经验 + 3：参考样本对多模态情感分类的作用

宋泰霖 202518013229043

1. 背景

1.1 引言

”经验+3“最初起源于中文互联网社区的评论区文化：在B站、贴吧等平台，用户发布一条评论通常可获得3点经验值，用于提升账号等级。账号的等级可以通过更多的评论来提升，那么大模型的理解能力能否通过更多的参考样本来提升呢?

本实验将**给大模型提供 3 个参考样本（3-shot learning）**形象地称为 "给模型加 3 点经验"。与人类"水经验"不同，大模型的"经验 + 3"并非无意义的重复，而是通过少量标注样本快速校准任务理解能力，这正是大语言模型（LLMs）最具革命性的能力之一 ——**上下文学习（In-Context Learning, ICL）**。

1.2 大模型上下文学习的研究背景

2020年GPT-3的问世首次证明了大模型具备强大的上下文学习能力：无需更新模型参数，仅通过在输入中提供几个示例，模型就能完成从未见过的下游任务。这一发现彻底改变了自然语言处理的范式，使得"零样本 / 少样本学习"成为大模型应用的主流方式。

上下文学习的效果高度依赖于提示词格式和参考样本质量。特别是在多模态场景下，模型需要同时理解文本和视觉信息，输入结构更加复杂，参考样本的作用变得尤为关键。

1.3 多模态情感分类的挑战与研究意义

情感计算旨在让机器理解、识别和建模人的情感状态，是实现人机自然交互的核心技术之一。在对话场景中，情感不仅体现在文本内容上，还通过面部表情、肢体动作、语音语调等多模态信号传递。多模态情感识别因此成为近年来的研究热点，在智能客服、心理健康监测、影视分析等领域具有广泛应用前景。

然而，多模态情感分类面临着三大核心挑战：

1. **模态异质性**：文本是离散的符号序列，而视觉是连续的像素空间，两种模态的表示方式差异巨大；
2. **上下文依赖性**：对话中的情感具有连续性，当前话语的情感往往依赖于前文语境；
3. **标注成本高**：多模态数据的情感标注需要同时考虑文本和视觉信息，标注难度大、成本高。

大模型的出现为解决这些挑战提供了新的思路。一方面，多模态大模型（如 Qwen3.5、LLaVA 等）通过统一的架构实现了文本和视觉的联合表示学习；另一方面，上下文学习能力使得模型可以利用少量标注样本快速适应情感分类任务，大幅降低了标注成本。

1.4 本实验的研究问题与设计

基于上述背景，本实验选取**MELD（Multimodal EmotionLines Dataset）**作为实验数据集，以**Qwen3.5-2B**为基础模型，围绕 "经验+3" 这一核心主题，系统研究参考样本在多模态情感分类中的作用。具体回答以下两个关键问题：

1. 在不进行参数更新时，3 个参考样本（经验 + 3）是否能提升文本与多模态情感分类性能？这种提升在多模态场景下是否更为显著?
2. 在进行LoRA参数高效微调后，"经验 + 3"的作用是否仍然存在？如果训练数据只包含零样本提示词格式，是否会削弱模型对少样本输入格式的适应性?

为回答上述问题，本实验设计了三类对比设置：

- **基线前向实验**：不训练，仅用原始模型做零样本 / 少样本前向分类；
- **LoRA（零样本提示词训练）**：训练数据只使用零样本形式；
- **Mixed LoRA（50% 零样本 + 50% 少样本提示词训练）**：用混合提示词格式微调，检验训练格式与测试格式一致性的重要性。

本文在所有实验分析中，重点关注**少样本相比零样本的增益变化**，同时包含对实验过程的时间成本分析，为多模态大模型在情感计算领域的应用提供可复现的实验依据和资源参考。

---

2. 技术原理

2.1 多模态大模型的工作原理

本实验使用的**Qwen3.5-2B**是一款支持文本和视觉输入的多模态大模型，其核心架构由**语言模型**、**视觉编码器**和**多模态对齐模块**三部分组成。

2.1.1 文本理解原理

Qwen3.5-2B 的语言模型基于 Transformer 架构，采用了**Linear Attention**与传统 Self-Attention 混合的设计，共包含 24 层 Transformer 块，总参数量为 22.3 亿。

1. **文本 Token 化**：输入文本首先通过 Qwen 专属的 tokenizer 转换为整数序列。Qwen 的 tokenizer 采用字节级 BPE（Byte Pair Encoding）算法，词汇表大小为 151936，支持中英文等多种语言。
2. **嵌入层**：将 token 序列转换为维度为 2048 的词嵌入向量，同时添加位置嵌入以保留序列的位置信息。
3. **Transformer 编码器**：
    1. **Self-Attention 层**：计算序列中每个 token 与其他所有 token 的注意力权重，捕捉文本的上下文依赖关系。Qwen3.5 在部分层使用了 Linear Attention 机制，将注意力计算的时间复杂度从 $O(n^2)$ 降低到 $O(n)$，显著提升了长文本处理效率。
    2. **MLP 层**：由上采样投影层（up_proj）、门控投影层（gate_proj）和下采样投影层（down_proj）组成，对每个 token 的表示进行非线性变换，增强模型的表达能力。
4. **输出层**：将最后一层 Transformer 的输出通过线性层映射到词汇表空间，预测下一个 token 的概率分布。

