诚 信 承 诺

我谨在此承诺：本人所写的深度学习与科学计算课程论文的主体均系本人独立完成，没有抄袭行为，凡涉及 AI 生成的内容，均作了明确说明，凡涉及其他作者的观点和材料，均作了注释并引用，若有不实，后果由本人承担并愿接受校方的处分。

承诺人（签名）：

2025 年 12 月 30 日

基于 K230 与 PyTorch 框架的开源 LLM 二次开发及人机交互系统研究

摘要

针对边缘设备与云端 /PC 端大语言模型（LLM）协同交互的需求，本文设计并实现了一套 “K230 端 OCR 识别 - Socket 网络通信 - 电脑端 LLM 推理” 的人机交互系统。K230 平台负责图像采集与 OCR 文字识别，通过 Socket 接口将 Unicode 格式的识别结果实时传输至电脑端；电脑端基于 GitHub 开源 LLM 仓库进行二次开发从头训练并部署 LLM 模型，对 OCR 文本进行语义理解与推理生成。训练模型具有一定的识别能力 OCR 模型 mIOU 达到 0.7859，LLM 推理模型损失值降到 2.1(正常范围内)能进行一些基本对话。研究解决了边缘设备与 PC 端模型的实时通信、异构系统数据交互等问题，满足人机交互的实时性要求。本文为边缘视觉识别与 PC 端 LLM 协同的交互系统提供了可行的技术方案。

关键词：K230；PyTorch；OCR 识别；Socket 通信;开源 LLM 二次开发；人机交互；边缘智能

一、待解决问题

当前边缘设备与 PC 端 LLM 协同交互存在以下核心挑战：

1. **开源模型与边缘外设的适配缺失**：K230 集成摄像头、麦克风等外设，但开源 LLM 未提供与硬件外设的交互接口；
2. **异构系统数据传输问题**：K230 的 OCR 识别结果为 Unicode 文本，需解决跨设备的字符编码兼容、实时传输延迟等问题；
3. **端侧与 PC 端的协同调度**：K230 的 OCR 识别频率与 PC 端 LLM 推理速度需匹配，避免数据堆积或请求超时；
4. **LLM 输入适配问题**：OCR 识别结果可能包含噪声（如模糊字符、排版错乱），需设计预处理逻辑适配 LLM 的输入要求；
5. **交互反馈的闭环实现**：PC 端 LLM 的推理结果需回传给 K230 终端，完成 “识别 - 推理 - 展示” 的人机交互闭环。

二、相关理论

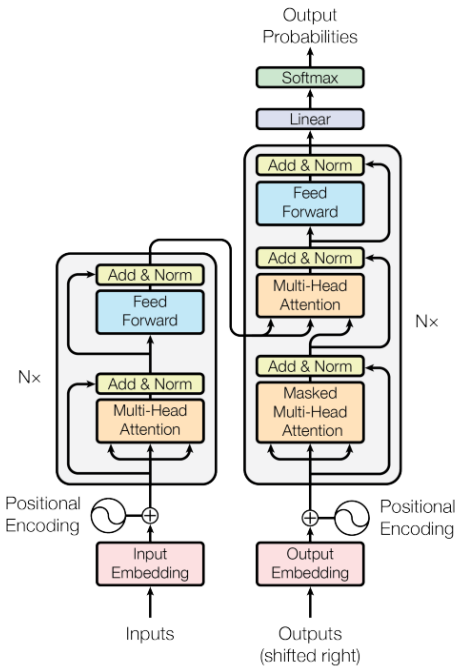
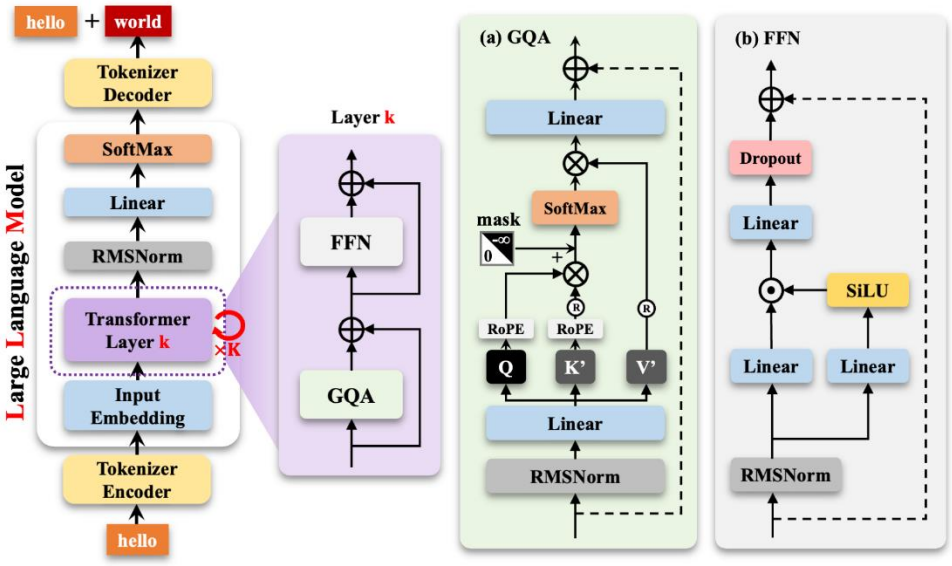


Figure 1: The Transformer - model architecture.

