# 论文笔记

## Actionlet-Dependent Contrastive Learning for Unsupervised Skeleton-Based Action Recognition（基于动作小波的无监督骨架动作识别对比学习）

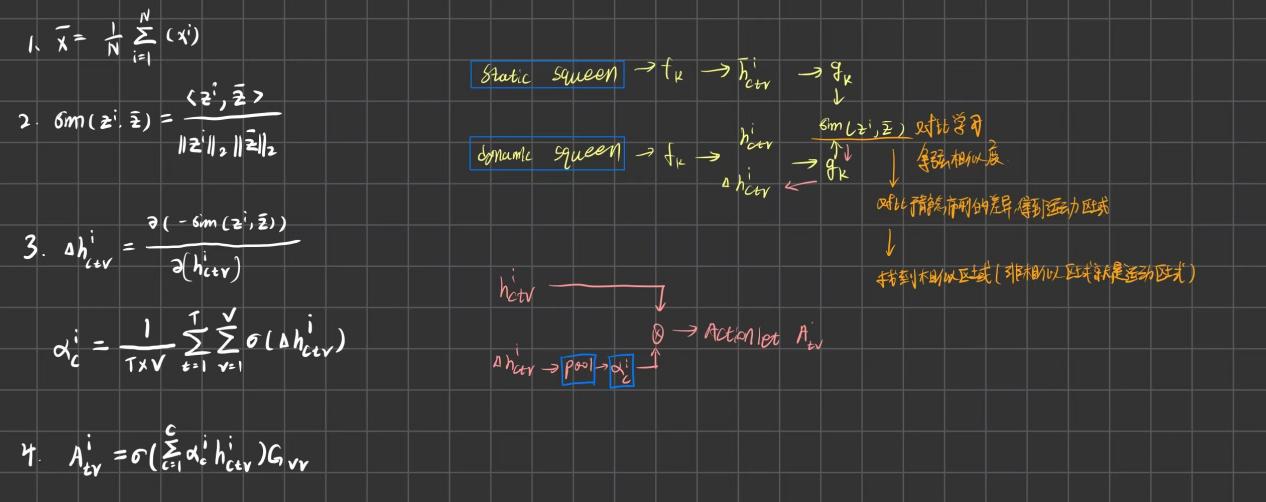
在以往中是平等去对待运动区域与静态区域，这对于最后的识别来说是复影响的。所以我们提出了一个动作依赖对比学习方法（ActCLR）。