2.1.2 视觉理解原理

Qwen3.5 的视觉编码器基于CLIP-ViT架构，负责将图像转换为与文本嵌入空间对齐的视觉token序列。

1. **图像预处理**：输入图像首先被调整为固定尺寸，然后进行归一化处理。
2. **图像分块与嵌入**：将图像划分为不重叠的 patch，每个 patch 被展平为一维向量，通过线性投影转换为视觉嵌入。同时添加位置嵌入以保留 patch 的空间位置信息。
3. **视觉 Transformer 编码器**：通过多层 Self-Attention 和 MLP 层，提取图像的全局和局部特征，最终输出一组视觉 token 序列。
4. **多模态对齐**：视觉 token 序列通过一个线性投影层，映射到与文本嵌入相同的维度空间，实现视觉与文本表示的对齐。

2.1.3 多模态输入的融合方式

在多模态任务中，Qwen3.5 采用了**interleaved 输入格式**：将视觉 token 序列插入到文本 token 序列的对应位置，形成一个统一的多模态 token 序列，然后输入到语言模型中进行处理。

本实验中，我们采用**联系表（Contact Sheet）**方式压缩视频信息：从每个视频片段中均匀抽取 4 帧，将这 4 帧拼接成一张 2×2 的联系表，然后将这张联系表作为单个图像输入到视觉编码器中。这种方式既保留了视频的时序信息，又避免了输入过多视觉 token 导致的显存和计算开销激增。

2.2 LoRA 参数高效微调原理

全参数微调需要更新大模型的所有参数，计算成本高、存储开销大，且容易导致灾难性遗忘。**LoRA（Low-Rank Adaptation，低秩适配）**作为一种参数高效微调方法，通过在 Transformer 的注意力层插入低秩矩阵，仅更新少量参数即可实现与全参数微调相当的效果。

2.2.1 LoRA 的基本思想

LoRA 的核心假设是：大模型在下游任务上的微调过程中，权重的更新具有低秩特性。也就是说，权重的变化可以分解为两个低秩矩阵的乘积。

对于 Transformer 中的线性层，其前向传播可以表示为：

$$h = Wx$$

其中$W \in \mathbb{R}^{d \times k}$是预训练好的权重矩阵。

在 LoRA 中，我们冻结预训练权重W，同时引入两个可训练的低秩矩阵$A \in \mathbb{R}^{r \times k}$和$B \in \mathbb{R}^{d \times r}$，其中$r \ll \min(d, k)$称为 LoRA 秩。前向传播变为：

$$h = Wx + BAx$$

初始化时，A使用高斯分布初始化，B初始化为零矩阵，因此训练开始时 LoRA 分支的输出为零，不会影响预训练模型的性能。

2.2.2 LoRA 的关键参数

本实验中使用的 LoRA 参数如下：

- **lora_r=16**：LoRA 秩，控制低秩矩阵的大小。秩越大，可训练参数越多，表达能力越强，但计算和存储开销也越大。
- **lora_alpha=32**：LoRA 缩放系数，用于缩放 LoRA 分支的输出。通常设置为 lora_r 的 2 倍，使得在改变秩时不需要调整学习率。
- **lora_dropout=0.05**：在 LoRA 分支中添加 dropout，防止过拟合。
- **lora_bias="none"**：不训练偏置项，进一步减少可训练参数。

2.2.3 本实验中的 LoRA 目标模块

Qwen3.5-2B 包含多种线性层，本实验选择微调以下关键模块：

文本模态：`model.language_model`下的**所有线性层**（除`lm_head`），包括：

- 所有 Transformer 层的注意力层：`layers.*.self_attn.q_proj`、`k_proj`、`v_proj`、`o_proj`
- 所有 Transformer 层的 MLP 层：`layers.*.mlp.up_proj`、`down_proj`、`gate_proj`
- 其他可能存在的线性层（如嵌入层投影等）

视觉模态：在文本模态基础上，额外增加`model.visual`下的**所有线性层**（视觉编码器的注意力和 MLP 层）

2.2.4 Mixed LoRA 的原理

传统的 LoRA 微调通常只使用一种提示词格式（如零样本）进行训练，这可能导致模型对训练格式产生过拟合，丧失对其他输入格式的适应性。

Mixed LoRA 的核心思想是：在训练数据中混合使用多种提示词格式（本实验中为 50% 零样本 + 50% 少样本），让模型在训练阶段同时学习不同格式的输入处理方式。这样既保留了模型在零样本下的性能，又不会丧失少样本上下文学习能力。

在验证阶段，Mixed LoRA 也使用混合格式的验证集计算评估损失，选择最优检查点，确保模型在两种格式下都能表现良好。

---

3. 实验环境

3.1 硬件环境

类别	配置详情	用途
CPU	x86_64 架构，Linux 5.15.0-173-generic 系统	通用计算与任务调度
GPU	NVIDIA GeForce RTX 3090 NVIDIA A800 80GB PCIe	文本 LoRA 训练（RTX 3090） 视觉 LoRA 训练（A800）