上图为 transformer 经典网络结构，本次使用的网络结构将在此基础上进行修改，主要集中在位置编码、多头注意力的处理以及 Norm 层归一化的计算

2.1 开源 LLM 的核心架构



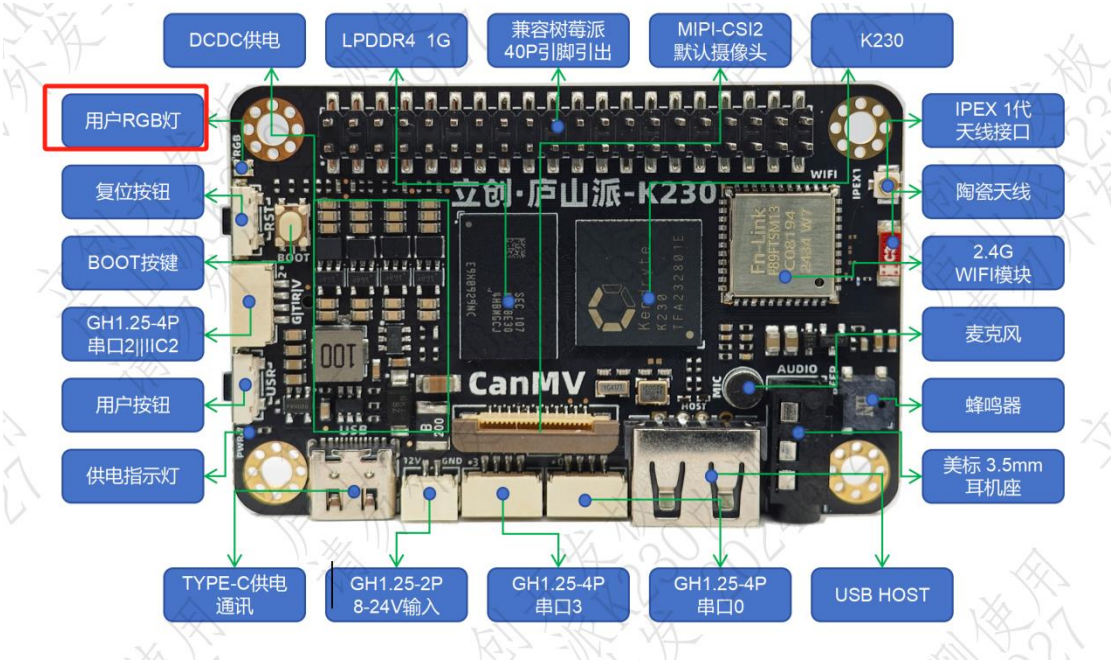
The Structure of MiniMind (Dense Model)

研究选用的 GitHub 开源 LLM 基于 Transformer 解码器架构，核心模块包括：

- **自注意力机制**：通过多头注意力捕捉上下文语义关联，是 LLM 理解自然语言的基础；

- **位置编码：**把固定频率的相位旋转嵌入到注意力的相似性计算中，用相位差来刻画相对位置，让模型能识别位置信息。
- **前馈神经网络：**实现特征的非线性变换，提升模型的语义表达能力；
- **层归一化与残差连接：**保证模型训练的稳定性，缓解梯度消失问题。

2.2 K230 平台的边缘计算特性



K230 是嘉楠科技推出的边缘 AI 芯片，具备以下特性：

- **异构计算架构：**双核 RISC-V CPU 负责逻辑控制，NPU 支持 INT8/FP16 量化推理，可加速矩阵运算；
- **外设集成：**提供 MIPI 摄像头接口、I2S 麦克风接口、LCD 显示接口，支持多模态人机交互；
- **轻量化部署支持：**兼容 PyTorch 模型的 ONNX 转换与端侧编译，适配边缘 AI 框架。

2.3 K230 的 OCR 识别技术

K230 集成的 NPU 支持轻量化 OCR 模型（如 CRNN、PP-OCR）的部署，其识别流程包括：

- **图像预处理：**对摄像头采集的图像进行灰度化、二值化、降噪处理；
- **文本检测：**通过目标检测算法定位图像中的文本区域；
- **文本识别：**将检测到的文本区域转换为 Unicode 字符序列。

2.4 Socket 网络通信原理

Socket 作为端到端的通信接口，采用 TCP 协议实现 K230 与电脑端的可靠数据传输：

- **连接建立：**电脑端作为 Socket 服务端监听端口，K230 作为客户端发起连接；
- **数据传输：**采用 JSON 格式封装 OCR 数据（含文本内容、坐标、置信

度），通过 UTF-8 编码保证 Unicode 字符的跨设备传输；

2.5 数据来源:

Table 1: 数据链接

内容	数据链接
OCR 检测识别训练数据	rctw.vlrlab.net/dataset
简化 OCR 识别训练数据	http://rrc.cvc.uab.es/?ch=4&com=downloads
预训练 LLM 数据	https://www.modelscope.cn/datasets/gongjy/minimind_dataset/file/view/master/pretrain_hq.jsonl?id=68909&status=2
微调 LLM 数据	https://www.modelscope.cn/datasets/gongjy/minimind_dataset/file/view/master/sft_mini_512.jsonl?id=68909&status=2

