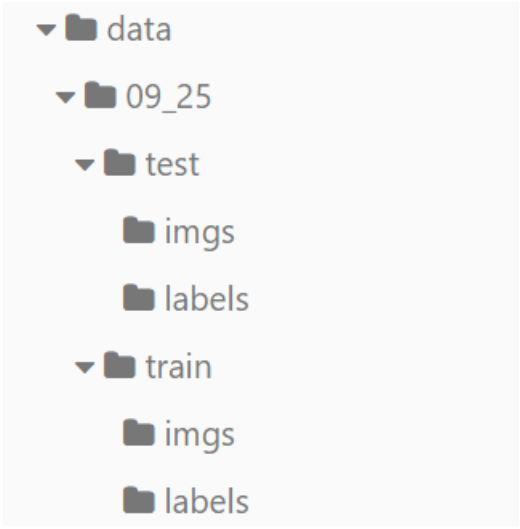
代码说明文档

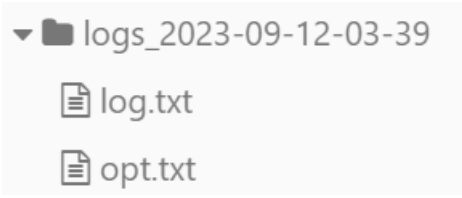
1 文档整体架构

1.1 data



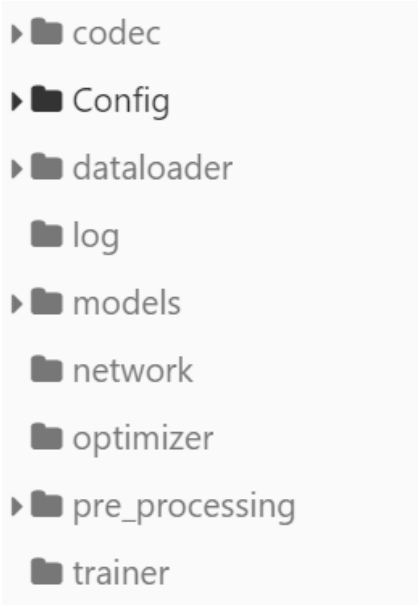
存放训练和测试图片文件夹，不同的训练集和测试集以日期命名。imgs目录存放图片，labels目录存放labelme标注的json文件。

1.2 exp



输出文件夹，用于储存本次训练过程中的配置文件opt.txt和输出日志log,txt。

1.3 lib



主要的文件夹。项目配置文件（Config）、数据加载（dataloader）、日志输出配置（log）、模型结构（stem、backbone、head等）、半监督学习网络（network）、优化器配置（optimizer）、数据预处理（pre_processing）、集成后的训练器（trainer）、其他文件（utils）。

1.4 output

用于调试和理解网络输出，得到网络特征提取部分的图片。

1.5 results

储存网络推理后输出的图片。

1.6 tools

用于网络部署，pytorch转onnx和mnn。

2. 代码和网络架构

2.1 数据预处理

2.1.1 json关键点文件转为numpy坐标

`lib.pre_processing.data_pre_loc.py`

```
def json_to_numpy(dataset_path):
    """
    input: json标签的文件路径
    output:转换为numpy格式(shape:[num_keypoints,2]
    return landmarks(K,2),K为关键点的个数
    """
```