3.2 软件环境

软件包	版本	功能说明
Python	3.12.12	Anaconda 发行版，实验基础运行环境
PyTorch	2.9.0+cu128	深度学习框架，支持 CUDA 加速
CUDA	12.1	NVIDIA 显卡并行计算框架
cuDNN	9.0.1	深度神经网络加速库
transformers	5.6.0.dev0（评估）/ 4.57.1（训练）	模型加载、Prompt 构造与推理，本地 vendor 定制化支持 Qwen3.5
peft	0.17.1	LoRA 低秩适配实现
Pillow	10.3.0	视觉模态图像处理（帧拼接、尺寸调整）
ffmpeg/ffprobe	-	视频帧提取与元数据解析

3.3 模型基础配置

- 模型名称：Qwen 3.5-2B（量化版本）
- 参数量：2,230,060,864（2.23B）
- 架构特征：24 层语言模型、Linear Attention 机制、MLM 与 Self-Attention 混合设计
- 计算精度：`torch\.bfloat16`（混合精度训练 / 推理）

---

4. 方法

4.1 任务定义

任务为 **七分类对话情感识别**，标签集合为：

- anger（愤怒）、disgust（厌恶）、fear（恐惧）、joy（快乐）、neutral（中立）、sadness（悲伤）、surprise（惊讶）

输入样本来自 MELD 中的单条 utterance，模型需预测当前说话人当前 utterance 的情感标签。

4.2 数据集

本实验使用 MELD 数据集，核心特征：

- 英文多模态对话情感识别数据集，源自情景喜剧《Friends》；
- 包含文本转写、视频切片及情感标注；
- 数据划分：官方 `train / dev / test`；
- 下载地址：https://affective-meld.github.io/。

4.3 Prompt 设计

4.3.1 文本 zero-shot

包含 system prompt（任务定义、标签集合、输出约束）和 user prompt（最近 3 句文本历史、当前说话人、当前 utterance、情感标签查询问题），模型输出约束为 7 个标签之一。

4.3.2 文本 few-shot（经验 + 3）

在 zero-shot 基础上增加 3 个参考样本（训练集固定随机种子选取，测试集共享），每个参考样本包含文本历史、当前说话人 /utterance、gold label，保证实验可复现。

4.3.3 多模态 zero-shot

文本 zero-shot 基础上增加视觉参考：从当前 utterance 视频 clip 均匀抽取 4 帧，拼接为 2×2 contact sheet（`cell_size=304`，`image_max_pixels=401408`），输入包含文本历史、contact sheet、说话人utterance文本。

4.3.4 多模态 few-shot（经验 + 3）

多模态 zero-shot 基础上，额外加入 3 个带视觉参考的参考样本（当前样本 + 3 个 demo 均含视觉信息，历史上下文仅保留文本）。

4.4 基线方法

由于预训练模型Qwen 3.5-2B本身具备先验知识，极限方法不对模型做任何修改，直接在测试集上将prompt输入模型并评估输出结果。

4.5 LoRA 微调设计

为验证微调对 few-shot 作用的影响，设计三类 LoRA 微调方案：

4.5.1 通用 LoRA 参数

参数	值	说明
lora_r	16	LoRA 秩
lora_alpha	32	LoRA 缩放系数
lora_dropout	0.05	Dropout 概率
lora_bias	"none"	无偏置
目标参数占比	0.76%	仅微调 186 个关键线性层

4.5.2 Text LoRA

仅微调语言塔，训练数据为 zero-shot prompt 格式，监督信号为 gold label。训练超参数：

- `batch_size=2`，`grad_accum=8`，学习率 `2e-4`，`num_train_epochs=10`，梯度检查点开启。

4.5.3 Visual LoRA

同时微调语言模型线性层与视觉编码器线性层，训练数据为多模态 zero-shot 格式。训练超参数：

- `batch_size=1`，`grad_accum=8`，其余同 Text LoRA。

4.5.4 Mixed LoRA

训练集包含 50% zero-shot + 50% few-shot prompt，验证集按混合格式计算 `eval_loss` 选优，测试集仍分别评测 zero-shot/few-shot 性能，保证与传统 LoRA 对比公平性。

---

5. 实验设置

5.1 核心配置

配置项	值
文本历史窗口	3（最多保留前 3 句对话历史）
few-shot 参考样本数	3
视觉帧采样	4 帧 / 样本，2×2 contact sheet，cell_size=304
评估指标	Accuracy、Macro-F1、时间成本

配置项	值
解码策略	贪心解码（num_beams=1，无采样）
随机种子	42（保证结果可复现）
最大输入长度	4096（训练）/ 2048（评估）

5.2 时间成本测量方法

- 训练时间：记录每个 epoch 从数据加载到参数更新完成的总耗时（单位：秒 / 小时）；
- 评估时间：记录测试集全量样本推理 + 指标计算的总耗时，区分 zero-shot/few-shot；
- 数据来源：实验日志中的 `elapsed_seconds` 字段，统一统计 test 集结果。

---

6. 实验结果

6.1 性能结果

6.1.1 基线前向实验结果

设置	Zero-shot Acc	Zero-shot Macro-F1	Few-shot Acc	Few-shot Macro-F1
文本基线	45.10	29.36	47.55	33.92
多模态基线	33.10	22.09	49.31	35.85

6.1.2 LoRA 与 Mixed LoRA（epoch 1）

设置	Zero-shot Acc	Zero-shot Macro-F1	Few-shot Acc	Few-shot Macro-F1
文本基线	45.10	29.36	47.55	33.92
文本 LoRA	64.29	42.86	62.41	41.54
文本 Mixed LoRA	67.89	48.64	68.77	49.36
多模态基线	33.10	22.09	49.31	35.85
多模态 LoRA	63.41	44.46	61.84	42.73
多模态 Mixed LoRA	67.62	45.74	68.47	46.29