### ActCLR

具体来说，通过与没有运动的静态锚进行对比，以无监督的方式提取骨架数据的运动区域，作为动作。然后，以 actionlet 为中心，构建了运动自适应数据转换方法。

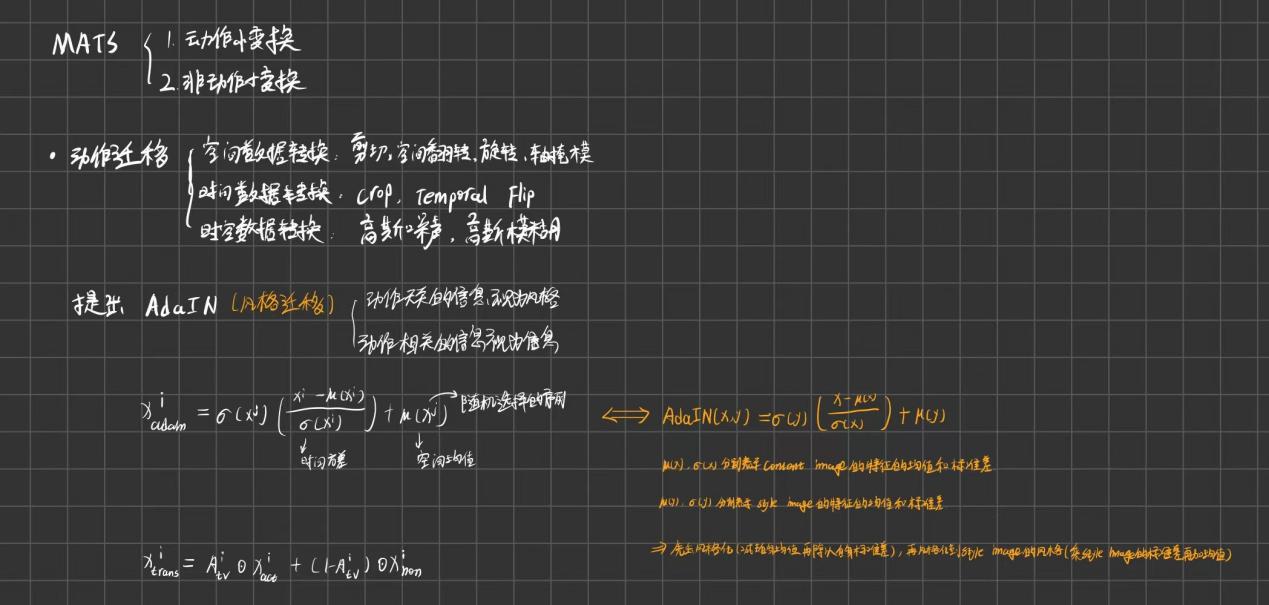
#### 如何获得静态序列？

直接计算一个数据集中所有骨架序列所有节点的平均值，这样就可以得到一个骨架序列，这个骨架序列就是相对的静态序列。意味着数据集不能太小，一定要大，越大越好。但是如果不同数据集的静态序列相互引用的话效果没有本数据集好，除非数据集足够大，可以无限逼近。



