以支持任务训练。

表 2硬件环境

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 类别 | 参数 |  | 类别 | 参数 |
| 处理器配置 | -CPU型号：Intel(R) Xeon(R) Platinum 8352V  -基准频率：2.10GHz  -物理核心数：48核  -缓存配置：L3缓存16MB，L2缓存192MB |  | 系统内存 | -总容量：235GB -可用内存：217GB -Buffer/Cache：13GB |
| GPU配置 | -型号：NVIDIA GeForce RTX 4090 × 4 -显存：24GB GDDR6X/卡（总计96GB） -最大功耗：450W/卡 |  | 存储配置 | -系统盘：57GB SSD（已使用45%） -数据盘：393GB（已使用7.2GB） -共享存储：1PB（public目录） |

表 3软件环境

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 类别 | 版本/环境 |  | 类别 | 版本/环境 |
| 基础环境 | -操作系统：Ubuntu 22.04.5 LTS (Jammy Jellyfish)  -CUDA版本：12.5.82  -Python版本：3.10.13  -NVIDIA驱动版本：555.42.06 |  | 核心框架 | -PyTorch：2.5.1+cu121  -Transformers：4.48.0  -DeepSpeed：0.16.2  -PEFT：0.14.0 |

## 系统整体架构

系统采用模块化设计思想，主要包含配置管理、数据处理、模型训练和优化四个核心模块。系统的整体架构如下图所示。

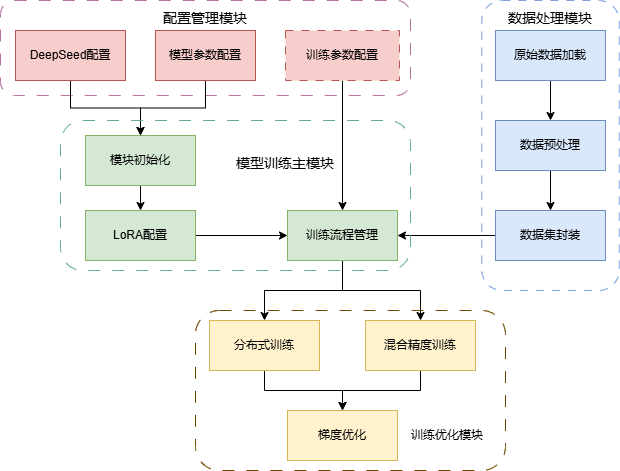


图 5系统架构图

训练过程涉及到的代码文件有：config\_danka.py，danka.py，trainer.py，data\_process.py。

其中配置管理模块(config\_danka.py)负责处理DeepSpeed相关配置，包括分布式训练策略、优化器设置和混合精度训练等参数的管理。该模块通过数据类(dataclass)实现了配置的规范化定义和序列化。

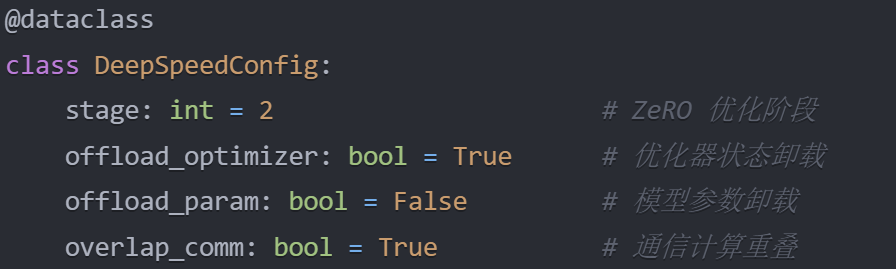
模型训练主模块(danka.py)作为系统的核心，负责协调各个组件的工作。它包含了模型参数配置、数据参数配置和训练参数配置等重要组件，实现了模型与分词器的创建、LoRA配置的应用以及训练流程的启动等功能。

数据处理模块(trainer.py中的ChIDDataset)负责训练数据的加载、预处理和批处理。该模块实现了自定义的数据集类，支持对成语补全任务的特殊数据格式进行处理。

训练流程模块(trainer.py)实现了自定义的训练器，负责模型的训练过程管理，包括损失计算、优化器更新、模型保存等功能。该模块通过继承Hugging Face的Trainer类，实现了对DeepSpeed的集成工作。