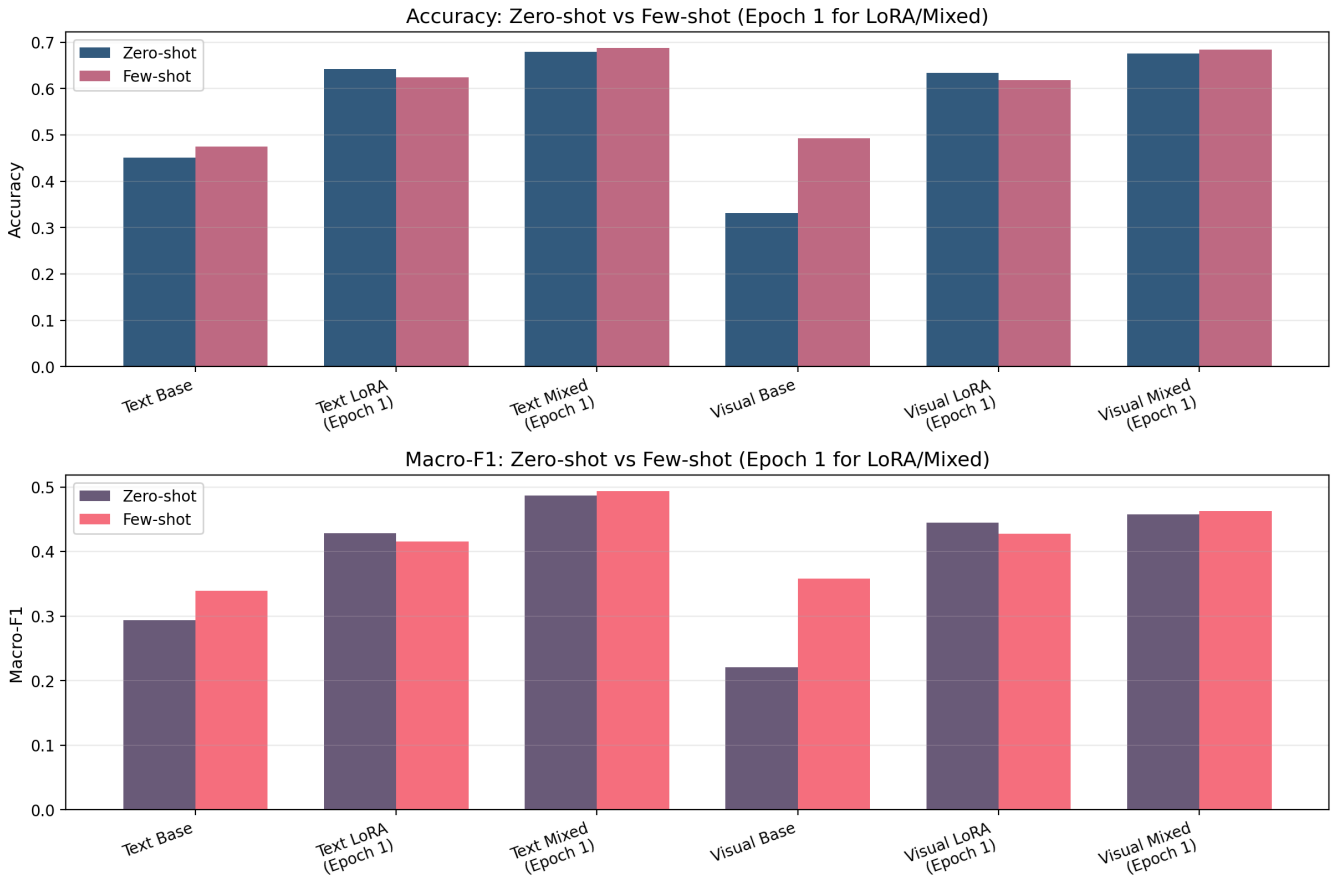

图1 Accuracy与Macro-F1性能结果图

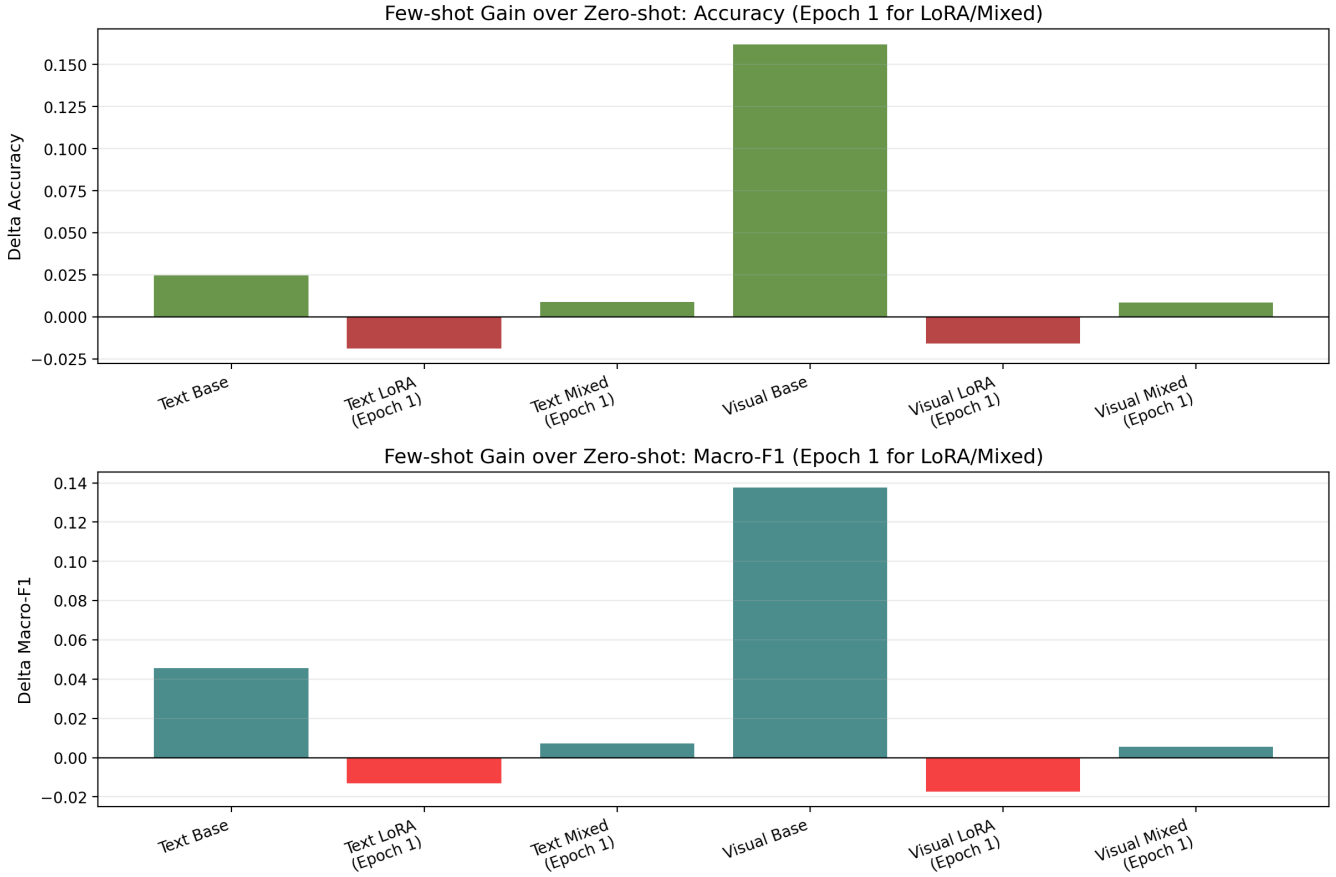

图2 few-shot相比zero-shot的性能提升图

6.1.3 实验结果混淆矩阵