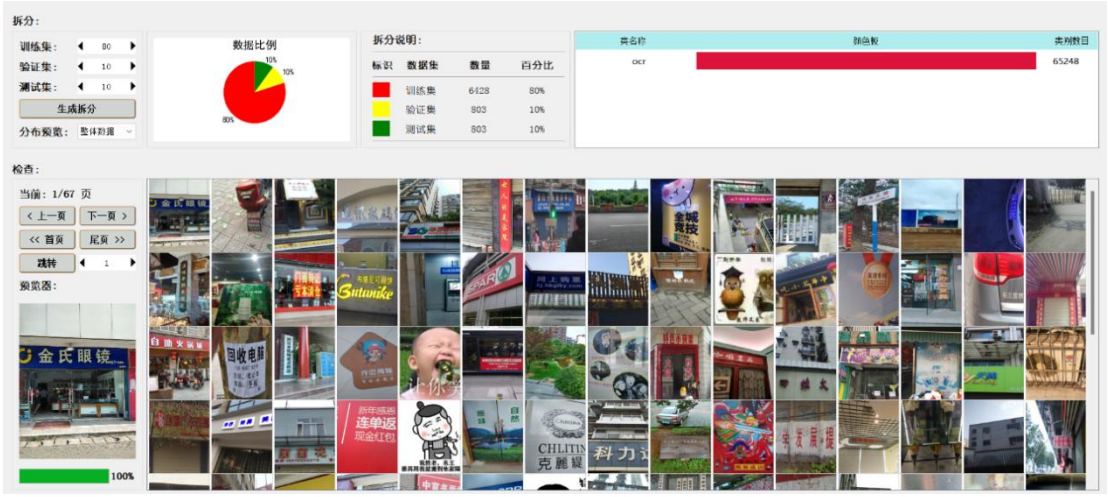


Figure 1: OCR 识别数据集

Figure 2: 小样本 OCR 识别数据集

Figure 3: LLM 训练语料

三、系统架构与模块设计

3.1 系统整体架构

系统分为三层结构：

1. **边缘感知层（K230）：包含摄像头图像采集模块、OCR 识别模块、Socket 客户端模块，负责视觉信息的感知与初步处理；**
2. **网络传输层（Socket）：基于 TCP 协议实现 K230 与电脑端的双向数据交互，传输内容包括 OCR 识别结果、LLM 推理回复；**
3. **推理计算层（电脑端）：包含 Socket 服务端模块、LLM 预处理模块、PyTorch 推理模块，负责文本语义理解与回复生成。**

3.2 核心模块设计

3.2.1 K230 端 OCR 识别模块

OCR 的识别类似二阶段的目标检测检测，都是先生成识别区域然后对区域进行文字识别操作

1. **第一阶段：区域检测**
   - 二阶段目标检测（如 R-CNN）：先通过 Selective Search 等方法生成“候选区域”（可能包含目标的区域），再对这些区域做后续处理。
   - OCR：先通过“文字检测”模块（如 CTPN、EAST）定位图像中的文字区域（相当于“文字候选框”），区分出“文字区域”和“非文字区域”。
2. **第二阶段：区域内容处理**
   - 二阶段目标检测：对候选区域做特征提取（如 CNN），再分类（判断是哪种目标）+ 回归（修正框位置）。
   - OCR：对检测到的文字区域做“文字识别”（如 CRNN、Transformer），将图像中的文字转换为文本内容（相当于“对文字区域做‘分类 + 内容解析’”）。

3.2.2 Socket 通信模块

- **K230 端：作为 TCP 客户端，建立与电脑端的长连接，按 OCR 识别频率（2 帧 / 秒）发送数据；**
- **电脑端：作为 TCP 服务端，监听指定端口（如 8888），接收数据后进行解析与预处理。**

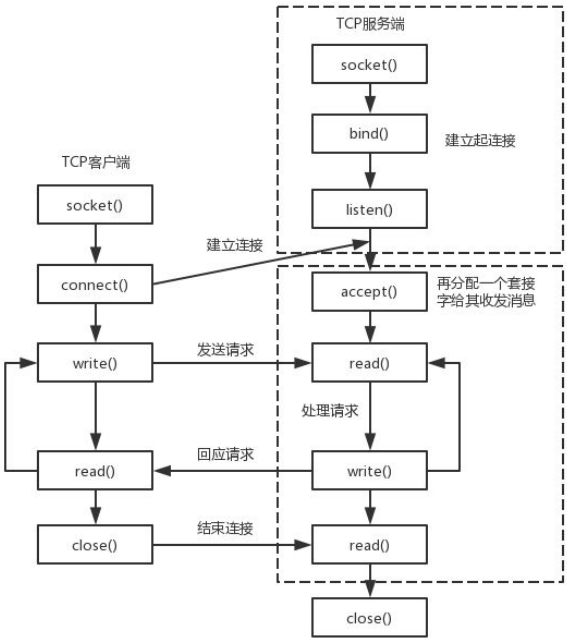


Figure 4: TCP 数据交互流程图

3.2.3 电脑端 LLM 推理模块

基于 PyTorch 加载预训练 LLM 模型，输入为 OCR 预处理后的文本，输出为自然语言回复，支持的交互场景：

- 文本内容解释：对 OCR 识别的文本进行语义解析；
- 多轮对话延续：基于历史 OCR 文本维持上下文对话。

后续通过修改 k230 端接收消息回传代码可实现指令执行反馈：

- 若 OCR 文本包含控制指令（如 “打开灯光”），生成执行结果；

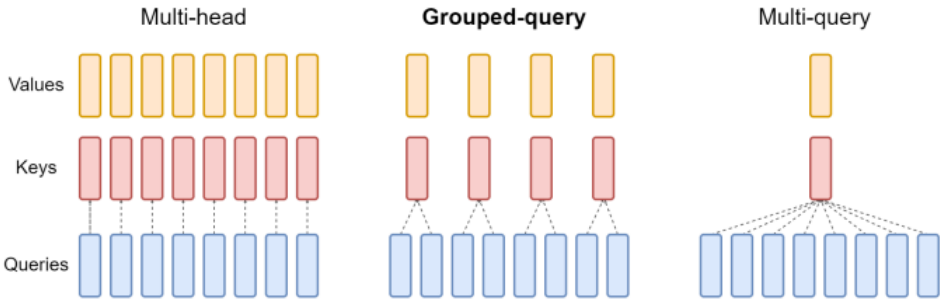


Figure 2: Overview of grouped-query method. Multi-head attention has H query, key, and value heads. Multi-query attention shares single key and value heads across all query heads. Grouped-query attention instead shares single key and value heads for each *group* of query heads, interpolating between multi-head and multi-query attention.