2.1.2 heatmap数据前处理和后处理

`lib.codec.msra_heatmap.py`

```
class MSRAHeatmap:
    """
    用于类外调用的只有两个函数：encoder和decode分别用于将关键点编码为热图和热图解码为关键点
    Args:
        input_size(W,H):图像本身的大小，在这个项目中是(1280,704)
        heatmap_size(w,h):热力图大小，W/w=4;H/h=4
        sigma(float):超参数，用于高斯分布的平缓性控制
        blur_kernel_size(int):
    """
    def _genetrate_unbiased_gaussian_heatmaps(heatmap_size,keypoints,sigma):
        """
        输入关键点坐标，热力图大小和高斯分布超参数返回产生的热力图
        Args:
            heatmap_size(w,h)
            keypoints(K,2):K为关键点的个数，也就是num_keypoints
            sigma:hyperparameter
        return:
            heatmap(B,w,h,C):C=num_keypoints
        """
     def encoder(keypoints):
```

```
        """
        输入关键点坐标，通过调用_genetrate_unbiased_gaussian_heatmaps函数得到对应的热力图
        Args:
            keypoints(K,2)
        return:
            heatmap(B,w,h,C)
        """
    def _get_heatmap_maximum(heatmaps):
        """
        得到热力图的16个通道中每个通道的关键点相应最大的位置和相应的分数，用于最后的解码
        Args:
            heatmap(B,w,h,C)
        return:
            locs(B,K,2):关键点的位置，B为batch_size
            vals(B,K):关键点对应的分数def
        """
    def _gaussian_blur(heatmaps,blur_kernel_size):
        """
        对高斯热图进行高斯平滑
        Args:
            heatmap(B,w,h,C)
            blur_kernel_size(int=11)
        return:
            heatmap_after_blur(B,w,h,C)
        """
     def _refine_keypoints_dark(keypoints,heatmaps,blur_kernel_size):
        """
        利用dark pose的trick对下采样后的关键点的量化误差进行修正，返回修正后的关键点
        论文地址：https://arxiv.org/abs/1910.06278
        Args:
            keypoints(K,2)
            heatmaps(B,w,h,C)
            blur_kernel_size(float)
        return:
            refined_keypoints(K,2)
        """
     def decode(encoded_heatmaps):
        """
        通过调用_get_heatmap_maximum和_refine_keypoints_dark对heatmap进行解码得到关键点坐标
        Args:
            encoded_heatmaps(B,w,h,C):model(img)['heatmap'],推理预测过程得到的热力图
        return:
            keypoints(K,2):最终解码到原图分辨率的关键点坐标
            scores:关键点的得分置信度
        """
```

2.1.3 dataloader

`lib.dataloader.transforms.py`

- 自定义图片和对应关键点畸变的数据增强

```
def distort(input_image,keypoints,distortion_probability):
    """
    用于数据增强模拟图像畸变的情况
    Args:
        input_image(PIL):输入图片
        keypoints(np.ndarray):图片对应关键点
        distortion_probability(float)：畸变概率
    return:
        undistorted_image(PIL):模拟畸变后的图片
        distorted_keypoints(np.ndarray):畸变后的图片对应的关键点
    """
```

- 数据增强类，将imgaug数据增强序列封装到类中，直接得到处理后的图像和点

```
class ImageAugmentation
    """
    DataLoader: 数据增强类
    Args:
        ia_sequence:用imgaug库进行的图像和对应关键点的增强变换
        num_joints(int):关键点数量
    return:
        img_aug:增强后的图片
        keypoints_aug:对应图片增强后的关键点
    """
```

`cardata.py`

```
# 利用imgaug库获得数据增强序列
seq = iaa.Sequential([
                    iaa.Fliplr(0.5).
                    iaa.Numtiply((0.1,1.9)),
                    ......
])
```

- 全监督情况下的dataloader

```
class Label_Dataset(torch.utils.data.Dataset):
    """
    获得需要的数据
    Args:
        dataset_path:训练图片路径
        config:配置文件路径
    return:
        items:需要的dataloader列表
            items['img']:原始图片
            items['keypoints']:缩放后的关键点坐标
            items['heatmaps']:缩放后的关键点的热力图
            items['img_name']:原始图片对应的图片名称，用于调试
    """
```

2.2 网络架构

2.2.1 stem

用于对输入backbone的图片进行缩放，从而减慢推理速度。

```
class CompressingNetword(nn.Module):
    """
    input_feature:(C,H,W,3)
    Args:
        num_residual_blocks:残差块的数量，影响输出的图片分辨率。如输入1，分辨率/4；输入2，分辨率/8。
        num_filters:输出通道的数量
    out_feature:(C,H/n,W/n,num_filters),其中n为缩小的分辨率倍数
    """
```

2.2.2 backbone

```
class U_net(nn.Module):
    """
    input_feature:(C,H/n,W/n,num_filters)
    Args:
        in_channel:输入的通道数目，必须等于stem输出的num_filters
        feature_scale:决定Unet中间层的通道数目，一般取1和2。数字越小，网络参数越少结构越简单
    output_feature:(C,H/n,W/n,out_filter_num),只测试了16通道输出还没有测试输出其他数值的结果
    """
```