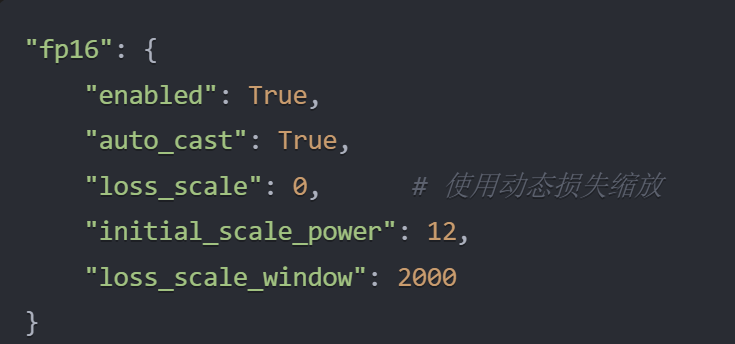
### DeepSpeed 分布式训练方案

本系统采用 DeepSpeed 框架实现分布式训练，主要配置包括:



系统采用了ZeRO Stage-2优化策略，该策略将优化器状态和梯度分片到不同设备上，同时保持模型参数的完整副本，在内存效率和训练性能之间取得了良好的平衡。为了进一步优化显存使用，系统启用了优化器状态向CPU内存的卸载(Optimizer Offload)机制。

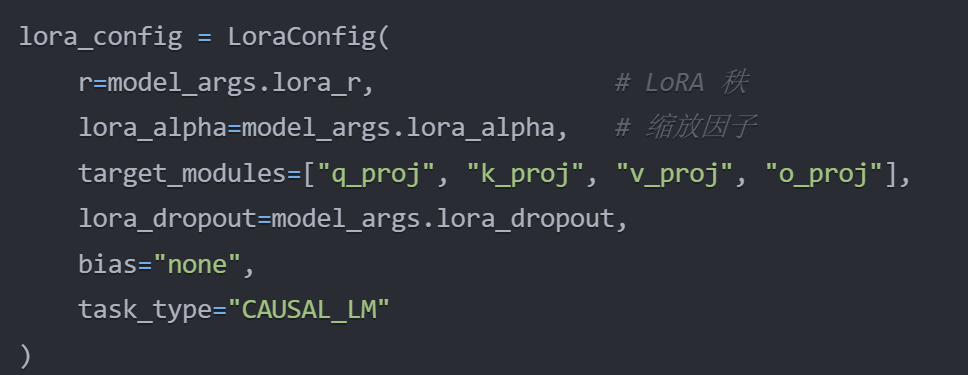
在混合精度训练方面，系统配置如下:



系统采用FP16混合精度训练，通过动态损失缩放机制自适应调整梯度尺度，有效防止了梯度消失和溢出问题。

### LoRA 参数高效微调

系统采用LoRA技术实现参数高效微调，主要配置包括:



LoRA配置针对模型的注意力层进行了优化，包括查询(query)、键(key)、值(value)和输出(output)投影矩阵。通过设置适当的秩(r=8)和缩放因子(alpha=32)，在保持模型性能的同时显著降低了可训练参数量。

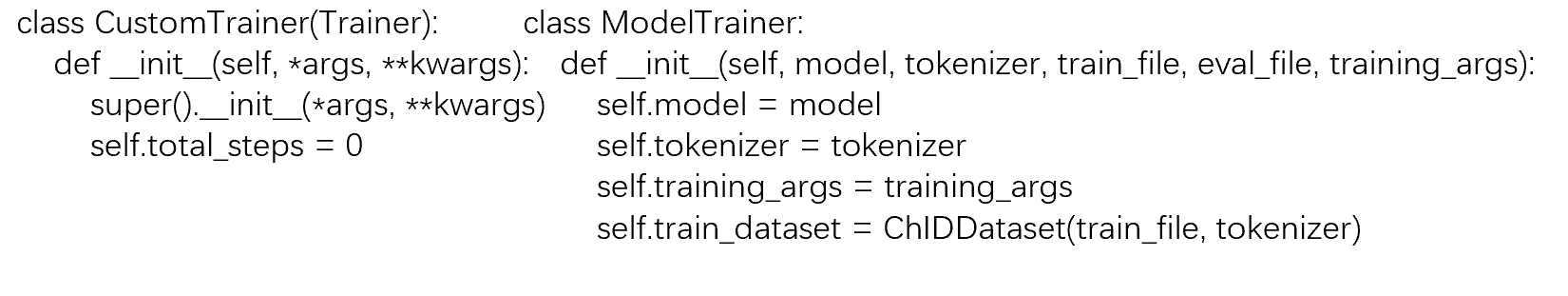
## 训练流程实现

### 训练器设计与实现

本项目基于Hugging Face的Trainer框架设计了自定义训练器，通过继承和扩展实现了对分布式训练的深度支持和精确的训练过程控制。

训练器采用了模块化的架构设计，主要包含训练器类（CustomTrainer）和训练管理类（ModelTrainer）两个核心组件。

CustomTrainer通过继承Transformer的Trainer类，在保持原有功能的基础上扩展了分布式训练支持、训练状态追踪等功能。训练管理类则负责训练流程的统筹和资源管理。



