# RTMPose姿态估计论文

**1.人体姿态估计常用方法**

1）坐标点回归（APA车位角点检测，人脸68关键点检测）

2）heatmap 2D热图回归（基于bottom-up的OpenPose）

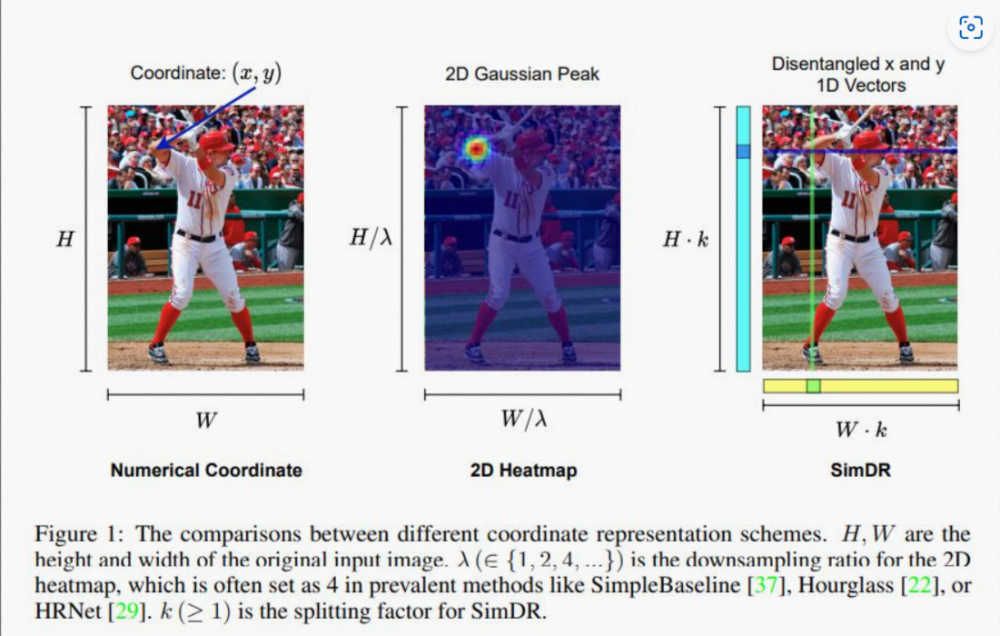
3）2022的论文，SimCC Head，通过把2D降维到1D上，分别对W和H进行分类，利用分类任务代替回归方法

**2.大名鼎鼎的SimCC建模**

很长一段时间内，都是heatmap 2D热图算法一直刷榜，精度优秀！

SimCC的挑战：

**对于姿态估计任务而言，2D高斯热图形式的特征表示真的是必要的吗？**



**HeatMap热图过去争议最大的点**

1. **在低分辨率图片上掉点严重**：对于HRNet-W48，当输入分辨率从256x256降到64x64，AP会从75.1掉到48.5

1. **为了提升精度，需要多个上采样层来将特征图分辨率由低向高进行恢复**：通常来说上采样会使用转置卷积来获得更好的性能，但相应的计算量也更大，骨干网络输出的特征图原本通道数就已经很高了，再上采样带来的开销是非常庞大的

