MPCViT: 基于SecretFlow的Vision Transformer框架

# 神经网络架构优化

底层协议：基于半诚实威胁模型的SS方案

上层模型：Vision Transformer

基于Transformer编码器的视觉模型

包括输入embedding层、注意力层、非线性记录函数等模块

模型作用：图像分类

## 研究原因

1. SEMI2K和Cheetah协议下，注意力中的softmax和mlp中的GeLU占用了大部分延迟

（主要通信瓶颈）

注意力中的求最大值、求倒数和指数耗时严重，想要去掉这些操作提高效率

1. 含有不同操作的注意力机制的模型的准确率和延迟对比

融合推理的高效率和高准确率

1. 不是所有的注意力都同等重要

## 算法流程

目标：

权衡模型准确率、推理效率

融合高准确率注意力机制、低延迟注意力机制

具体流程：

1. 设计合适的搜索空间：三种颗粒度，粗->细分别是，

Transformer层级 -> 注意力头级 -> 注意力行级(token级)

1. MPC感知的神经搜索算法

每个注意力都有一个架构参数α，搜索过程中自动决定保留哪个注意力

这个搜索算法可以微分，搜索效率高

考虑到注意力的开销不同，把MPC条件加入损失函数的正则项

1. 基于延迟限制的架构参数二值化

α较大是高准确率类型，较小是低延迟类型

1. 重训练异构注意力ViT

## 具体算法

问题：直接训练搜索后的异构ViT模型，性能不好（准确率显著下降）

解决：多粒度自蒸馏方式

1. 引入原始softmax ViT模型，让异构ViT去学习，没有额外训练、推理开销
2. 传统基于logits的蒸馏（软标签）
3. 基于特征的token蒸馏（最后一层特征）

# 搭建隐私推理框架

## SPU

优点：可用性、可扩展性、高性能

## 隐私推理和通信参数配置

隐私推理协议和节点配置：

{

    "nodes": {

        "node:0": "127.0.0.1:9327",

        "node:1": "127.0.0.1:9328",

    },

    "devices": {

        "SPU": {

            "kind": "SPU",

            "config": {

                "node\_ids": [

                    "node:0",

                    "node:1"

                ],

                "spu\_internal\_addrs": [

                    "127.0.0.1:9327",

                    "127.0.0.1:9328"

                ],

                "runtime\_config": {

                    "protocol": "SEMI2K",

                    "field": "FM64",

                    "enable\_pphlo\_profile": true,

                    "enable\_pphlo\_trace": true,

                    "enable\_hal\_profile": true

                }

            }

        },

        "P1": {

            "kind": "PYU",

            "config": {

                "node\_id": "node:0"

            }

        }

    }

}

网络参数配置：

通过tc进行配置

## 用JAX搭建ViT模型

class PatchEmbedding

class GeLU

class MultiHeadAttention

* class Transformer

## 隐私推理实际计算

main函数

参考secretnote\_example\_mpcvit\_model文件

# 实验结果

1. 和近年的SOTA模型Linformer, THE-X, MPCFormer相比，MPCViT在不同数据集、不同架构上，降低了很多延迟、提高了一定准确率
2. 消融实验中，两种粒度的知识蒸馏对于异构ViT训练很重要
3. 搜索算法在不同模型架构、不同超参数情况下，一致性较好，可以扩展到不同规模的架构和数据集上，有扩展性