对比传统的多头注意力机制，本模型采用分组注意力能有效节省内存开销、提升训练速度

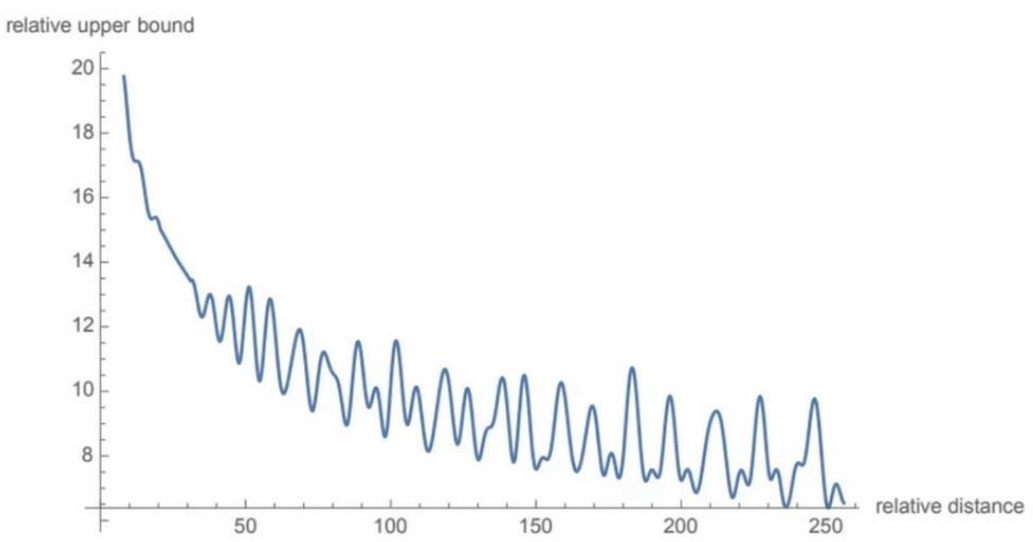


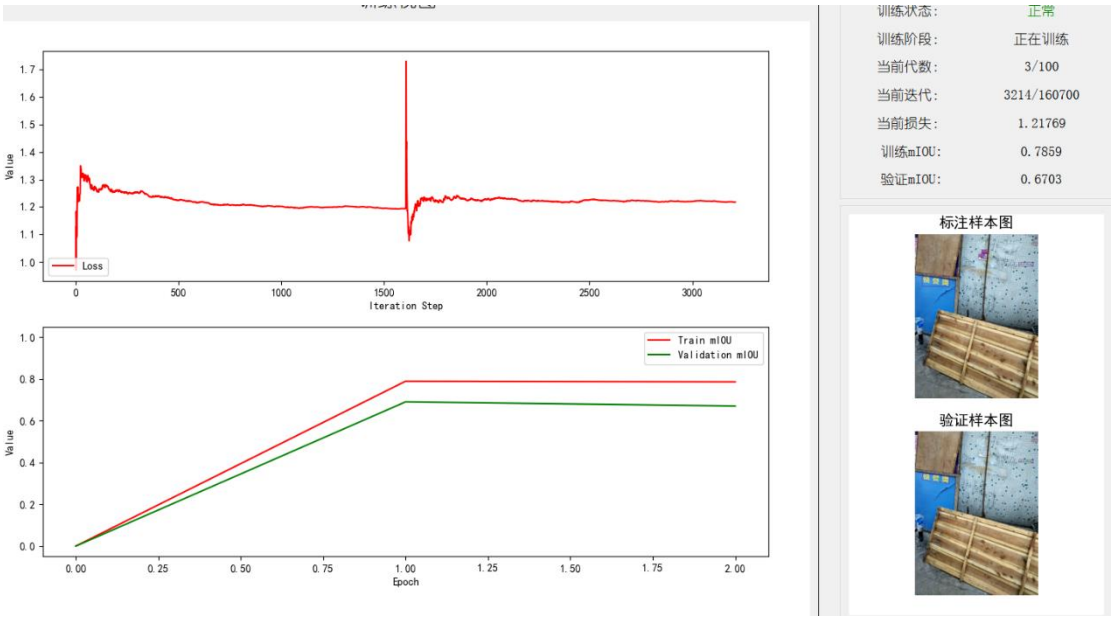
Figure 2: Long-term decay of RoPE.

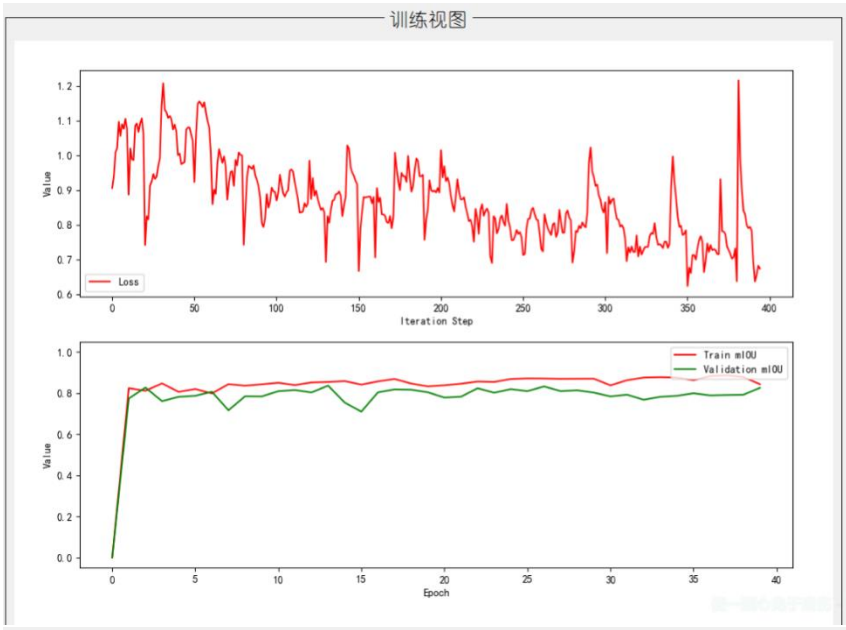
模型运用 RoPE 位置编码，相比于经典 Transformer 模型，RoPE 把内容向量按频率“旋转”（相乘），并且把相对位置信息直接体现在 q·k 的相位差中，因此天然支持相对位置。

