附件4

AI算法挑战活动案例

“五禽戏”是中国传统导引养生的重要功法，由东汉末年的名医华佗创编，是国家级非物质文化遗产项目。“五禽戏”结合人体脏腑、经络和气血的功能，仿效了虎、鹿、熊、猿和鸟五种动物的动作，发展至今形成了多个不同的流派。“AI 算法挑战”活动以“‘五禽戏’动作识别”为主题，引导学生通过改进数据处理、特征工程、模型训练等方法提高动作识别精度，探索开发能够指导“五禽戏”练习的虚拟现实应用、智能健身教练系统等作品，同时感受“五禽戏”这一传统健身方法的独特之处，体会中华传统文化的魅力。

学生需参考学习平台提供的基础数据集和基础代码，使用计算机视觉和机器学习等人工智能算法识别五禽戏的动作。（数据集描述见附录 4-1）

为便于学生开展创新实践，活动提供了一千余条视频（训练集）支持学生训练 AI 模型，以有效识别视频画面动作。学生可参考浦育平台会陆续发布的 2024 年 AI 算法挑战系列教程、课程和示例项目完成挑战（相关教程见附录 4-2、附录 4-3、附录 4-4）。学生可从浦育平台 AI 体验中的图像分类、语音分类入手，理解 AI 分类任务。此外，学生也可参考“八段锦姿态分类”项目，使用关键点检测技术提取人体关键点的坐标数据训练模型，熟悉同类任务的技术路线。

