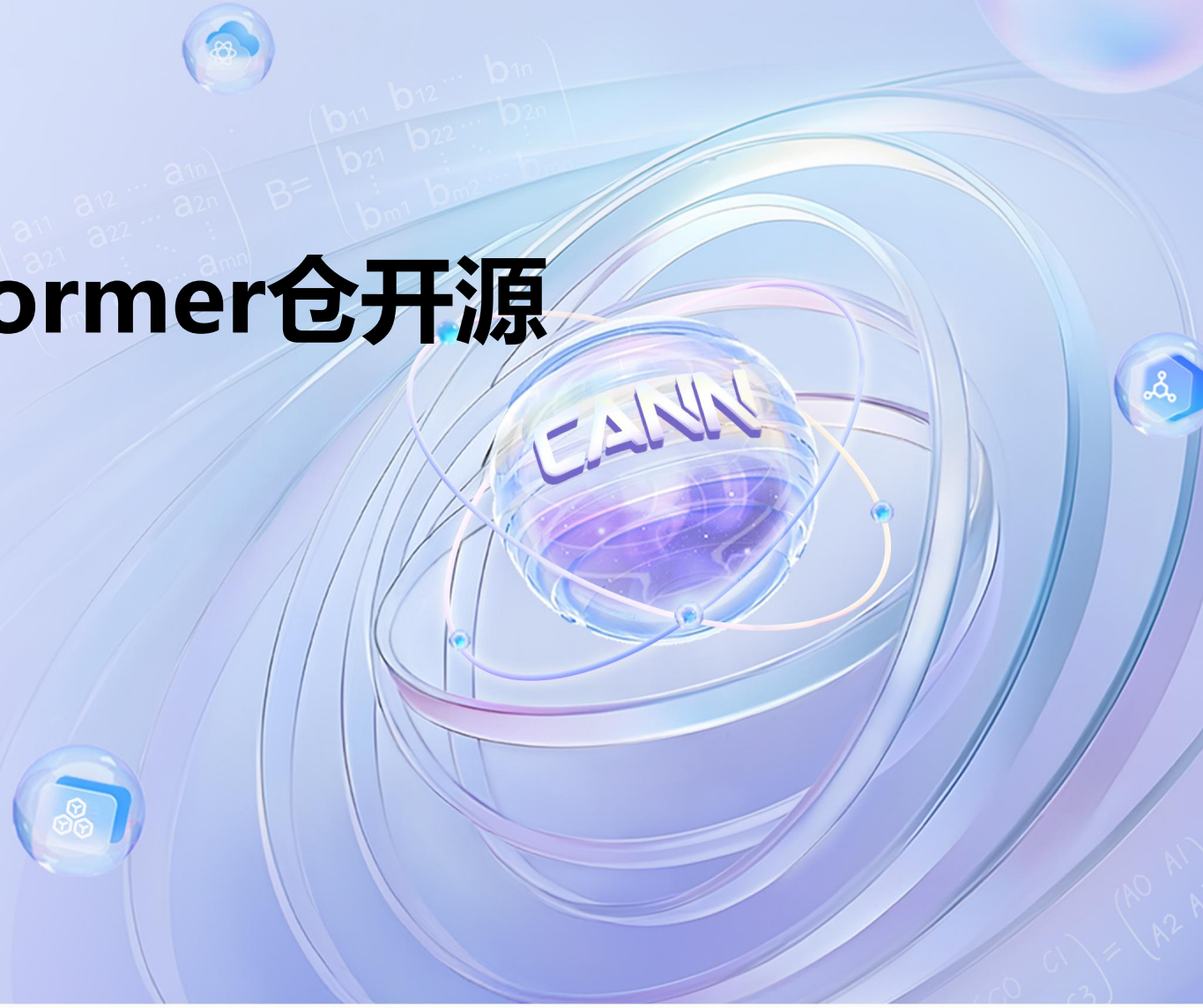
CANN ops-transformer仓开源

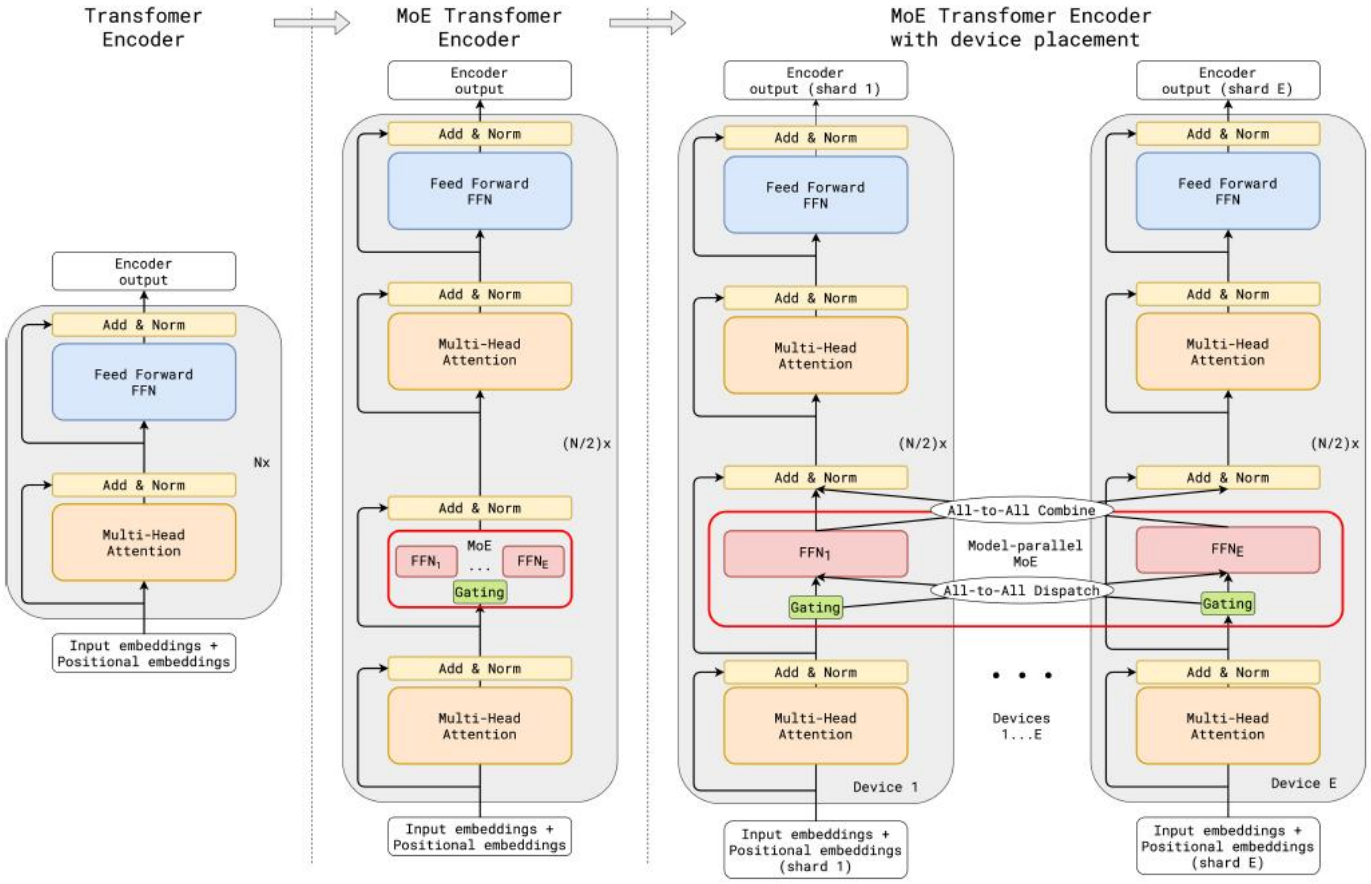


https://gitcode.com/cann

CANN

ops-transformer仓基础介绍(一)

ops-transformer是CANN（Compute Architecture for Neural Networks）算子库中提供transformer类大模型计算的进阶算子库。



- 生态优先，兼容主流生态使用习惯
- 高内聚，低耦合原则
- 隔离原则等

https://gitcode.com/cann

CANN

ops-transformer仓基础介绍（二）

本仓包含六大类算子，Attention（注意力类算子）、 MC2（通信和计算融合类算子） 、GMM（Grouped Matmul相关算子）、FFN 类算子、MOE 类算子和Posembedding (位置编码类算子)。