### 损失函数设计与实现

#### 交叉熵损失

针对遮蔽掩码式（Masked Prediction）的成语填空任务，本项目选择交叉熵损失（Cross-Entropy Loss）作为损失函数。交叉熵损失函数在深度学习和自然语言处理领域中被广泛应用，尤其在分类任务中表现出色。它通过衡量预测概率分布与真实标签分布之间的差异，指导模型参数的更新，以提高模型的预测准确性。损失函数原理

交叉熵损失函数的定义源于信息论，旨在量化两个概率分布之间的信息差异。具体而言，对于一个具有个类别的分类问题，交叉熵损失函数定义为：

其中，是真实分布中第类的概率，通常在分类任务中，真实标签为one-hot编码，即某一类的概率为1，其余类为0。则是模型预测的第类的概率。

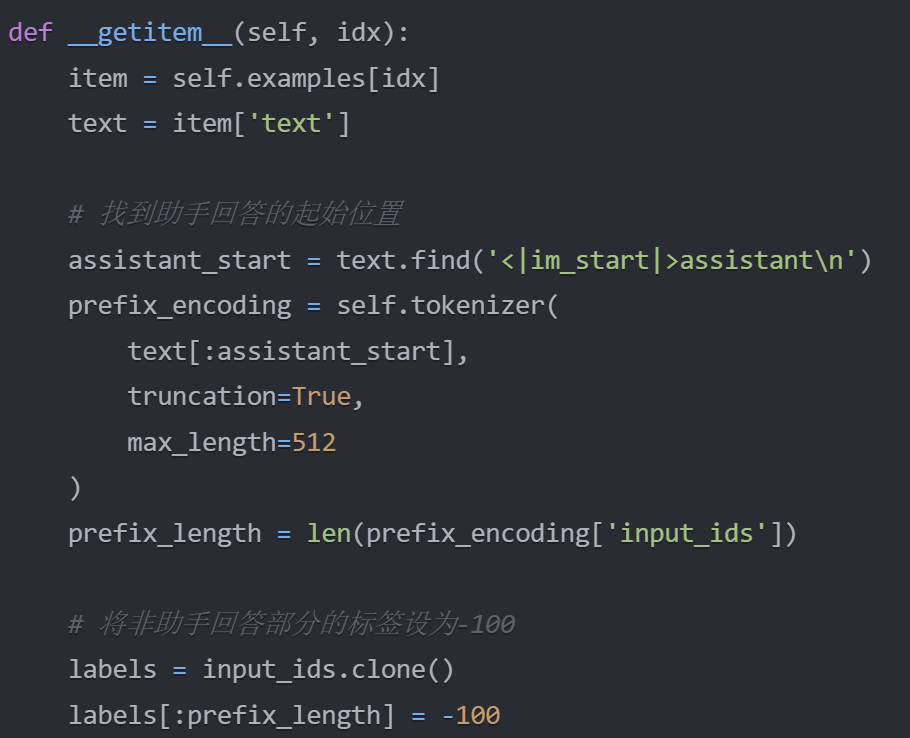
在语言模型中，尤其是成语完型填空任务中，模型需要预测缺失的成语或词汇。交叉熵损失函数在此场景下能够有效衡量模型预测的词概率分布与真实标签分布之间的差距，从而指导模型的优化过程。通过最小化交叉熵损失，模型能够逐步调整参数，使预测分布更接近真实分布，提高填空的准确性。

#### 具体实现

本次任务中，模型需要处理用户输入与助手回答的交互。为了确保损失函数的计算仅针对助手的回答部分，特对用户输入部分的标签进行了特殊处理。具体来说，用户输入部分的标签被设置为-100，这一设置在损失计算过程中会被忽略，确保模型的优化仅基于助手生成的回答。此外，对于输入序列中的填充（padding）部分，标签同样被设置为-100，以避免这些无关部分对损失计算的干扰。