2.2.3 head

承接backbone输出的特征图，用head连接最后得到的缩放后的目标热力图，每个通道表示每个关键点的热力图。

```
class Unet_Head(nn.Module):
    """
    input_feature:(C,H/n,W/n,in_filter_num),输入特征图大小等于backbone层输出的大小。
    Args:
        num_keypoints:关键点的数量，在这个项目中是确定的16
    output_feature:(C,H/n,W/n,num_keypoints)
    """
```

2.2.4 detector

对`stem`、`backbone`、`head`做连接和整合。

```
class Unet_Detector(nn.Module):
    def forward(self, x):
        """
        训练及测试的前向推理过程
        """
        x = self.setm(x)
        x = self.backbone(x)
        x = self.header(x)
        return x
    def predict(self, items):
        """
        测试集预测过程
        Args:
            items:model(img)输出的数据集加载器
        return:
            batch_kayepoints(num_keypoints,2):每个batch下预测的keypoints
            batch_score(num_keypoints,2):keypoints对应的score，用于阈值筛选
        """
    def show_point_on_picture(self,_img,_pre):
        """
        将预测的关键点画在图上
        Args:
            _img:待预测图片
            _pre:已经好的关键点
        return:
            img:将预测点画在原图上的新图
        """
```

2.2.5 loss

`BoneLoss`计算公式：

$$E_{\text{bone}}(\beta,\omega,T,s)=\sum_{(i,j\in\varepsilon)}\left|\left\|J_{2D_j}-J_{2D_i}\right\|-\left\|Y_{2D_j}-Y_{2D_i}\right\|\right|$$

其中J_{2D}是模型预测的关键点，Y_{2D}是GroundTruth，该公式约束了每个关键点之间的空间关系，同时相当于约束了每个关键点之间的长度，避免预测出太偏移的点。

```
class BoneLoss(nn.Module):
    """
    Args:
        target(B,H,W,C):图片中关键点的热力图，其中H和W是缩放后的大小，C=16也就是关键点个数
        output(B,H,W,C):model(img)['heatmap']前向推理预测出的热力图
        loss_weight(float):loss所占权重
    return:
        loss:计算出的boneloss值
    """
```

`MSELoss`：计算预测热力图和实际热力图的MSE值

`AdaptiveWingLoss`：在处理关键点位置不确定性和噪声方面表现出色，可以提高关键点检测的性能。

```
class AdaptiveWingLoss(nn.Module):
    """
    Args:
        alpha(float),omega(float),epsilon(float),theta(float):超参数，论文指定
    return:
        loss:计算出的adaptivewingloss值
    """
```

2.3 optimizer

`lr_scheduler.py`

```
class LinearWarmupCosineAnnealingLR(torch.optim.lr_scheduler._LEScheduler):
    """
    余弦退火学习率
    """
```

`optimizer.py`：优化器的封装，可以选择`Adam`或`AdamW`

2.4 log

```
class Logger(object):
    """
    将训练日志以及本次训练的配置文件保存
    Args:
        config：配置文件
        time_str：保存日志的时间，自动获取当前时间
    """
    def write(self, txt):
        """
        将训练loss，lr等写入log.txt
        """
     def scalar_summary(self, tag, value, step):
        """
        将log.txt的内容同步写入tensorboard
        """
```

2.5 trainer

`fully_supervised_trainer.py`：全监督训练器

```
class FullySupervisedTraner:
    """
    Args:
        config: 配置文件
    """
    def train_epoch(self, epoch):
        """
        遍历每一个batch：清零梯度->获取每一个batch数据，并移到设备上->前向传播->计算损失并更新->反向传播->裁          剪梯度->更新模型参数->更新学习率
        Args:
            epoch:当前训练的轮数
        return:
            result：一个字典，包括surpervised_loss、bone_loss和total_loss
        """
    def test_epoch(self):
        """
        读取图像->前向传播获得热图预测->热图解码为关键点->将关键点可视在图像上->保存到指定路径
        """
    def train(self, resume, pre):
        """
        获取当前时间戳->根据train_epoch计算每个epoch的loss->根据loss历史数据选择是否保存当前参数
        Args:
            resume(bool):选择是否从之前中断的训练数据中恢复训练
            pre(bool):是否加载预训练模型
        """
     def test(self, weight_path):
        """
        load权重文件->调用test_epoch函数进行推理
        Args:
            weight_path:权重文件的路径
        """
```