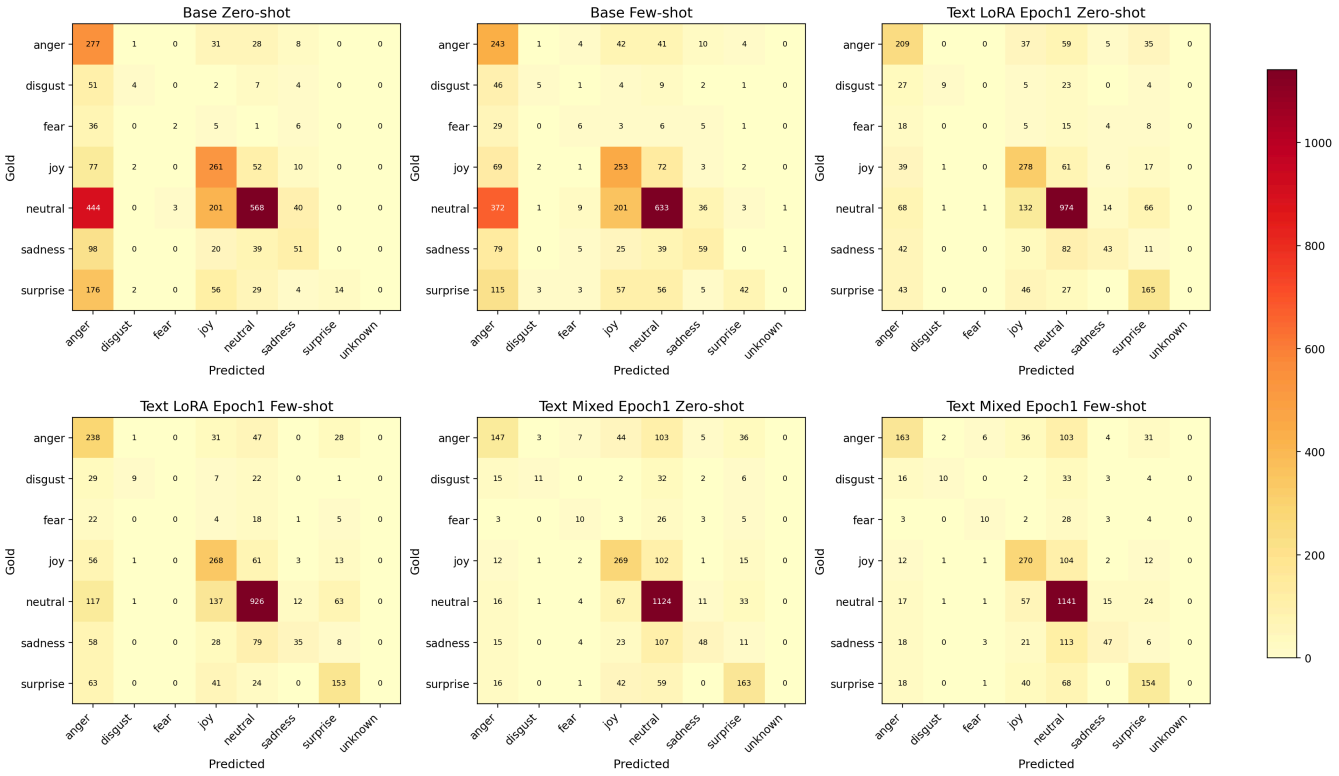

图3 文本实验混淆矩阵

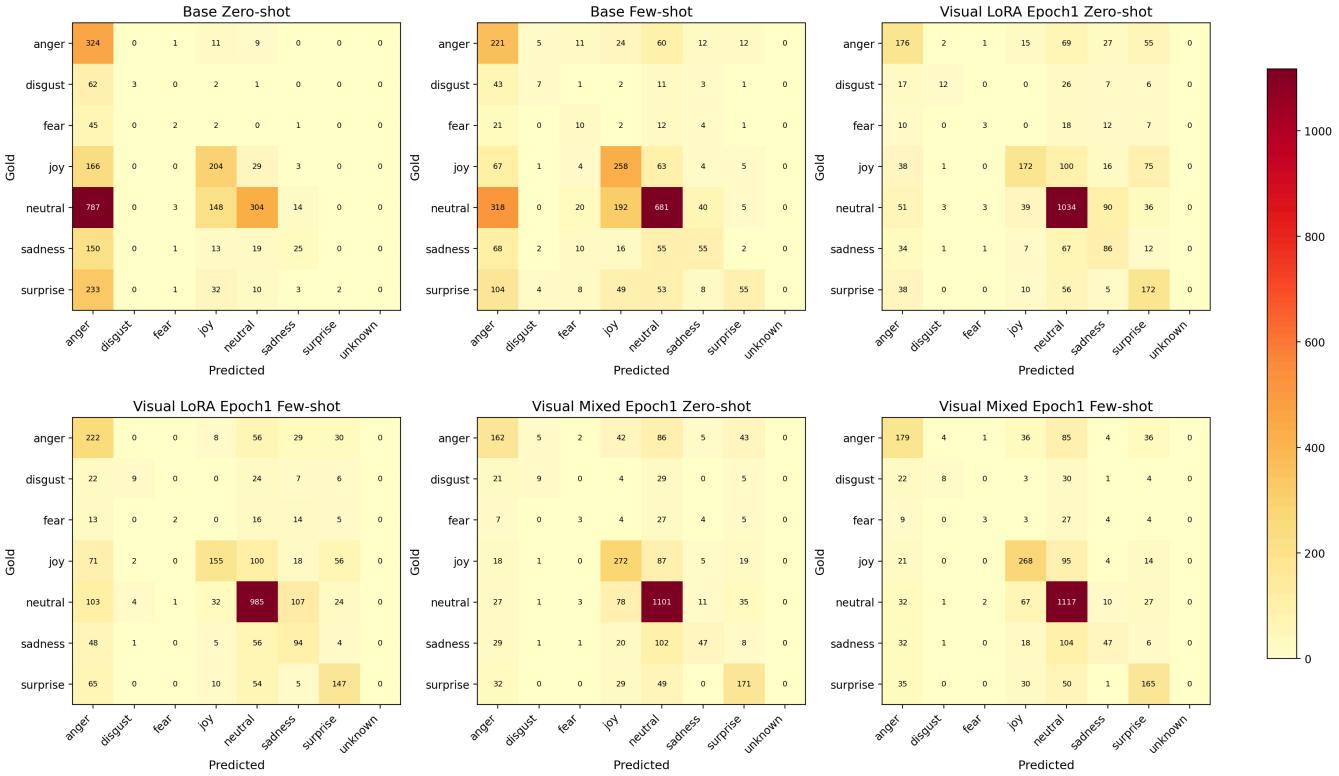

图4 多模态实验混淆矩阵

6.2 时间成本结果

6.2.1 训练时间（epoch 1）

配置	模态	训练耗时（小时）	相比文本 LoRA 增幅
文本 LoRA	文本	2.69	-
文本 Mixed LoRA	文本	3.73	+38.7%
多模态 LoRA	视觉 + 文本	50.06	+1760%
多模态 Mixed LoRA	视觉 + 文本	101.06	+101.8%（相对视觉 LoRA）