### MATS

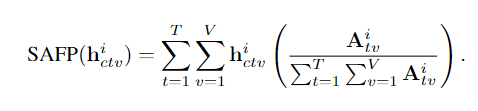
我们通过运动自适应数据转换策略模块(MATS)对不同区域进行不同的数据转换。



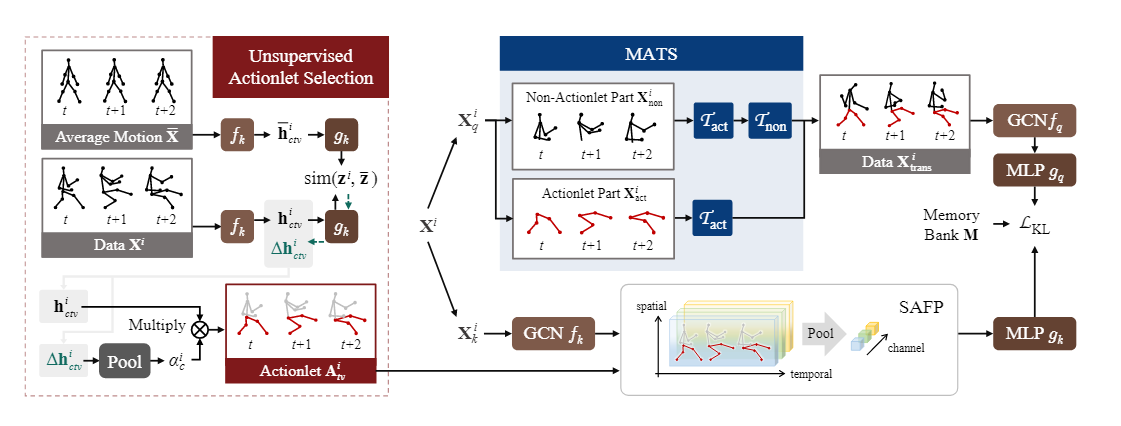
### SAFP

语义感知特征池模块(SAFP)，对动作let区域的特征进行聚合，从而更好地进行动作建模。

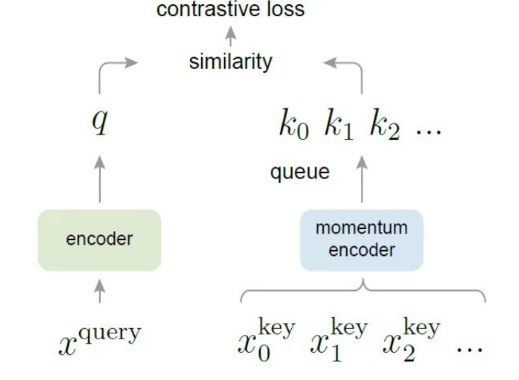
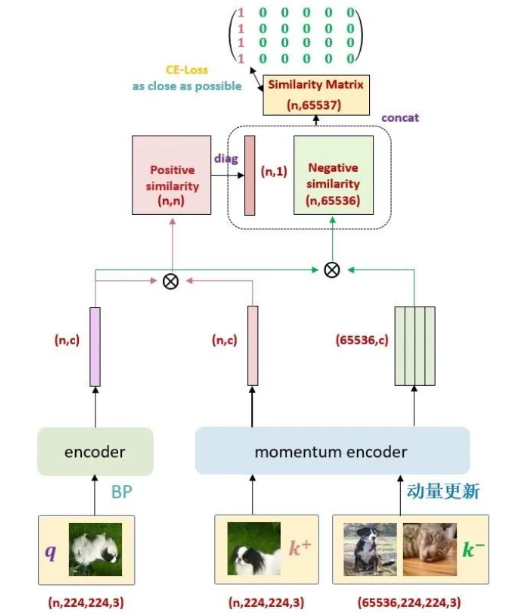
该方法只关注动作区域的特征表示，从而减少了其他静态区域对运动特征提取的干扰。



### 模型结构



# MoCo V2



## MoCoV2队列问题

在模型初始化的时候创建，在第一个批次运行第一次的时候队列是空的，这个时候对比学习效果不好，但是随着迭代的进行，是不断依次把负样本加入到队列中，每次用完key的时候就把这个key加入到队列中，给下一个批次数据用。当然这个队列是循环队列。

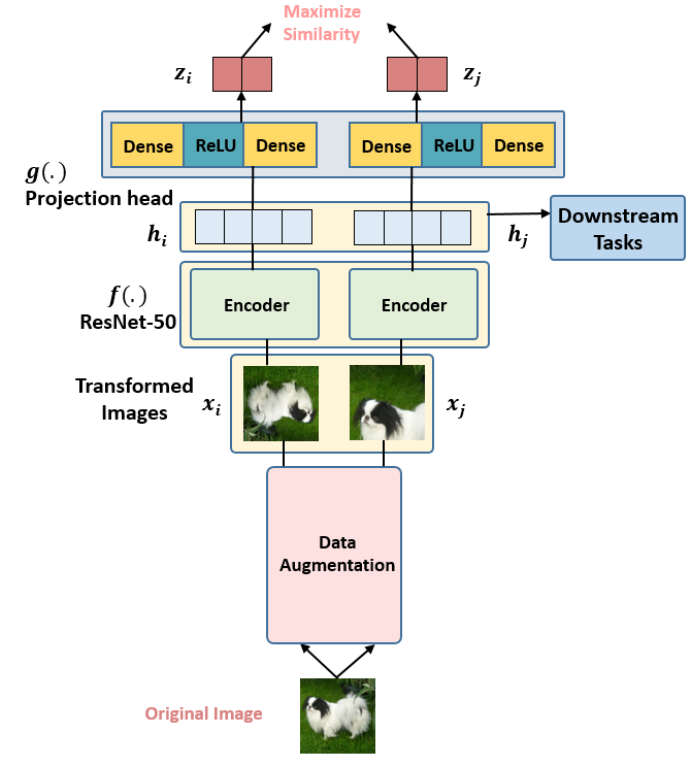
## MoCoV2与SimClr模型的区别

首先两者都是用的对比学习但是在其实现上是有不同的。

比如在MoCoV2上重点提出了队列，这个队列是一个非常大数字，起码是万起步，所以如果一个批次是32，则可以保存很多个批次数据了，这样当前的批次的训练数据就会有大量的负样本数据进行学习，提升学习效果。而且这种方式用不到显存，相应的就不能用梯度更新负样本数据了，所以需要用动量更新去更新负样本数据，正样本数据还是保持梯度更新。

那么在SimClr上提出了利用图像增强来实现对比学习，这是一个非常大的创新点，但是缺点也很明显就是需要batch非常大，因为必须要有足够的负样本数据进行学习，但是这样就要求显存很大了。MoCoV2继承了它的优点并解决了其缺点。

# SimClr

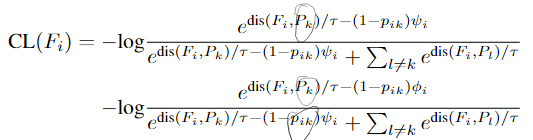


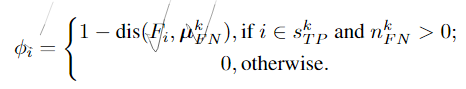
# Learning Discriminative Representations for Skeleton Based Action Recognition