2.6 其他函数

1. `lib.utils`

```
class AverafeMeter(object):
"""计算和储存平均值以及现在的值，用于train时loss等值的更新"""
class InfiniteDataLoader(torch.utils.dataDataLoader):
"""无限数据加载类"""
    def __next__(self):
        """创建next(dataloader)方法，获得每个batch下的数据"""
```

2. `lib.pre_processing.utils`

- `labeltococo.py`

  将labelme格式的关键点标注文件转换为coco格式

- `mean.py`

计算整个数据集图片的std和mean(可用于img输入图片的标准化，但是测试过有无标准化结果没有太大差别)

2.7 tools

1. `pytorch2onnx.py`

   将pytorch模型转换成onnx样式输出

2. `mnn_inference.py`

   将pytorch模型转换成mnn样式输出(可选)

3. `onnx_inference.py`

   对简化后的onnx后缀的文件新型推理，得出模型在cpu和gpu上的推理速度

2.8 config

`lib.Config.config.py`：见代码注释

3. 尝试的方法

3.1 半监督学习

介绍与全监督方法中的不同之处

- `lib.trainer.semi_surpervised_trainer.py`

```
class EMATrainer:
    """
    FixMatch:https://arxiv.org/abs/2001.07685
    Args:
        config:配置文件
    Diff from surpervised trainer:
        unlabeled_batch_heatmap_pre_w:测试集中的弱增强后的热图
        unlabeled_batch_heatmap_pre_s:测试集中的强增强后的热图
    loss在全监督的基础下加入unlabeled_batch_heatmap_pre_w和
unlabeled_batch_heatmap_pre_s的差      别。可以选择MSE做loss的量化方法，也就是
trian_epoch中的loss_u
    """
```

- `lib.dataloader.transforms.py`

FixMatch中封装两种不同的数据增强的类

```
class TransformFixMatch(object):
    """
    Args:
        x: 输入的没有增强的图片
        choose:选择'weak'或者'strong'
    return:
        strong(x) or weak(x):返回强增强或者弱增强的图片，用户选择
    """
```

- `lib.dataloader.cardata.py`

增加没有标签的测试集的dataloader

```
class Unlabel_Dataset(torch.utils.data.Dataset):
    """
    获得需要的数据,比之前的lable_Dataset少了两个返回值，因为没有标签，所以无法获得关键点坐标
    Args:
        dataset_path:训练图片路径
        config:配置文件路径
    return:
        items:需要的dataloader列表
            items['img']:原始图片
            items['img_name']:原始图片对应的图片名称，用于调试
    """
```

3.2 迁移学习

介绍与全监督方法中的不同之处，主要使用对抗迁移学习：DANN

- `lib.utils.mlp.py`

```
class MLP(nn.Module):
    """
    Args:
        in_feature(int):在backbone阶段提取的中间特征展开后的大小
        hidden_layer(int,int):隐藏层，长度一般等于in_feature
        activatetion(char):'relu'或者'leaky_relu'
        bn(bool):选择是否使用batch_normalization
        dropout(float):drop的比例
    return:
        mlp(x):返回变换后的x
    """
```

- `lib.utils.reverse_grad.py`

```
class ReverseGrad(nn.Module):
    """
    DANN中的对抗分支中的梯度反转层，不对输入进行任何操作，但在反向传播中将梯度的符号取反
    Args:
        grad_scaling(float):反向传播系数,这里指定为1
        x:输入的特征图
    return(float):
        -grad_scaling*grad_output(表示当前操作层对于上一层梯度的贡献)
    """
```

- `lib.trainer.transfer_trainer.py`

```
class Transfer_trainer:
    """
    DANN项目地址:https://github.com/jindongwang/transferlearning
    Diff from surpervised trainer:
        在train_epoch增加domain_loss
    """
```