这种标签处理方法的核心思想在于精确控制损失函数的计算范围，使其专注于模型需要优化的部分。这不仅提高了训练效率，还能有效防止由于无关信息引入的噪声，提升模型的整体性能。

图 6计算损失前的处理



损失计算的具体实现被封装在了在CustomTrainer中。在具体实现中，模型的输出为每个 [MASK] 标记位置上各候选成语的概率分布。交叉熵损失函数通过计算预测概率与真实标签之间的负对数似然，指导模型优化其参数以提高正确成语的预测概率。

### 训练策略

为了降低模型的计算和存储开销，本项目采用8比特量化（8-bit Quantization）技术对Qwen2.5-3B模型进行量化训练。8比特量化通过将模型参数从32位浮点数（FP32）压缩为8位整数（INT8），显著减少了模型的内存占用和计算需求，同时在大多数情况下，量化对模型性能的影响较小。

此外固定学习率为0.00005，以保证长时间训练过程中的稳定性，并且启用梯度累计，每2个小批次累积一次梯度。其余细节内容可参见danka.py、trainer.py代码文件。

# 实验过程及结果分析

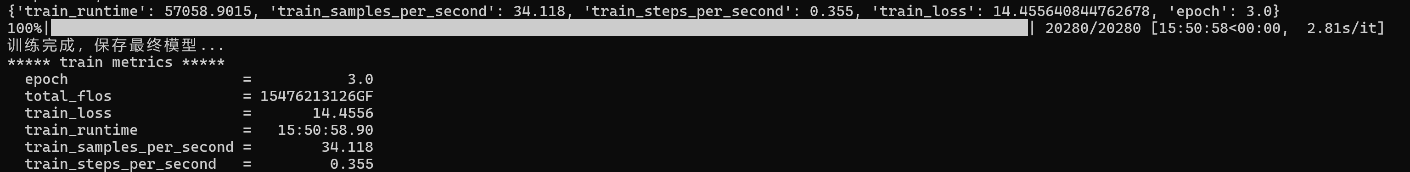
在历经15:50:58后，模型训练完毕。

图 7训练完成

## 训练过程分析

### 训练损失演变

训练过程记录了405个时间点的损失值，训练损失整体保持在14.44左右波动，显示出良好的稳定性。在训练前期（大约前100步），损失值略微偏高且波动较大。中期阶段（约100-300步）趋于稳定，波动幅度明显减小。后期阶段（300步以后）维持在一个相对稳定的水平，表明模型达到了较好的收敛状态。