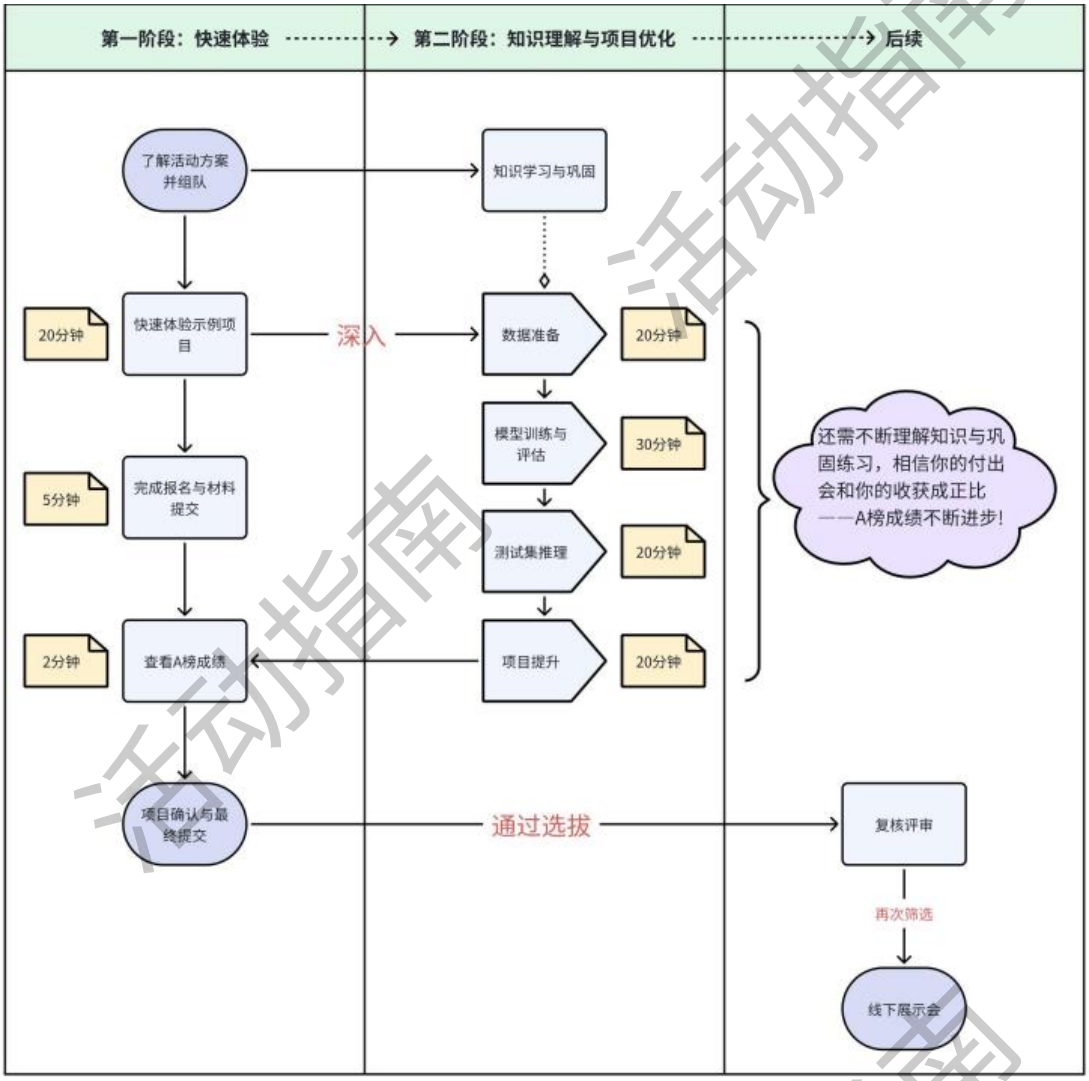


图1 活动进度安排

附录 4-1

活动数据集

1. 训练集

训练集提供 1000 余条已标注类别的视频数据，作为模型训练的基础数据。训练集包含 10 个类别的“五禽戏”动作，分类存放于相应名称的文件夹中。学生可自行采集一定的新数据对基础数据进行补充，从而提高模型训练的准确度。训练集的类别如下：

类别序号	类别标签	对应中文
0	0_TigerLift	虎举
1	1_TigerPounce	虎扑
2	2_DeerButt	鹿抵
3	3_DeerGallop	鹿奔
4	4_BearSwing	熊运
5	5_BearWave	熊晃
6	6_MonkeyLift	猿提
7	7_MonkeyPick	猿摘
8	8_BirdStretch	鸟伸
9	9_BirdFly	鸟飞

2. 测试集

测试集分为测试集 A 与测试集 B，其中测试集 A 包含 500 条未标注的视频数据，供学生用于报名期间的算法学习和测试；测试集 B 包含活动学生提交的数据，数量待定，不公开发布。

数据集的地址：

训练集	https://openinnolab.org.cn/pjlab/dataset/662e0cb34e6d836f0f099f80
测试集 A 集	https://openinnolab.org.cn/pjlab/dataset/662f3fe14e6d836f0f1ee1cc

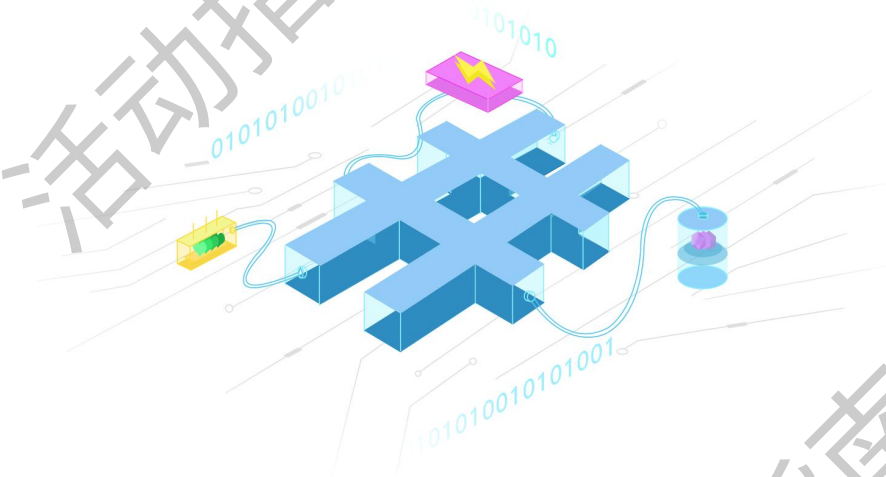
附录 4-2

“‘五禽戏’动作识别”技术分析

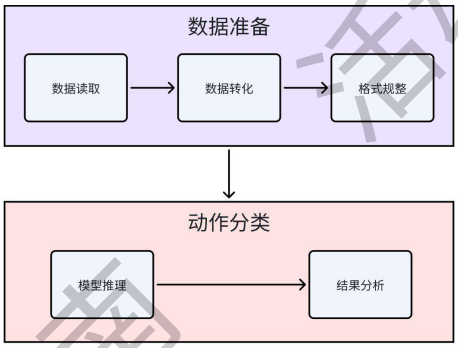
兵马未动，粮草先行。本篇文档将带你梳理“‘五禽戏’动作识别”的相关技术路线，并了解如何为 AI 模型训练准备合适的数据。

1. 动作识别的流程

实现动作识别需要经历两个过程，分别是“数据准备”和“动作分类”。首先需要对五禽戏的视频文件进行预处理，使得视频数据经历复杂的管线关系网络处理过程（如下图所示），以符合“动作分类”的要求，然后再进行动作分类。