1. **需要额外的后处理来减小尺度下降带来的量化误差**：如DARK修正高斯分布，用argmax获取平面上的极值点坐标等

综上所述，HeatMap并不适用于在低算力平台上落地！

**3.SimCC建模方法：**

1)降维策略，原本我们都是在2D上推理高斯热图，能否在1D完成输出？

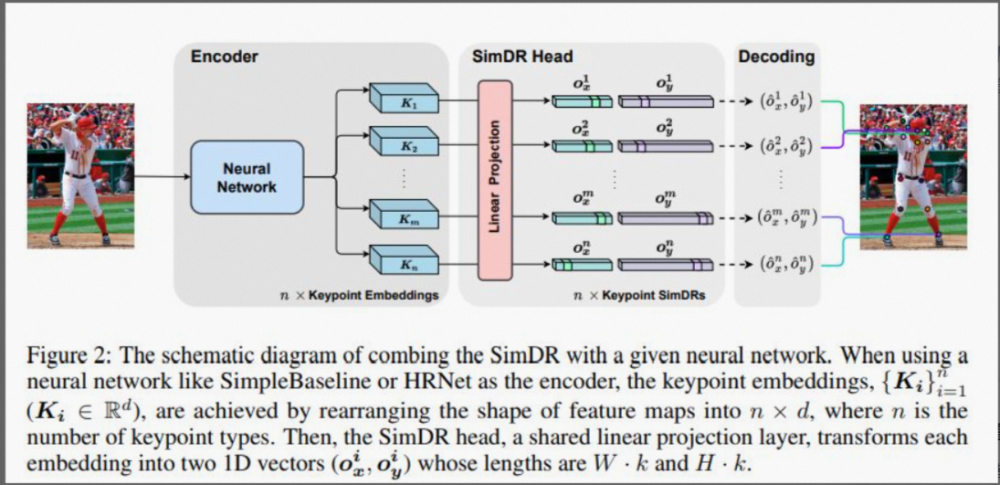
2）backbone输出（N,K,W,H），其中N表示N个框，K表示关键点类别数，W,H表示feature的宽高

3）将（N,K,W,H）flatten成（N,K,WxH），再通过简单的Linear全连接layer，变成（N,K,W\*k）和（N,K,H\*k）2个输出。

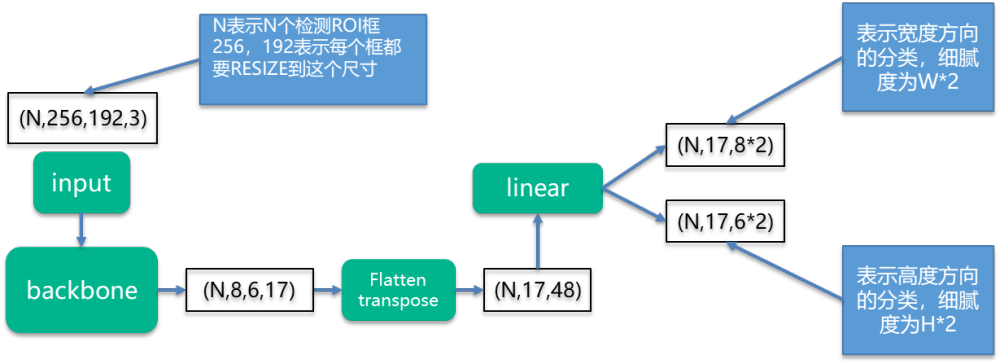
4）2个输出分别表示在宽度方向做宽度为W\*k个类别的分类，那么所处的位置如果概率较高，那么关键点就在该位置，同理H方向做一个类似的事情，这样子结合W，H两个1维方向的位置分类，就得到了关键点最终的位置。