## FR-Head

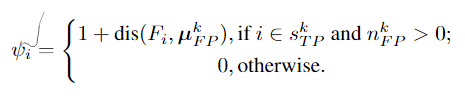
这是模型的创新点，做了一个即插即用的模块，这个模块用的是对比学习来实现的，计算思路如下：

计算出真正例（tp）、假负例（fn）、假正例（fp），然后来计算损失。而且是基于当前批次的。

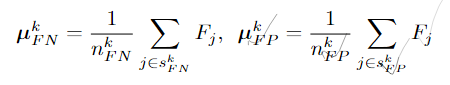




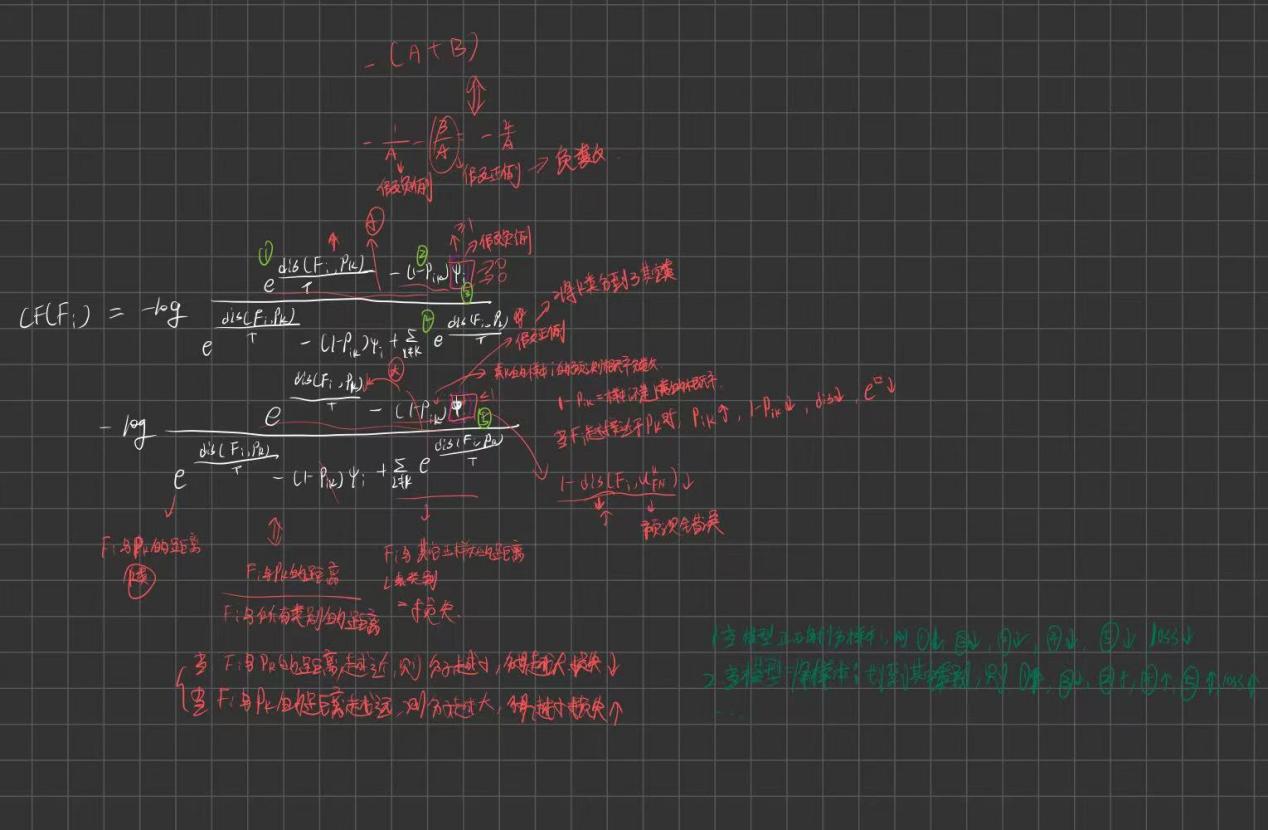
当没有FN样本或余弦距离收敛到1时，φi达到最小值0。这可能会激励模型将这些模糊样本纠正为动作k。