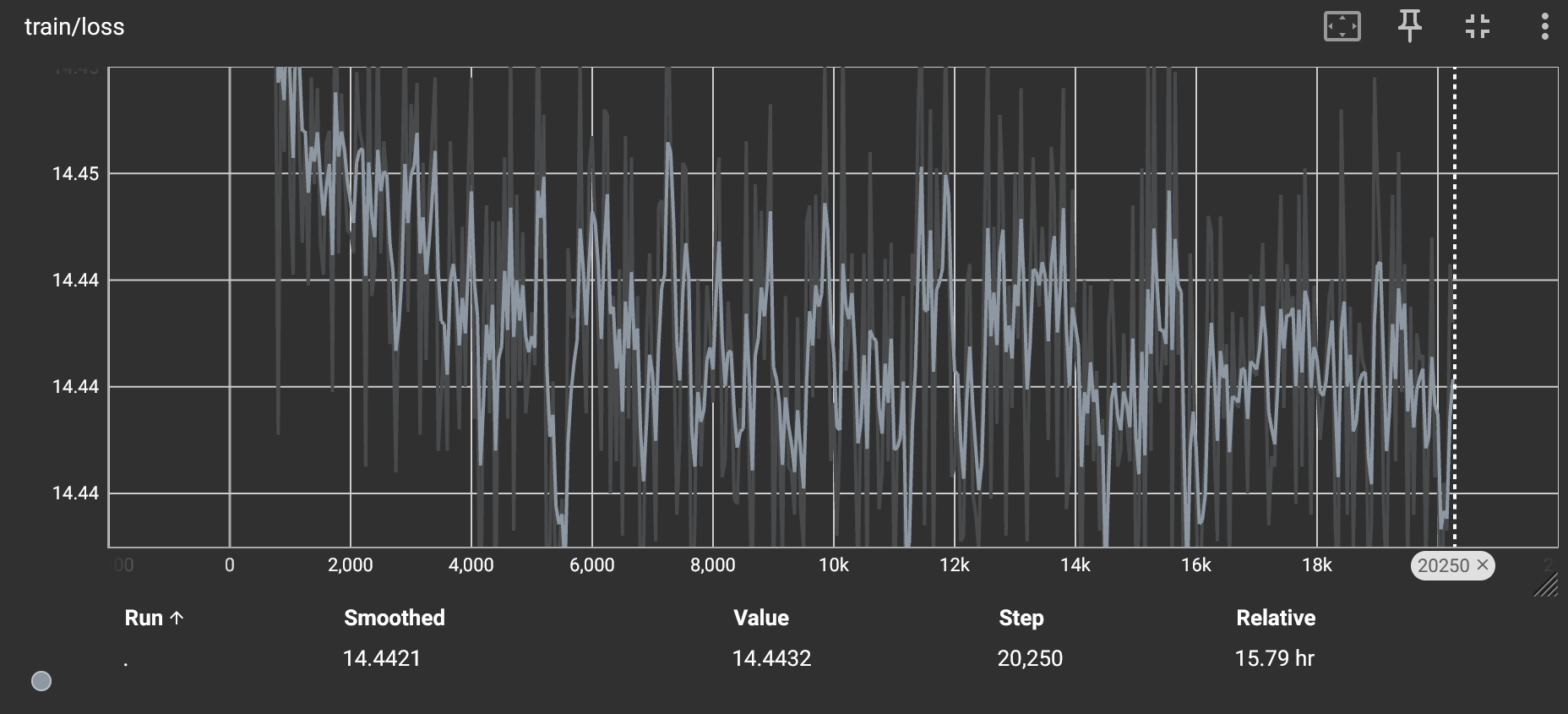


图 8损失变化

### 梯度范数变化

梯度范数在前几个时间点上从训练初期的1.6796迅速下降到了0.05附近并最终稳定在0.03左右。初始阶段：梯度较大，表明模型参数更新幅度较大。随后梯度逐渐降低，显示模型逐步接近最优解。后期时梯度范数的小幅波动说明模型在最优解附近进行微调