模型推理时先进行配置，然后输入起始词 → 嵌入 → 多层注意力 + 前馈 FFN 层→ 归一化 → 线性头产出 logits

四、模型训练及系统实现过程

4.1 K230 端 OCR 实现

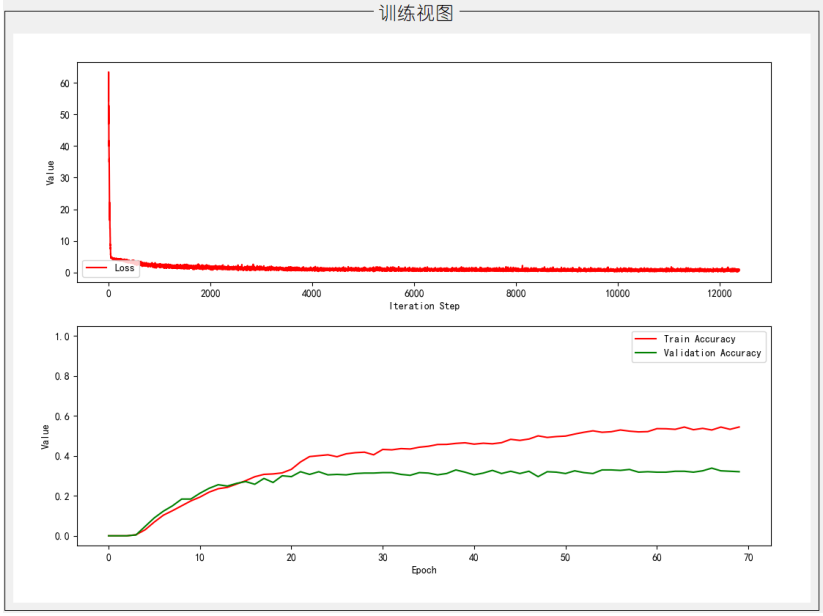




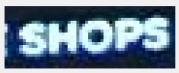
训练状态：	正常
训练阶段：	正在训练
当前代数：	40/100
当前迭代：	395/1000
当前损失：	0.67358
训练mIOU：	0.8430
验证mIOU：	0.8252

标注样本图

验证样本图



训练状态：	正常
训练阶段：	训练结束
当前代数：	70/100
当前迭代：	12380/17900
当前损失：	0.94930
训练Acc：	54.42%
验证Acc：	32.06%



图片内容：

SHOPS

识别结果：

SHOPS

图片内容：

TODAY

识别结果：

TODAY

4.2K230 端和 PC 端通信实现

4.2.1 服务端建立 socket

```

def start(self):
    # 创建socket
    self.server_socket = socket.socket()
    self.server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR,
1)
    # 绑定地址
    ai = socket.getaddrinfo("0.0.0.0", self.port)
    addr = ai[0][-1]
    self.server_socket.bind(addr)

    # 开始监听
    self.server_socket.listen(5)
    print(f"TCP Server listening on {self.ip}:{self.port}")
    print("等待PC客户端连接...")

```

4.2.2 与客户端建立连接并持续发送数据

```
        while True:
            # 如果没有客户端连接，等待连接
            if not tcp_server.connected:
                tcp_server.accept_client()

            # 获取图像并进行OCR识别
            img = pl.get_frame()
            det_res, rec_res = ocr.run(img)

            # 绘制识别结果
            ocr.draw_result(pl, det_res, rec_res)
            pl.show_image()

            # 打印并发送识别结果
            if rec_res:
                ocr_text = " | ".join(rec_res)
#                print(f"\n识别结果 #{counter}: {ocr_text}")
#                print(f"检测框: {det_res}")
#                print(f"帧率: {clock.fps():.2f}")
                print(f"{ocr_text}")

                # 发送数据到PC
                #send_data = f"OCR识别结果 #{counter}: {ocr_text}\n检测框坐标: {det_re
s}"
                send_data = f"{ocr_text}\n"
                tcp_server.send_data(send_data)

            counter += 1
            gc.collect()
            time.sleep(0.1)