当没有FP样本或余弦距离收敛到-1时，ψi达到最小值0。这可能会阻止模型将这些模糊样本识别为动作k。

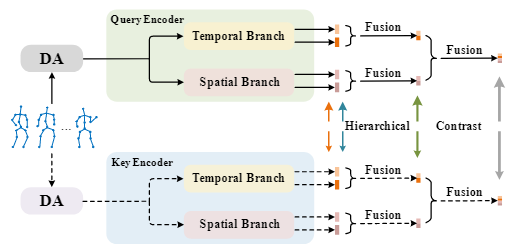


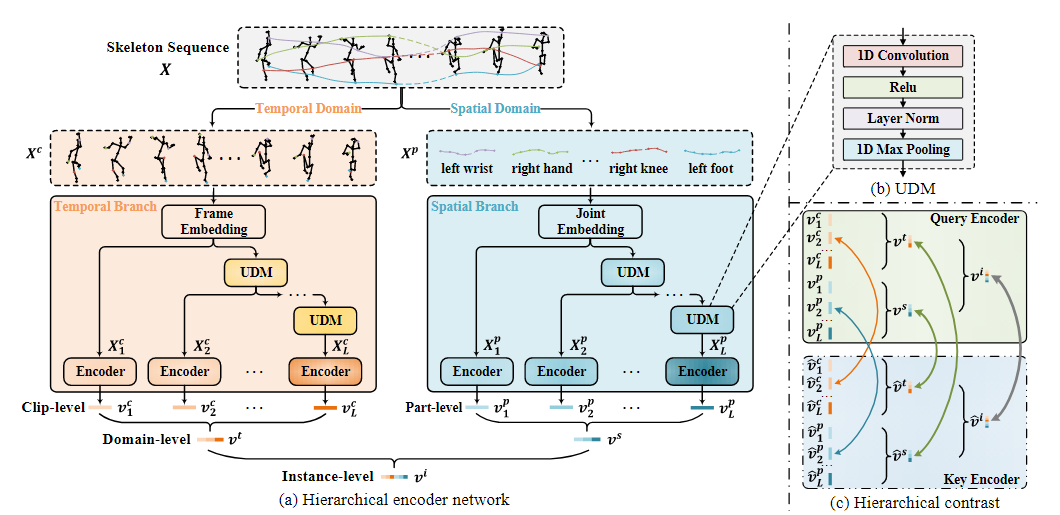
## 手写解析



# Hierarchical Contrast for Unsupervised Skeleton-Based Action Representation Learning(基于无监督骨架的动作表示学习的分层对比)

## 模型结构





## 如何得到X^c,X^p的？

通过把输入的骨架序列进行reshape操作得到的，时间维度上把TxJx3 -> Tx3J，空间维度上TxJx3 -> Jx3T。

## Encoder是什么？

这个全称为S2SEncoder，可以是GRU，LSTM，Transformer。这里好像在时间上与空间上是不一样的需要在代码里面去看看。

论文里写了时间上与空间上用的是不一样的，但是在代码中就是用三个中的一个。

## UDM是什么？

这个是论文的一大核心支撑点。由1D卷积（这里一维卷积层用于收集附近的上下文信息），Relu激活函数，Norm正则化函数，1D最大池化组成（最大池化用于信息聚合），并且可以在Branch中进行串联（这会导致更细粒度的剪辑）。

在时空维度上的作用是：它依次合并多个连续的帧/剪辑特征以获得更粗粒度的剪辑。

## 分层对比？

分成了4个层次：实例级对比，域级对比，剪辑级对比，部分级对比（其中剪辑级与部分级是一个级别的对比）

## 为什么Branch要这么设计，理由是什么，有什么好处？

## 为什么UDM要这么设计，理由是什么，有什么好处？

1D卷积可以聚合信息，然后激活，正则化来增强与稳定数据表达能力，最后最大池化来提取突出的特征信息。

### 那为什么不用2D要用1D?

## 为什么要在损失函数计算那里使用多级损失对比？

## 模型设计中如何体现无监督学习的？

我对无监督的学习是使用一批有标签的数据以及一批无标签的数据进行同时的训练。

对比学习是无监督学习的一种，通常是通过使用增强来产生一张图片，然后去让模型识别出这个增强的图片与源图片是同一类，这样也就实现了无监督学习。并且如果有10张图片去训练通过增强至少可以变成20张。