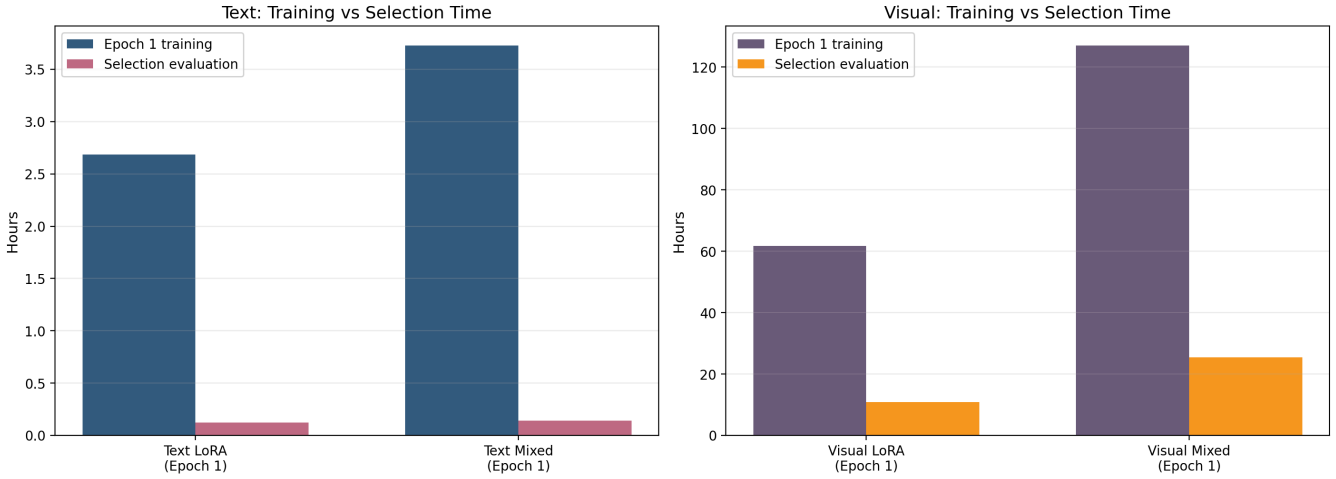

图5 训练和选择评估时间对比图

6.2.2 评估时间（test 集）

配置	模态	Zero-shot 耗时（分钟）	Few-shot 耗时（分钟）	Few-shot 相对增幅
文本基线	文本	7.5	8.6	+14.7%
文本 LoRA	文本	7.5	12.4	+65.3%
文本 Mixed LoRA	文本	9.2	10.5	+14.1%
多模态基线	视觉 + 文本	592.4	2342.3	+295.4%
多模态 LoRA	视觉 + 文本	653.8	2613.9	+300.0%
多模态 Mixed LoRA	视觉 + 文本	615.4	2748.1	+346.5%

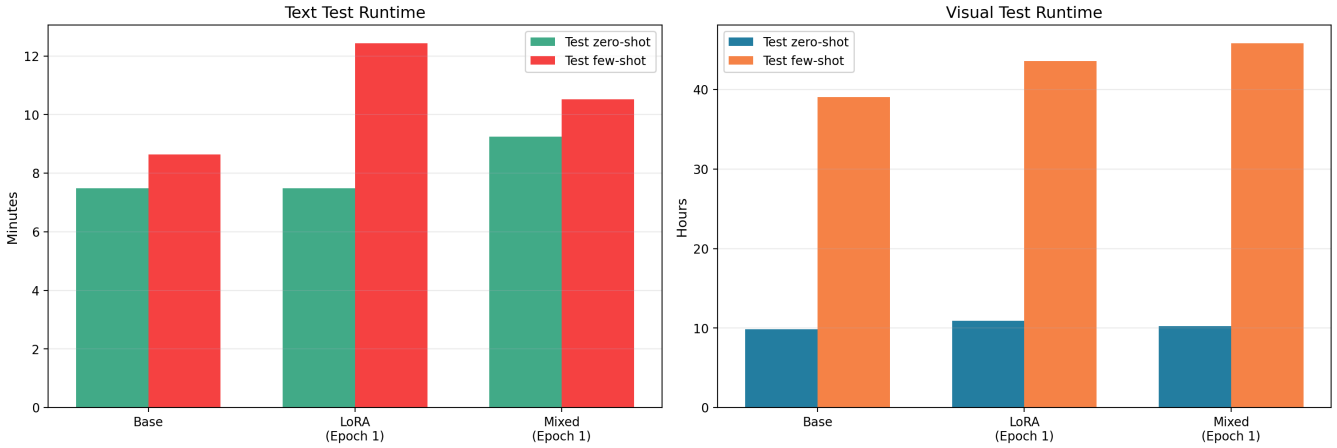

图6 测试集上评估时间对比图

6.2.3 全流程耗时（epoch 1 训练 + 测试集评估）

配置	总耗时（小时）	训练时长占比
文本 LoRA	2.88	~ 91.5%
文本 Mixed LoRA	3.83	~ 97.4%
多模态 LoRA	52.94	~ 94.6%
多模态 Mixed LoRA	104.06	~ 96.2%

6.3 视觉 token 开销

多模态 few-shot 平均输入 token 数从 zero-shot 的793增至2344，是耗时激增的核心原因。

---

7. 结果分析

7.1 “经验+3”在未微调模型中的作用

在原始模型上：

- 文本 few-shot 相比 zero-shot 提升：
  - `+2.45` Acc
  - `+4.56` Macro-F1
- 多模态 few-shot 相比 zero-shot 提升：
  - `+16.21` Acc
  - `+13.77` Macro-F1

这说明参考样本对原始模型尤其重要，特别是多模态场景。原因在于：

1. 多模态 zero-shot 的 prompt 更复杂
2. 模型需要同时理解文本和压缩后的视觉参考
3. fixed few-shot demos 实际上帮助模型校准了“如何读懂这种多模态输入”