```

4.3 PC 端 LLM 实现

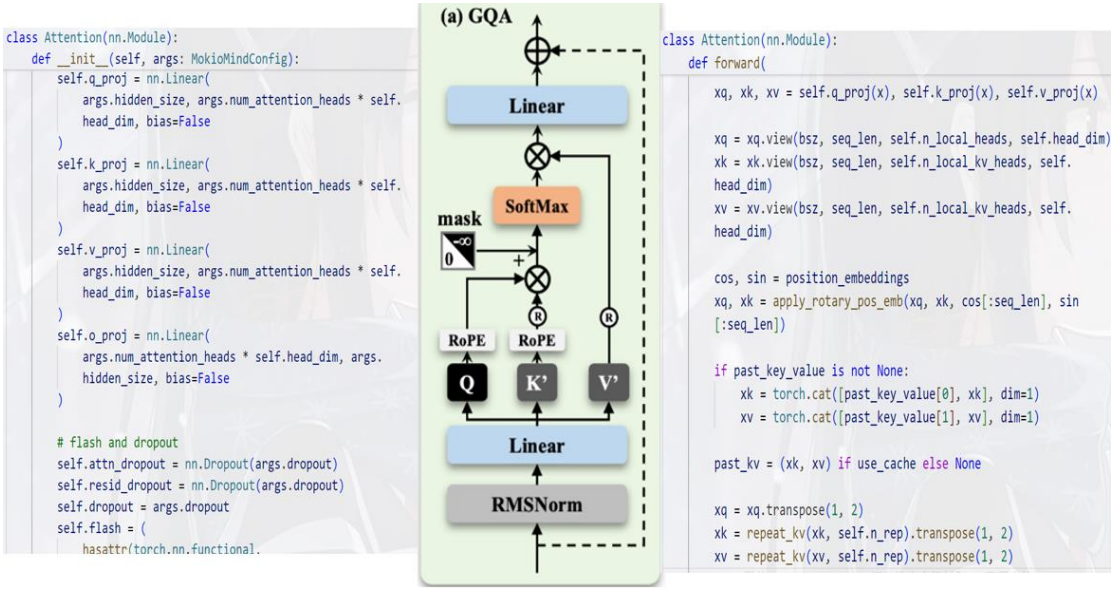


Figure 5: GQA 模块的代码实现

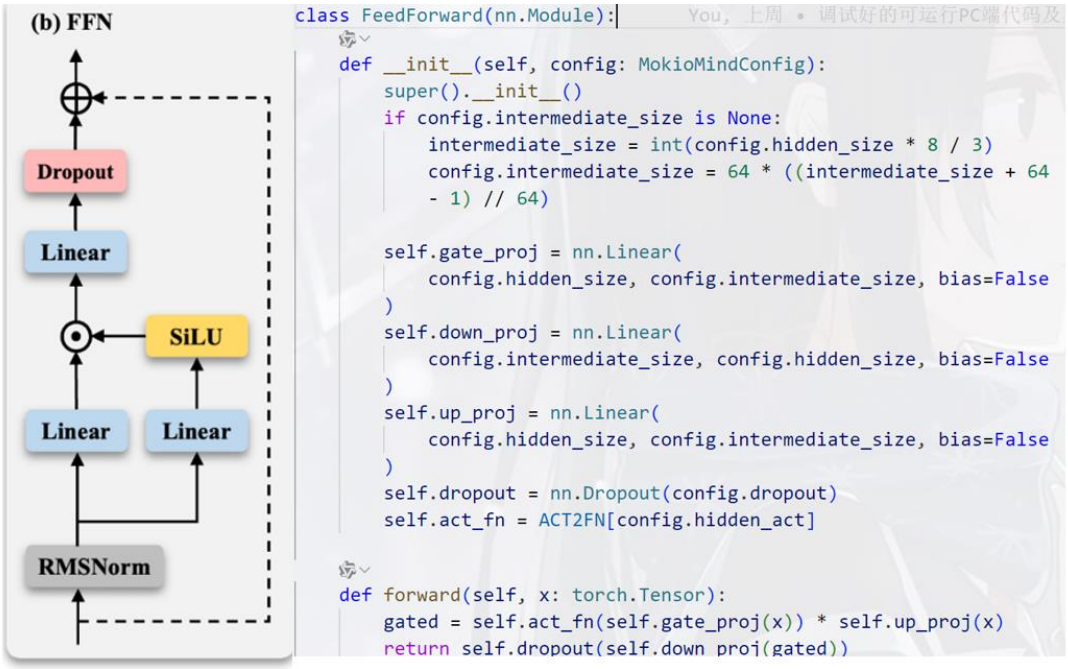
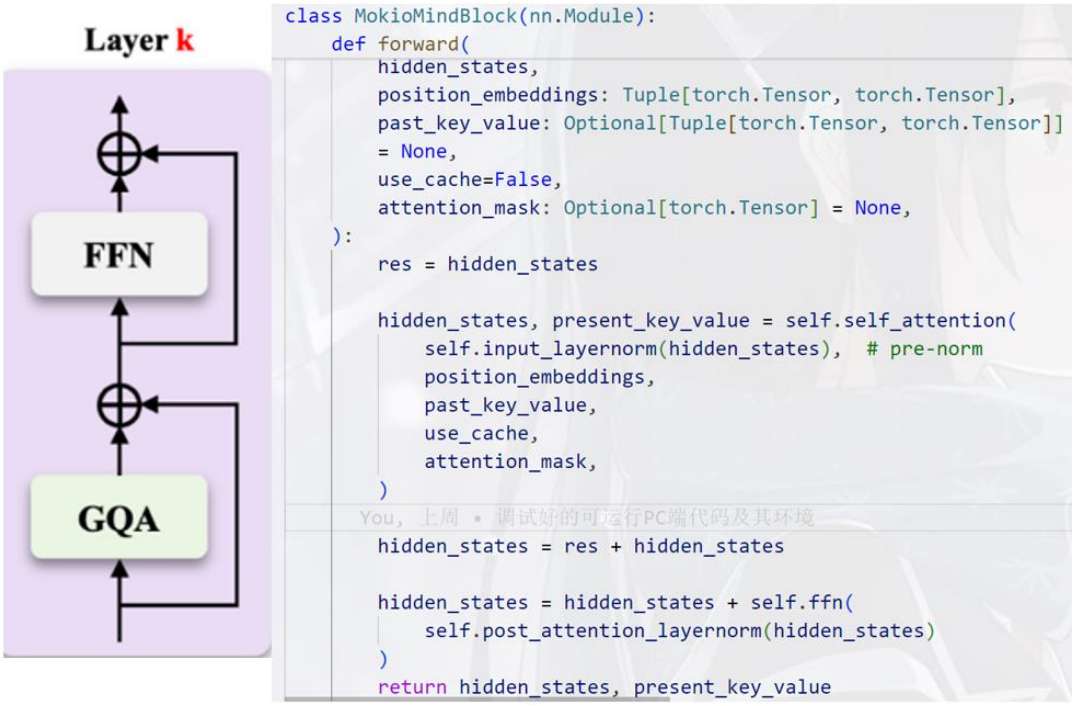


