DARKPose Distribution-Aware Coordinate Representation for Human Pose Estimation笔记

- Paper: Distribution-Aware Coordinate Representation for Human Pose Estimation
- Code: ilovepose/DarkPose

0. Summary Keywords

- **分布感知坐标解码**，在**亚像素**级别进行精确定位
- **无偏坐标编码**；

1. Introduction

1.1 Why

因整数导致回归目标有偏，对网络效果有较大影响。

- **坐标解码**：当前的坐标解码方法的设计存在缺陷，应用时需将关节点热图转换回关节点坐标，这会导致得到的关节点是整数，与真实标注产生误差。同时因为计算量的关系，**热图尺寸通常会比输入图片缩小n倍，分辨率下采样期间会引入量化误差**，因此坐标解码会将误差放大。在本文提出之前，使用的方法是取最高峰位置m和第二高峰位置s（因为网络生成的热图会有多峰如下图），输出位置p=m+0.25(s-m)，即将第二高峰位置作为小数补充。
- **坐标编码**：训练时需将关节点坐标转换为关节点热图，**会导致带小数坐标被转换到临近的整数位置**；因为将高分辨率图片上的坐标投影到热图坐标空间上，再用量化后的投影坐标生成热图，分辨率是降低的，因此热图生成过程中也会引入量化误差。

1.2 What

主要在坐标解码和坐标编码两方面做了改进：

- **坐标解码**：提出了一种更遵循本质的**分布感知坐标解码**方法；
  - **热图分布调整**；
  - 通过泰勒展开实现**基于分布感知的最大激活点重定位，可以达到亚像素级别的精度**；
  - 通过**分辨率恢复**将热图上的关键点坐标投影到原始图片的坐标空间上；

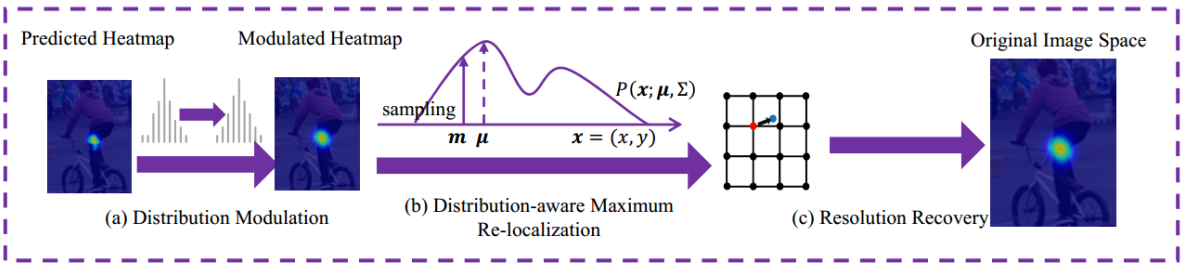

Figure 2: Overview of the proposed distribution aware coordinate decoding method.

- **坐标编码**：提出了一种更精确的热图分布生成方法，用于无偏模型训练；
  - 用没有经过量化的精确的热图投影坐标代替经过了量化的有偏差的热图投影坐标，生成更精确的热图；

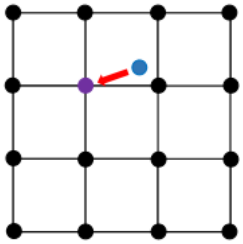

Figure 4: Illustration of quantisation error in the standard coordinate encoding process. The blue point denotes the accurate position (g') of a joint. With the *floor* based coordinate quantisation, an error (indicated by red arrow) is introduced. Other quantisation methods share the same problem.

1.3 How

- **坐标解码**：通过泰勒展开实现**基于分布感知的最大激活点重定位，可以达到亚像素级别的精度**；
- **坐标编码**：用**没有经过量化的精确的热图投影坐标**代替经过了量化的有偏差的热图投影坐标，生成更精确的热图；

1.4 Contributions

- 通过一种**估计**方法，从热图中获得更精确的坐标，具体步骤如下：
  1. 对热图使用高斯核平滑（参数与训练时使用的高斯核相同）
  2. 使用本文的分布感知方法估计坐标位置
  3. 分辨率恢复

2 Method

2.1 坐标解码

从热图中获得更精确的坐标，具体步骤如下：

- **热图分布调整**：对热图使用**高斯核平滑**（参数与训练时使用的高斯核相同）
  - 对热图使用**高斯核平滑**（参数与训练时使用的高斯核相同）

$$\boldsymbol{h}' = K \circledast \boldsymbol{h} \tag{10}$$

  - 保留原始热图的幅度

$$\boldsymbol{h}' = \frac{\boldsymbol{h}' - \min(\boldsymbol{h}')}{\max(\boldsymbol{h}') - \min(\boldsymbol{h}')} * \max(\boldsymbol{h}) \tag{11}$$