Transformer仓

- **Attention**：FlashAttention类一系列衍生融合算子，如PFA、IFA、FIA和FAG等;
- **MC2类**：通算融合类算子，包含一系列通信和计算融合类算子，如all_gather_matmul，moe_distribute_dispatch和moe_distribute_combine等算子;
- **GroupMatmul类**：实现分组矩阵乘计算及相关融合算子
- **FFN类**：提供FFN和MoeFFN的计算算子;
- **MOE类**：提供MOE计算中，包括moe_init_routing、moe_gating_top_k、moe_finalize_routing等算子

算子列表

CANN

ops-transformer 仓目录结构介绍

```
├── build.sh                    # 项目工程编译脚本
├── cmake                       # 项目工程编译目录
├── common                      # 项目公共头文件和公共源码
├── attention                   # attention类算子
├── docs                        # 项目文档介绍
├── example                     # 使用通用算子开发和调用示例
├── ...
├── moe                         # moe类算子
├── posembedding                # posembedding类算子
├── mc2                         # mc2类算子
│   ├── all_gather_matmul       # all_gather_matmul算子所有交付件，如Tiling、Kernel等
│   │   ├── CMakeLists.txt      # 算子编译配置文件
│   │   ├── docs                # 算子说明文档
│   │   ├── examples            # 算子使用示例
│   │   ├── op_graph            # 算子构图相关目录
│   │   ├── op_host             # 算子信息库、Tiling、InferShape相关实现目录
│   │   │   └── op_api          # 算子aclnn接口实现目录
│   │   ├── op_kernel           # 算子kernel目录
│   │   └── README.md           # 算子说明文档
│   ├── ...
│   └── CMakeLists.txt          # 算子编译配置文件
├── scripts                     # 脚本目录，包含自定义算子、kernel构建相关配置文件
├── tests                       # 测试工程目录
├── CMakeLists.txt
├── README.md
├── build.sh                    # 项目工程编译脚本
├── install_deps.sh             # 安装依赖包脚本
└── requirements.txt            # 本项目需要的第三方依赖包
```


- **Docs**：包含aclnn接口介绍等md文档；
- **Examples**：算子调用示例；
- **op_graph**：算子图模式场景交付件，例如InferDatatype等；
- **op_host**：算子基础host侧交付件，例如Tiling、aclnn接口等；
- **op_kernel**：算子device侧交付件，包含算子核心实现。

CANN

算子基本开发流程介绍

基于开源仓开发新算子，核心交付件包含Tilling与Kernel两部分内容

1. 前提条件：

   ① 环境部署：开发算子前，请确保基础环境已安装，例如依赖的驱动、固件、CANN软件包等。

   ② 算子设计：分析实际业务诉求，合理设计算子规格，包括算子输入、输出、属性的数据类型、shape等。

2. 工程创建：开发算子前，需按要求创建算子目录，方便后续算子的编译和部署。

3. Tiling实现：实现Host侧算子Tiling函数。

4. Kernel实现：实现Device侧算子核函数。

5. aclnn适配：自定义算子推荐aclnn接口调用，需完成二进制发布。如需入图，请参考附录。

6. 编译部署：通过工程编译脚本完成自定义算子的编译和安装。

7. 算子验证：通过常见算子调用方式，验证自定义算子功能。

详细开发指导

- **Host侧Tiling实现：**
  由于NPU中AI Core内部存储无法完全容纳算子输入输出的所有数据，需要将数据分批次进行处理。切分数据的算法称为Tiling算法或者Tiling策略。
- **Device侧Kernel实现：**
  Kernel实现即算子核函数实现，通过调用计算、数据搬运、内存管理、任务同步API，实现算子逻辑。其核心逻辑基本上都为计算密集型任务，需要在NPU上执行。

CANN

Thank you.

社区愿景：打造开放易用、技术领先的AI算力新生态

社区使命：使能开发者基于CANN社区自主研究创新，构筑根深叶茂、跨产业协同共享共赢的CANN生态

Vision: Building an Open, Easy-to-Use, and Technology-leading AI Computing Ecosystem

Mission: Enable developers to independently research and innovate based on the CANN community and build a win-win CANN ecosystem with deep roots and cross-industry collaboration and sharing.

上CANN社区获取干货

关注CANN公众号获取资讯

https://gitcode.com/cann