7.2 为什么纯 zero-shot LoRA 会削弱 few-shot 增益

文本和多模态 LoRA 都使用了 zero-shot prompt 监督训练。结果显示：

- 微调后模型对 zero-shot 适应性很强
- 但 few-shot 输入格式与训练时不一致
- 因而模型对“前面多了 3 个参考样本”的输入没有学会稳定利用，甚至出现轻微退化

这与本文的假设一致：**如果训练数据只覆盖 zero-shot prompt，模型容易对该输入排布产生过拟合。**

7.3 Mixed LoRA 为什么有效

Mixed LoRA 中，训练集和选择评测都采用了：

- 50% zero-shot
- 50% few-shot

这相当于显式让模型在训练阶段同时看到两种 prompt 结构，因此：

1. 模型保留了强 zero-shot 能力
2. 也学会了 few-shot 输入下如何利用前缀中的参考样本

最终表现为：

- zero-shot 不降反升
- few-shot 能力恢复并再次超过 zero-shot

因此，Mixed LoRA 的结果支持如下结论：

few-shot 的作用并不是在 LoRA 后自然消失了，而是在“训练格式与测试格式不匹配”时被削弱；一旦训练阶段显式加入 few-shot prompt，这种能力就可以恢复。

此外，Mixed LoRA下 few-shot 相比 zero-shot 的效果提升幅度要小于基线条件下。这说明模型通过LoRA微调学到了相对丰富的先验知识，从而减弱了“经验+3”带来的效果提升。

7.4 时间成本分析

1. **训练效率**：
    - 文本微调轻量级，多模态微调计算密集，核心原因是视觉帧处理与token编码的高开销；
    - Mixed数据格式会增加训练耗时，但为保留 few-shot 能力是必要代价。
2. **评估效率**：
    - 多模态评估耗时远超文本（~75-80 倍），few-shot 因输入 token 数倍增进一步放大耗时（视觉 +300% 以上）；
    - 文本评估耗时占比低，多模态评估耗时虽高，但训练仍是全流程核心瓶颈。
3. **资源权衡**：多模态 few-shot 虽性能增益显著，但需付出数倍的计算时间与显存成本，实际应用中需平衡性能与资源开销。

---

8. 总结

本实验围绕“经验 + 3：参考样本对多模态情感分类的作用”展开，结合实验结果与时间成本分析，得出以下核心结论：

8.1 性能层面

1. 未微调模型中，few-shot 参考样本对情感分类性能提升显著，多模态场景下增益更大（Acc +16.21），验证了参考样本对复杂多模态输入的解释校准作用；
2. LoRA 微调显著增强了模型的情感分类能力，但纯 zero-shot 训练会导致模型对 few-shot 格式适配性下降；
3. Mixed LoRA（50% zero-shot + 50% few-shot）可有效恢复并保留 few-shot 优势，证明训练 - 测试格式一致性是大模型情感分类的关键因素。

8.2 资源层面

1. 文本与多模态实验的资源开销差异显著：视觉微调 / 评估的时间成本是文本的数十倍，few-shot 会进一步放大这种差异；
2. Mixed LoRA 虽增加训练耗时，但为保留 few-shot 能力的性价比合理，是参数高效微调中兼顾格式适配性的有效方案；
3. 多模态 few-shot 的性能增益需结合计算资源成本综合考量，实际落地中可根据硬件条件调整参考样本数或视觉帧采样策略。

8.3 核心观点

参考样本（经验 + 3）对多模态情感分类具有显著作用；这种作用在未微调模型中表现最为明显，而在参数高效微调场景下，只有当训练阶段显式覆盖 few-shot 输入格式时，这种优势才能被稳定保留，同时需兼顾训练 / 推理的资源开销平衡。

---

9. 参考文献

1. Poria S, Hazarika D, Majumder N, et al. Meld: A multimodal multi-party dataset for emotion recognition in conversations[C]//Proceedings of the 57th annual meeting of the association for computational linguistics. 2019: 527-536.
2. Hu E J, Shen Y, Wallis P, et al. Lora: Low-rank adaptation of large language models[J]. Iclr, 2022, 1(2): 3.
3. Brown T, Mann B, Ryder N, et al. Language models are few-shot learners[J]. Advances in neural information processing systems, 2020, 33: 1877-1901.
4. Team Q. Qwen3. 5-omni technical report[J]. arXiv preprint arXiv:2604.15804, 2026.

---

10. 数据集与代码补充

10.1 数据集下载

- MELD 官方仓库：https://github.com/declare-lab/MELD
- MELD 项目主页：https://affective-meld.github.io/

10.2 核心脚本清单

脚本	用途
`evaluate_qwen35_text_meld.py`	纯文本 zero/few-shot 评估
`evaluate_qwen35_visual_meld.py`	多模态 zero/few-shot 评估
`train_qwen35_text_lora_meld.py`	文本 LoRA 微调
`train_qwen35_text_lora_meld_mixed.py`	文本 Mixed LoRA 微调
`train_qwen35_visual_lora_meld.py`	视觉 LoRA 微调
`train_qwen35_visual_lora_meld_mixed.py`	视觉 Mixed LoRA 微调
`plot_meld_results.py`	结果可视化与绘图
`lora_meld_common.py`	通用 LoRA 微调配置和工具