5）为什么有k，因为W,H可能细腻度还不够，所以可以把其扩展为W和H的k倍数，在论文里叫做bins，即亚像素的意思，这样子可以得到更高精度的关键点位置。

6）利用这个方法，我们非常简单有效的把一个回归任务或者heatmap任务转换成了2个一维的分类任务。



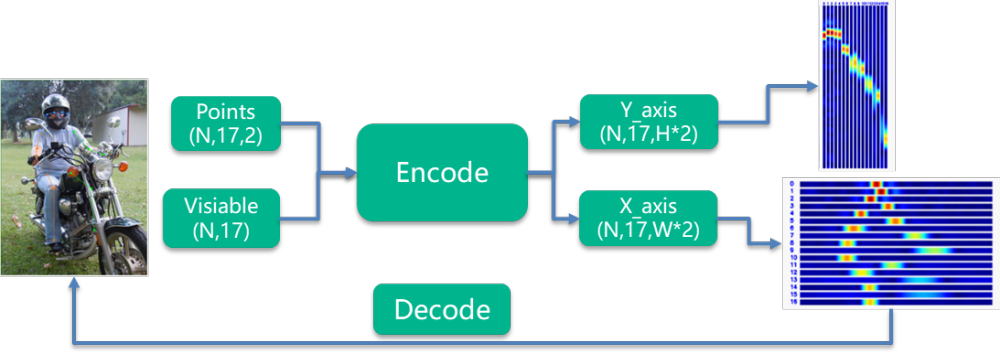
7）上图结合下图一起来观察整个mmpose的关键点估计流程。



**3.SimCC的Encode和Decode**

1)模型的训练和GT的制作需要Encode

2)模型的推理需要Decode



#Encode kernel code  
#将GT值进行编码后，可以和模型推理值进行损失函数计算  
W = W\*2, H=H\*2  
keypoints = [N,K,2]  
keypoints\_split = keypoints \* 2  
x = np.arange(0, W, 1, dtype=np.float32)  
y = np.arange(0, H, 1, dtype=np.float32)  
target\_x = np.zeros((N, K, W), dtype=np.float32)  
target\_y = np.zeros((N, K, H), dtype=np.float32)  
  
for n, k in product(range(N), range(K)):  
 mu = keypoints\_split[n, k]  
 mu\_x, mu\_y = mu  
 target\_x[n, k] = np.exp(-((x - mu\_x)\*\*2) / (2 \* self.sigma[0]\*\*2))  
 target\_y[n, k] = np.exp(-((y - mu\_y)\*\*2) / (2 \* self.sigma[1]\*\*2))  
  
  
#Decode kernel code, only used for inference  
#模型推理时，对输出做进一步DECODE，得到关键点的坐标  
N=1  
simcc\_x = [N,K,384] = [K,384]  
simcc\_y = [N,K,512] = [K,512]  
x\_locs = np.argmax(simcc\_x, axis=1) #[K]  
y\_locs = np.argmax(simcc\_y, axis=1) #[K]  
locs = np.stack((x\_locs, y\_locs), axis=-1).astype(np.float32) #[K, 2]  
max\_val\_x = np.amax(simcc\_x, axis=1) #[K]  
max\_val\_y = np.amax(simcc\_y, axis=1) #[K]  
  
#取最小概率值作为输出  
mask = max\_val\_x > max\_val\_y  
max\_val\_x[mask] = max\_val\_y[mask]  
vals = max\_val\_x  
  
#每个像素有2个bins，从亚像素还原为像素级别  
locs /= 2.0

**4.SimCC Head损失函数**

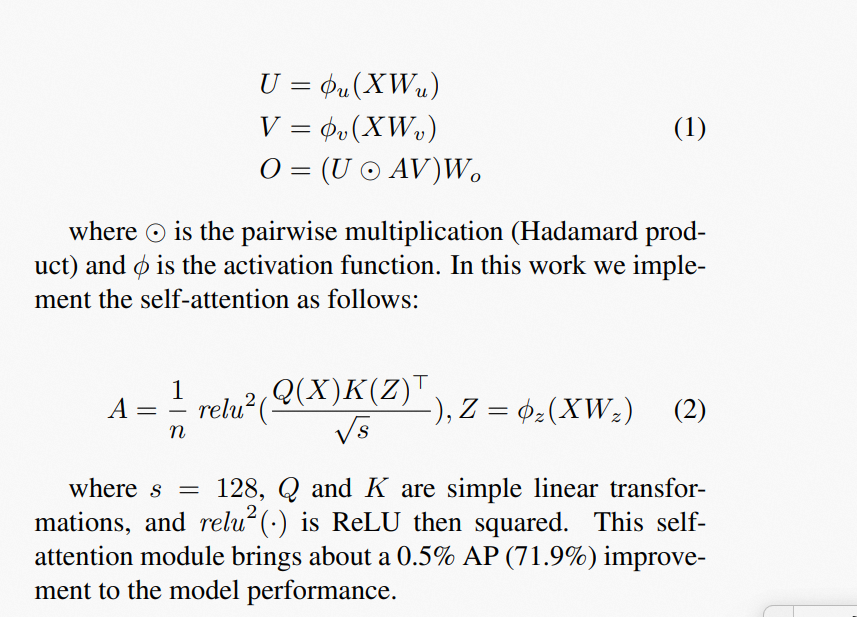
KL离散度损失

**5.RTmpose的注意力机制**

**如何表示一个手势上不同关键点之间的关系？**

**GAU，门控注意力机制模块**

**它也是从基础的Transformer算子变换而来。**



GAU门控注意力代码流程图