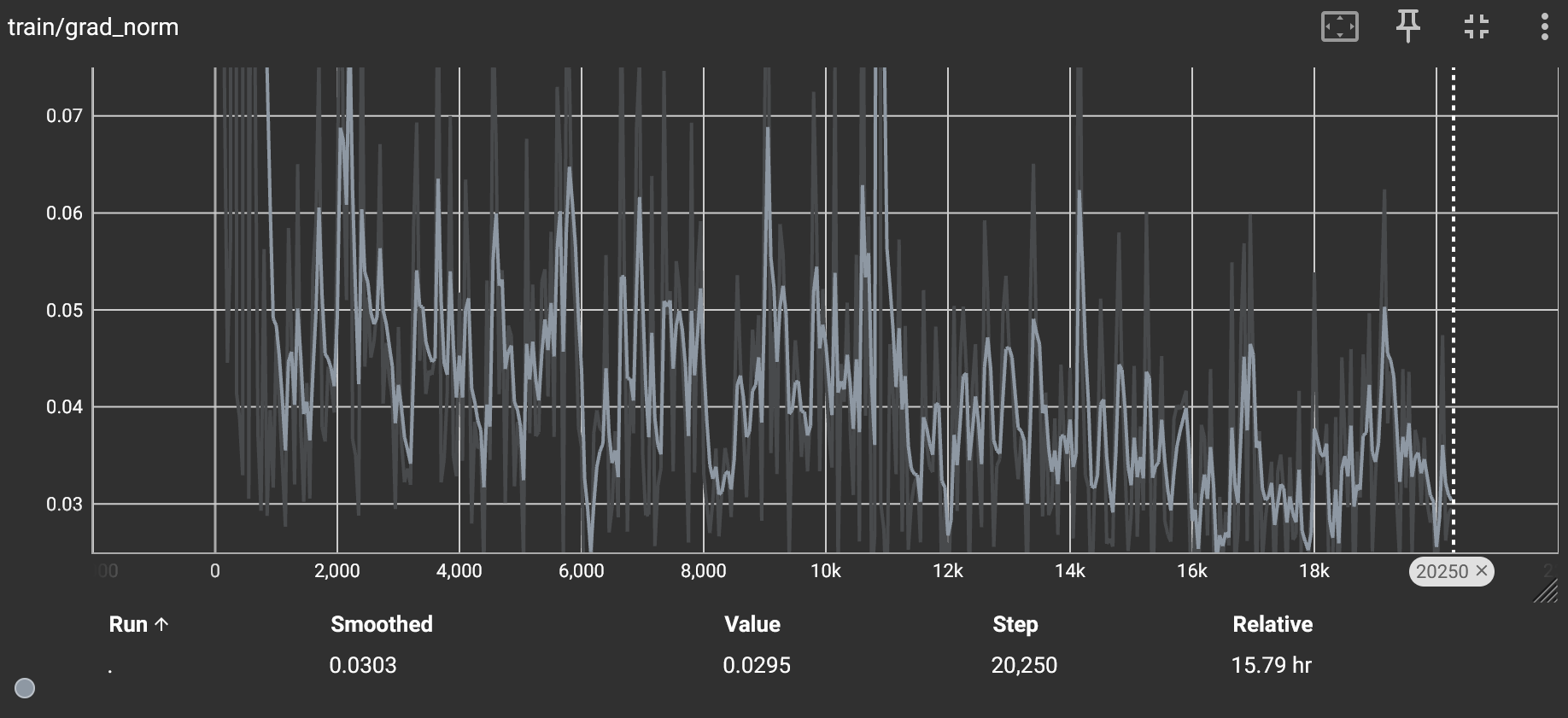


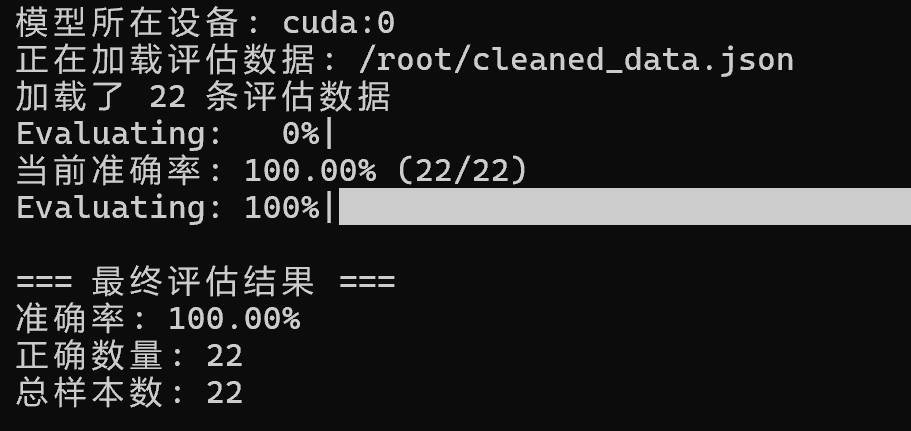
图 9梯度范数变化

以上的损失值与梯度范数的表现，表明模型成功找到了或接近于局部最优解，并且在训练过程中未出现显著的梯度爆炸或消失现象，可以认定模型成功训练。

## 性能分析

经历长时间训练后，模型的准确率达到了惊人的99.83%，成功预测了24948条数据中的24905条。怀疑Qwen2.5-3B模型在预训练阶段可能已经接触过大量的中文成语数据，包括但不限于ChID数据集中的内容。这使得模型在微调前就具备了强大的成语理解能力。其在预训练过程中可能已经学习到了成语的上下文关系和语义特征，为后续的特定任务适应打下了良好基础。

为确保模型的泛化能力，本项目决定自行构建评测数据集，由于时间关系仅构建二十二条数据，数据来源于2021年的各地高考语文试题及模拟题。发现模型准确率依然达到惊人的100%。



针对这一现象，本项目提出以下思考：成语填空任务本质上是一个封闭域的语言理解任务，相比开放域的生成任务，其答案空间相对固定。故对于预训练充分的大模型而言，这类任务更多地考验模型对已有知识的提取能力，而非泛化能力。

若想全面检查大模型的能力或许可以设计无候选答案的开放式成语填空任务，直接考察模型的成语生成能力，从而更全面地检验模型对成语的理解和运用能力。另一方面，也可以设计经典的成语猜谜游戏，通过结合视觉线索和文字提示，综合考察大模型对成语的深刻理解。这种评测方式不仅能验证模型的基础知识储备，还能考察其在实际应用场景中的表现。此外，也可以要求模型对所生成的成语进行来源解释和含义阐述，进一步验证其对成语文化内涵的把握程度。通过这种多维度的评测体系，我们能够更准确地认识和改进模型在成语理解任务上的能力。