## 如何进行训练的？

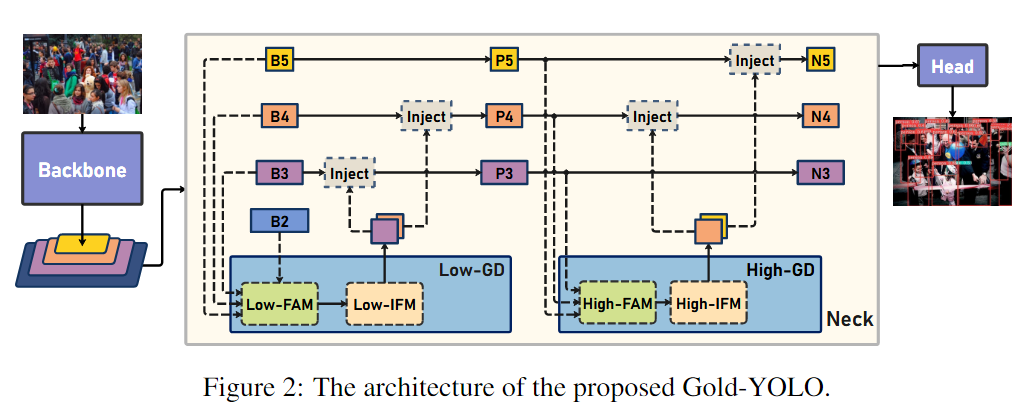
可以在上面模型图结构看出，对比学习这部分的模型只训练了一个无监督的学习，并没有把动作进行分类，在作者代码中是先训练这个无监督模型，让网络有足够的能力去区分不同动作类别，但是并没有进行最后的分类。所以还需要单独训练一个分类网络。

# Gold Yolo

由华为提出来的一个目标检测网络，在目标检测中一般把网络分为骨干，颈部，头部。骨干负责提出来主要特征，颈部负责特征的融合（其实一般这一步容易被忽视掉），头部负责进行把骨干网络提出来的特征计算到具体目标。

比如在骨架人体动作识别中，颈部相当于把特征拉成一维然后传到全连接层中进行动作分类；又或者在二分支网络中，时间与空间，计算出来各自的特征图，两者如何融合计算比较好，还是计算完再进行融合。

## Gold-Yolo架构图



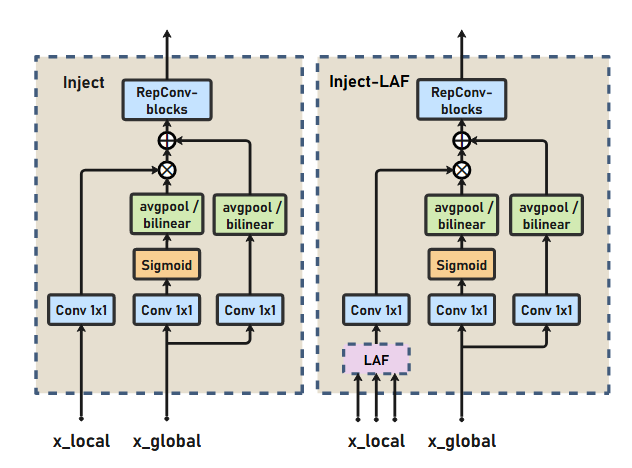
把特征分成了4份（这一步怎么分的？）

## 低级特征阶段

第一步：特征对齐，代码中是按照倒数第二个特征来对其，如果比它长则用平均池化，如何比它短则用线性插值。

第二步：特征融合，这里的特征融合在开始与结尾各使用一个1x1卷积来升维或降维，并在输出的时候添加一个split操作，中间插入一个RepConv-block模块，这个RepConv-block模块简单说是由Conv3x3（k=3,p=1,s=1）+Conv1x1（k=1,p=?,s=1）+id\_out组成。（这里为什么这样处理就是可以达到优秀的特征融合？）

第三步：插入，如图：



## 高级特征阶段

第一步：与低级阶段第一步一致。

第二步：这一步与低级阶段的第二步不同，由Transformer+残差网络组成。然后在最后加上一个split操作。

第三步：与低级阶段第三步一致。