- **分布感知解码**：
  - 假设预测的热图服从二维高斯分布，和ground-truth热图一样。x是预测的热图中的任意像素点坐标，u预测的关键点；

$$\mathcal{G}(\boldsymbol{x}; \boldsymbol{\mu}, \Sigma) = \frac{1}{(2\pi)|\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\boldsymbol{x} - \boldsymbol{\mu})^T \Sigma^{-1} (\boldsymbol{x} - \boldsymbol{\mu})\right) \tag{3}$$

$$\Sigma = \begin{bmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{bmatrix} \tag{4}$$

$$\mathcal{P}(\boldsymbol{x};\boldsymbol{\mu},\Sigma)=\ln(\mathcal{G})=-\ln(2\pi)-\frac{1}{2}\ln(|\Sigma|) \\ -\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu}) \tag{5}$$

- 我们的目标是估计u，u是实际的最大激活点；

$$\mathcal{D}'(\boldsymbol{x})\Big|_{\boldsymbol{x}=\boldsymbol{\mu}}=\frac{\partial\mathcal{P}^T}{\partial\boldsymbol{x}}\Big|_{\boldsymbol{x}=\boldsymbol{\mu}}=-\Sigma^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\Big|_{\boldsymbol{x}=\boldsymbol{\mu}}=0 \tag{6}$$

- 泰勒展开P：

$$\mathcal{P}(\boldsymbol{\mu})=\mathcal{P}(\boldsymbol{m})+\mathcal{D}'(\boldsymbol{m})(\boldsymbol{\mu}-\boldsymbol{m})+\frac{1}{2}(\boldsymbol{\mu}-\boldsymbol{m})^T\mathcal{D}''(\boldsymbol{m})(\boldsymbol{\mu}-\boldsymbol{m}) \tag{7}$$

$$\mathcal{D}''(\boldsymbol{m})=\mathcal{D}''(\boldsymbol{x})\Big|_{\boldsymbol{x}=\boldsymbol{m}}=-\Sigma^{-1} \tag{8}$$

- 用下面的公式估计u，m是预测的最大激活点：

$$\boldsymbol{\mu}=\boldsymbol{m}-\big(\mathcal{D}''(\boldsymbol{m})\big)^{-1}\mathcal{D}'(\boldsymbol{m}) \tag{9}$$

- **坐标还原**：将热图上的坐标u投影到原图坐标空间

$$\hat{\boldsymbol{p}}=\lambda\boldsymbol{p} \tag{2}$$

2.2 坐标编码

- 在训练过程中用ground-truth坐标生成热图的时候，用未量化的坐标g'代替量化的坐标g"

$$\mathcal{G}(x,y;\boldsymbol{g}'')=\frac{1}{2\pi\sigma^2}\exp\left(-\frac{(x-u'')^2+(y-v'')^2}{2\sigma^2}\right) \tag{14}$$

$$\boldsymbol{g}''=(u'',v'')=\text{quantise}(\boldsymbol{g}')=\text{quantise}(\frac{u}{\lambda},\frac{v}{\lambda}) \tag{13}$$

$$\boldsymbol{g}'=(u',v')=\frac{\boldsymbol{g}}{\lambda}=(\frac{u}{\lambda},\frac{v}{\lambda}) \tag{12}$$

3. code=f(method)

