首先是反思了之前的实验代码

1. 读了XCiT以后，发现我之前的代码不是真的Channel Attention，而是和它一样的Spatial Cross-covariance. 所以针对这点，对代码做了改动。假设输入为 ，那么：

Spatial Cross-covariance 的更新公式是：

（以上三者都是对每个patch的linear mapping，相当于是在spatial维度上的一一映射）

（这个是cross covariance）

而Channel Attention的更新公式应该是：

（这三者都是对每个feature 维度的linear mapping，相当于是在channel维度上的一一映射）

（这个是self-attention的attention weight）

1. 从原来的三种Unit，增加到了四种Unit，分别是Spatial Self-attention (SSA), Channel Self-attention (CSA), Spatial Multi-layer Perceptron (SMLP)和Channel Multi-layer Perceptron (CMLP)。

另外，我用的是swin-transformer的framework，所以之前的实现是：

其中

这里在代码实现上实际有两层MLP，但其实只要一层MLP就能实现在不同head组之间的信息交换，所以我删掉了一层MLP，Top-1 Acc 反而提高了0.16%。

对于如何在weight-sharing的情况下实现NAS训练，我在和Jiajun Liu老师开会的时候探讨了几个实现方向，如下：

1. 加入一个gate 来控制。

比如对于Spatial Self-attention和Spatial MLP，他们的更新公式都为：

如果选择SSA unit，那么；如果选择SMLP unit，那么。当的时候和MHSA是一样的公式；当足够大的时候近似一个单位阵。

这样子的话MLP和MHSA其实weight是共用且共同训练的。但是这个方法没有在Channel和Spatial之间实现sharing。

1. 还有一种更复杂的方法，并且也不是完全的weight-sharing，更偏向semi-supervised learning，如下：

对于每个输入 ，先会被分成四个head组，分别记作。每层的输出是 ，其中 是随机选择的某个unit。

先设计以下两种loss：

, 这个是对concatenate以后的输出做的交叉熵损失函数

, 这个是希望每个Unit输出都和最后的输出接近，这样相当于每个Unit组都可以用来单独进行prediction，以此来对每种Unit进行训练。

在训练的时候，每层都要有四种不同的Unit，但是这四种Unit的排序可以不同。比如可以是(CSA, SMLP, SSA, CMLP) 或者 (SMLP, SSA, CMLP, CSA)。每个epoch训练完都会随机更新每层的Unit排列顺序。

搜索的时候则允许Unit重复使用，比如可以出现 (SSA, SSA, SMLP, CSA) 这样的layer。并且同一层相同的Unit使用的weight一致（weight sharing体现在这里）。

Shuffle+ViT

去掉MLP的实验

不做weight sharing的训练

对channel和spatial做speed balance

Network optimization + lightweighten