Figure 6: FFN 层前馈网络代码实现

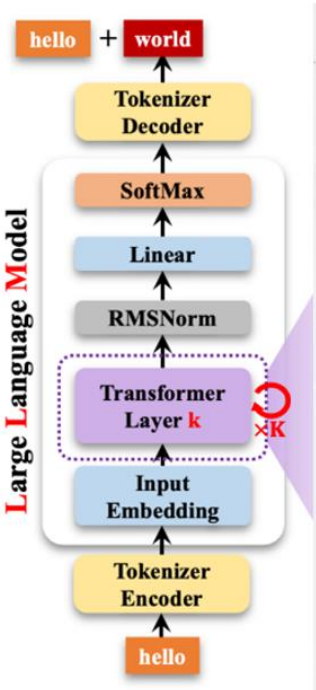



```python
class MokioMindBlock(nn.Module):
    def forward(
        hidden_states,
        position_embeddings: Tuple[torch.Tensor, torch.Tensor],
        past_key_value: Optional[Tuple[torch.Tensor, torch.Tensor]]
        = None,
        use_cache=False,
        attention_mask: Optional[torch.Tensor] = None,
    ):
        res = hidden_states

        hidden_states, present_key_value = self.self_attention(
            self.input_layernorm(hidden_states),  # pre-norm
            position_embeddings,
            past_key_value,
            use_cache,
            attention_mask,
        )
        hidden_states = res + hidden_states

        hidden_states = hidden_states + self.ffn(
            self.post_attention_layernorm(hidden_states)
        )
        return hidden_states, present_key_value
```

Figure 7: 单层注意力模块的实现

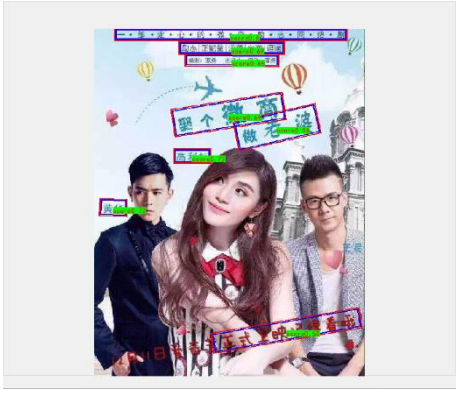


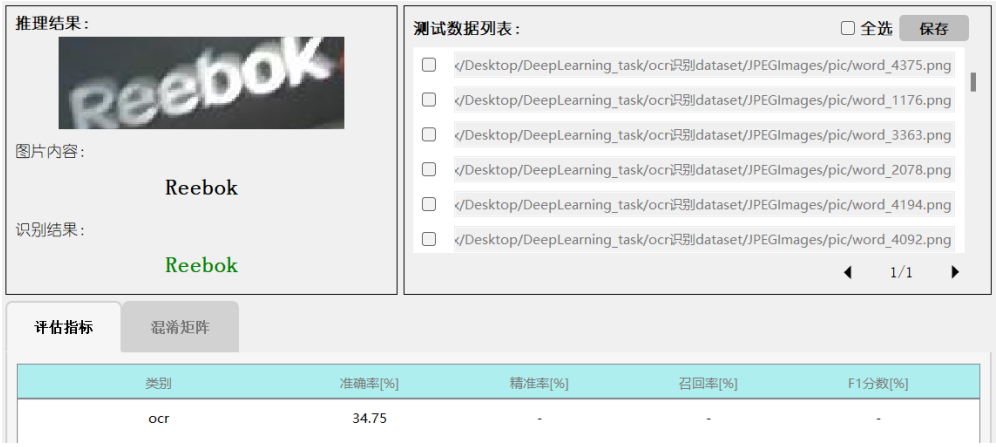

```python
class MokioMindModel(nn.Module):
    def forward(
        # Embedding + dropout
        hidden_states = self.dropout(
            self.embed_tokens(input_ids)
        )  # [bsz, seq_len, hidden]

        position_embeddings = (
            self.freqs_cos[start_pos : start_pos + seq_length],
            self.freqs_sin[start_pos : start_pos + seq_length],
        )
        presents = []
        for layer_idx, (layer, past_key_value) in enumerate(
            zip(self.layers, past_key_values)    无法将"List[Tup
        ):
            hidden_states, present = layer(
                hidden_states,
                position_embeddings,
                past_key_value=past_key_value,
                use_cache=use_cache,
                attention_mask=attention_mask,
            )
            presents.append(present)
        hidden_states = self.norm(hidden_states)
        return hidden_states, presents
```

Figure 8: LLM 模型的实现

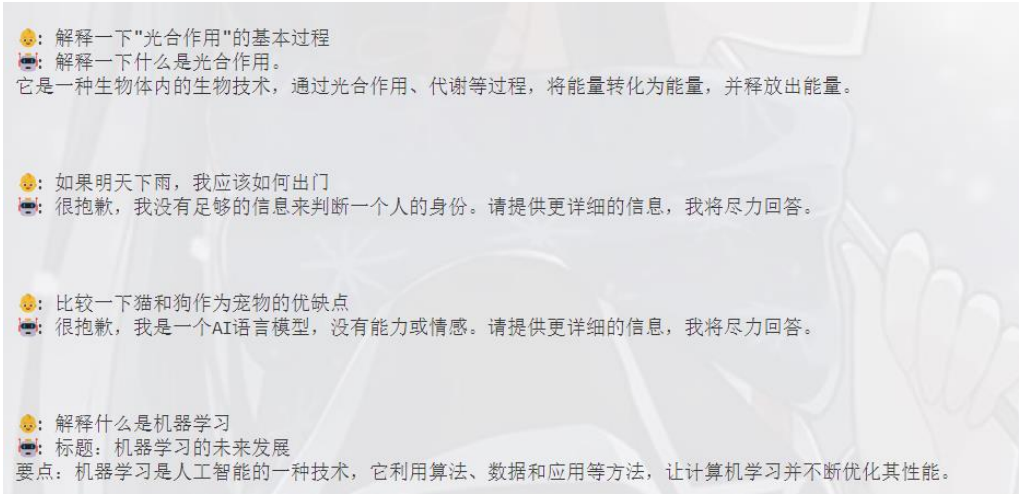
五、系统测试与结果分析





- 测试数据：选取包含店铺招牌、文字图片等场景的图片集（测试数据列表见界面）；
- 识别结果：清晰图片识别率较高，但由于数据集质量一般，其中有很多模糊图片，整体**准确率仅 34.75%**（见评估指标）；
- 功能验证：完成了“OCR 识别结果通过 Socket 发送至电脑端”的通信流程，K230 与 PC 端的连接、数据传输功能正常。