2.1 坐标解码优化模块

- 更遵循本质的分布感知坐标解码
  - 热图分布调整：用高斯核进行卷积，来平滑热图中的多峰
  - 基于泰勒展开的分布感知关键点坐标定位

```
# lib/core/inference.py中
def gaussian_blur(hm, kernel):
```

```
    border = (kernel - 1) // 2
    batch_size = hm.shape[0]
    num_joints = hm.shape[1]
    height = hm.shape[2]
    width = hm.shape[3]
    for i in range(batch_size):
        for j in range(num_joints):
            origin_max = np.max(hm[i,j])
            dr = np.zeros((height + 2 * border, width + 2 * border))
            dr[border: -border, border: -border] = hm[i,j].copy()
            dr = cv2.GaussianBlur(dr, (kernel, kernel), 0)
            hm[i,j] = dr[border: -border, border: -border].copy()
            hm[i,j] *= origin_max / np.max(hm[i,j])
    return hm
```

```
# lib/core/inference.py中
def taylor(hm, coord):
    heatmap_height = hm.shape[0]
    heatmap_width = hm.shape[1]
    px = int(coord[0])
    py = int(coord[1])
    if 1 < px < heatmap_width-2 and 1 < py < heatmap_height-2:
        dx  = 0.5 * (hm[py][px+1] - hm[py][px-1])
        dy  = 0.5 * (hm[py+1][px] - hm[py-1][px])
        dxx = 0.25 * (hm[py][px+2] - 2 * hm[py][px] + hm[py][px-2])
        dxy = 0.25 * (hm[py+1][px+1] - hm[py-1][px+1] - hm[py+1][px-1] \
            + hm[py-1][px-1])
        dyy = 0.25 * (hm[py+2][px] - 2 * hm[py][px] + hm[py-2][px])
        derivative = np.matrix([[dx],[dy]])
        hessian = np.matrix([[dxx,dxy],[dxy,dyy]])
        if dxx * dyy - dxy ** 2 != 0:
            hessianinv = hessian.I
            offset = -hessianinv * derivative
            offset = np.squeeze(np.array(offset.T), axis=0)
            coord += offset
    return coord
```

2.2 坐标编码优化模块

- 用没有经过量化的精确的热图投影坐标代替量化后的有偏差的热图投影坐标，生成更精确的热图
    - 用仿射变换将关键点坐标投影到热图坐标空间上
    - 用这个没有经过量化的投影坐标生成更精确的热图

```
# lib/dataset/JointDataset.py中的def __getitem__(self, idx):函数中
        joints_heatmap = joints.copy()
        # 进行仿射变换，样本数据关键点发生角度旋转之后，每个像素也旋转到对应位置
        # 获得旋转矩阵
        trans = get_affine_transform(c, s, r, self.image_size)
        trans_heatmap = get_affine_transform(c, s, r, self.heatmap_size)

        # 对人体关键点也进行仿射变换
        for i in range(self.num_joints):
            if joints_vis[i, 0] > 0.0:
                joints[i, 0:2] = affine_transform(joints[i, 0:2], trans)
```

```
                joints_heatmap[i, 0:2] = affine_transform(joints_heatmap[i,
0:2], trans_heatmap)

        # 获得ground truch，热图target[17, 64, 48]，target_weight[17, 1]
        target, target_weight = self.generate_target(joints_heatmap, joints_vis)
```

```
# lib/dataset/JointDataset.py中的def generate_target(self, joints, joints_vis):函
数中
            # 为每个关键点生成热图target以及对应的热图权重target_weight
            for joint_id in range(self.num_joints):
                target_weight[joint_id] = \
                    self.adjust_target_weight(joints[joint_id],
target_weight[joint_id], tmp_size)

                if target_weight[joint_id] == 0:
                    continue

                mu_x = joints[joint_id][0]
                mu_y = joints[joint_id][1]

                x = np.arange(0, self.heatmap_size[0], 1, np.float32)
                y = np.arange(0, self.heatmap_size[1], 1, np.float32)
                y = y[:, np.newaxis]

                v = target_weight[joint_id]
                if v > 0.5:
                    target[joint_id] = np.exp(- ((x - mu_x) ** 2 + (y - mu_y) **
2) / (2 * self.sigma ** 2))
```

4. Heatmap

heatmap有如下的一些优点：

1. 可以让网络全卷积，因为输出就是2维图像，不需要全连接。fully connected layers are prone to overfitting, thus hampering the generalization ability of the overall network。而坐标回归的方法缺少了空间和上下文信息，由于关键点定位存在固有的视觉模拟两可的特征，这就给坐标回归造成了很大的挑战。
2. 关节点之间（头和胸口，脖子和左右肩膀）是有很强的相关关系的。然而单独的对每一类关节点回归坐标值并不能捕捉利用这些相关关系，相反当回归heatmap时，一张输入图像对应的heatmap就存在这种相关关系，那就可以用来指导网络进行学习。简言之，头关节的回归可以帮助胸口关节，脖子关节的回归也可以帮助左右肩膀，反之亦然。
3. heatmap同时捕捉了前景（关节点）与背景的对比关系，可以用来指导网络进行学习。

5. 参考资料

1. Distribution-Aware Coordinate Representation for Human Pose Estimation 姿态估计 CVPR2019
2. [小结]Distribution-Aware Coordinate Representation for Human Pose Estimation
3. 寻找通用表征：CVPR 2020上重要的三种解决方案

问雪更新于2021-07-09