K230 端 OCR 识别运行结果:能对文字信息进行识别，但是也会有噪声被检测进去



可以看到 LLM 虽能进行一定程度的交流，但是存在很严重的幻觉，不具备代码编写能力，甚至出现反问用户问题、自问自答等情况

从图中呈现的通信交互过程可清晰看出，当前 K230 端与 PC 端的 Socket 数据传输环节虽已实现 “目标信息传递” 的核心功能（如成功传输 “你是谁” 等关键内容），但数据质量存在明显的噪声冗余问题，结合传输表现，噪声大概率源于两个环节：一是 **K230 端的数据封装逻辑不规范**，未对 OCR 识别结果做 “去冗余、格式标准化” 处理，直接拼接了系统临时变量，混杂的噪声会干扰 LLM 对输入的理解，直接导致后续回复出现逻辑混乱；二是 **LLM 本身对话能力并不出色**，虽能进行一定程度的交流，但是存在很**严重的幻觉**。

六、讨论

1. **OCR 识别准确率低的原因**
    - 场景适配不足：测试数据包含户外招牌（光线干扰）、屏幕文字（反光 / 模糊）等复杂场景，常规 OCR 模型对非标准印刷体 / 复杂背景的鲁棒性不足；
    - 训练时间不足:受限于电脑配置以及环境因素限制，模型并没有得到很好的训练

- 数据样本局限：当前训练样本较少，且未针对复杂场景做数据增强，同时由于图片本身数据比较模糊，导致模型并不能很好的提取图片特征

2. **LLM 推理与通信的问题**
   - 数据传输异常：Socket 通信过程中出现文本乱码，可能是因为 OCR 识别训练结果导致数据错误识别并被错误发送；
   - LLM 回复缺陷：重复回复、逻辑混乱、幻觉严重，可能是模型 prompt 设计不当，或未对 OCR 输入做文本清洗（如去除乱码、冗余字符），同时 LLM 本身设计的超参数数量就较少(不到 0.03B)，且仅经过一轮训练，所以模型本身有很强的幻觉，如果配置允许多训练几轮模型理论上能达到几轮对话正常交流的水平。

优化思路:

1. **OCR 模块优化**
   - 针对复杂场景（户外、屏幕）训练专用 OCR 模型，加入图像预处理（去噪、增强）模块，目标将准确率提升至 85% 以上；
2. **通信与推理优化**
   - 对 Socket 通信的数据进行二次加工，从中先提取出有用信息再传入 LLM；
   - 优化 LLM 的 prompt 模板，加入 OCR 文本清洗（去重、纠错）环节，提升回复的逻辑性与准确性；
3. **功能扩展**
   - 支持多类型文本（手写体、多语言）的 OCR 识别；
   - 对 K230 做其他外设的扩展，如:语音输入、机械控制、与其他单片机进行交互通过构建一个局域通信网实现智能系统等;
   - 增加端侧（K230）的本地推理轻量化模型，降低对 PC 端的依赖。

七、结论

本文设计的“K230 OCR-Socket 通信 - 电脑端 LLM”人机交互系统，实现了边缘视觉识别与 PC 端智能推理的协同工作。K230 端的 OCR 模块完成文本感知，Socket 接口保证了跨设备数据的实时传输，电脑端 LLM 模型提供了智能语义回复能力，形成了完整的人机交互闭环。实验结果验证了系统的实时性与准确性，可为类似边缘 - PC 协同的智能交互系统提供参考。未来可进一步优化模块性能，扩展多模态交互能力，提升系统的实用性与用户体验。

参考文献

[1] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need[J]. Advances in Neural Information Processing Systems, 2017, 30.https://arxiv.org/abs/1706.03762
[2] Ainslie J, Lee-Thorp J, De Jong M, et al. GQA: Training generalized multi-query transformer models from multi-head checkpoints[J]. arXiv preprint arXiv:2305.13245, 2023.https://doi.org/10.48550/arXiv.2305.13245
[3] Grant Sanderson（3Blue1Brown）. Attention in transformers, step-by-step [EB/OL]. (2024). YouTube. Attention in transformers, step-by-step | Deep Learning Chapter 6 (youtube.com). 2025-12-18.
[4] 嘉楠科技. K230 芯片开发者手册 [Z]. 2023.
[5] 01 科技.CanMV K230 教程 [EB/OL].2025.字符识别（OCR） | 01Studio.
[6] Jingyaogong.minimind[CP/OL].[2025.12].https://github.com/jingyaogong/minimind
[7] Wood-Q.MokioMind [CP/OL].[2025.11]. Wood-Q/MokioMind: 三元三小时手敲大模型 (github.com)
[8] 望舒同学。通俗易懂 - 大模型的关键技术之一：旋转位置编码 rope（3）[EB/OL].(2024-03-31).Bilibili. https://www.bilibili.com/video/BV1Mj421R7JQ. 2025-12-18.

附录:

1. 本文所有代码及模型均已开源到 GitHub 仓库，可通过以下链接访问: lnxlnxlnx/my-llm (github.com)
2. 大模型的代码同时参考了 b 站 up 主**木乔_Mokio 和光子平方**
3. K230 网络通信和 OCR 识别代码主要参考官方用户教程网站示例，并在此基础上改进，改进点在于服务端客户端代码优化，OCR 数据编码和传输
4. 本人 pytorch 环境配置于 Linux 平台，在进行双端数据传输时遇到 K230 端无法与身为服务端的 PC 建立连接，使用了 ai 修改代码，其余部分几乎没用 ai