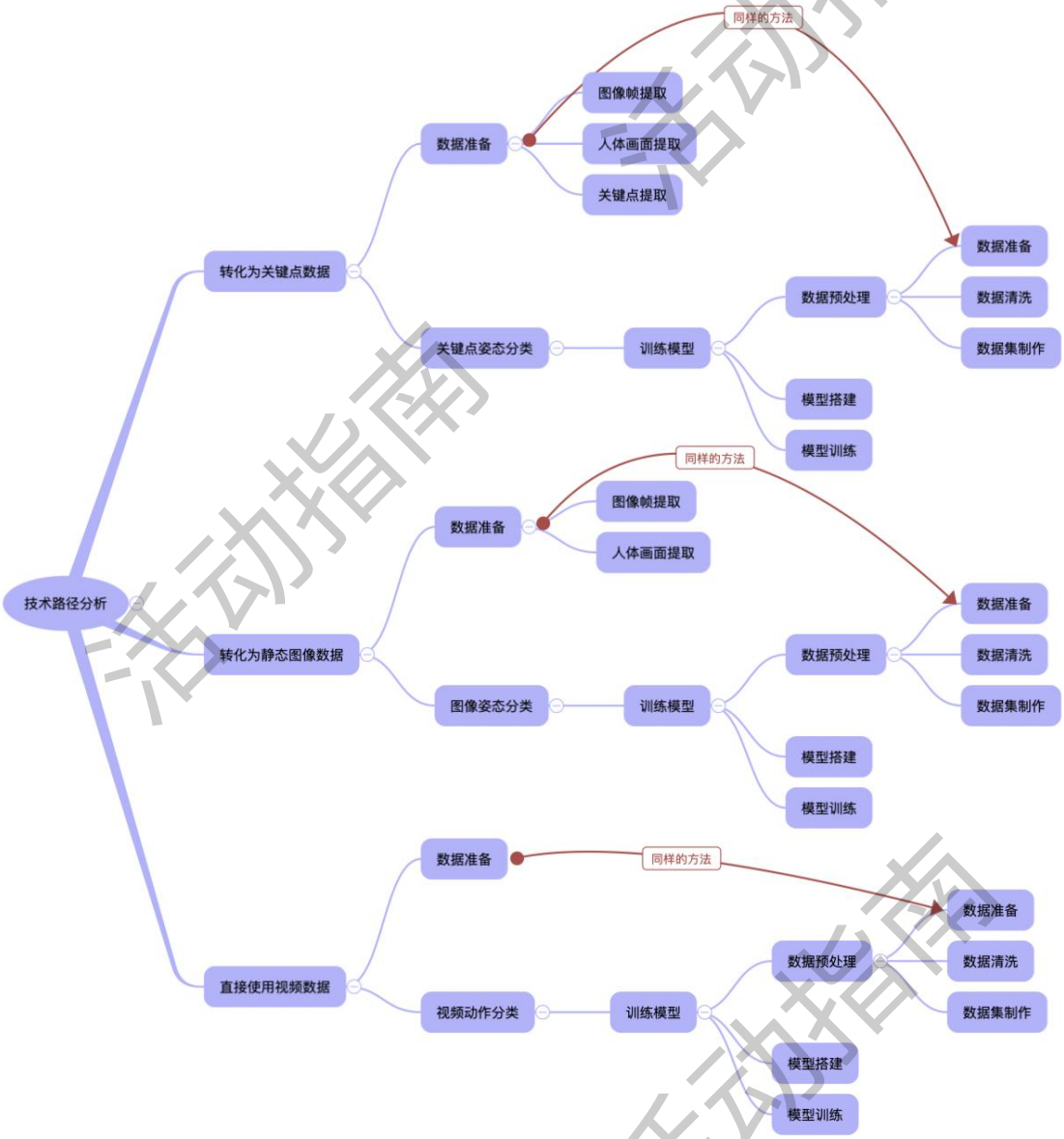
“数据准备”主要包括数据读取、数据转换、格式规整等步骤，“动作分类”主要包括模型推理和结果分析两大步骤。



2. 技术路径分析

为实现“‘五禽戏’动作识别”，“数据准备”和“动作分类”的每个环节都需要制定针对性的技术实现方案，以此形成核心技术路径图。数据准备主要有

三种实现方法，分别是“转化为关键点数据”、“转化为静态图像数据”和“直接使用视频数据”。



“转化为关键点数据”是一种行之有效的简便方法。关键点检测是人体姿态估计的核心技术之一，主要功能是在图片中识别和标注出关键的位置点，例如人脸上的眼睛、鼻子的位置，或者是身体的各个关节位置。一旦这些关键点被准确检测出来，它们通常会以图像中的坐标点坐标形式记录。通过识别和跟踪这些关键点，计算机能够理解图像中的物体形状、姿态和动作，这对于动作识别至关重要。精确的关键点检测技术能够帮助我们更精确、更高效地处理和分析图像数据，尤其是处理用于动作或姿态分类的图像，关键点是这些图像中最具代表性的特征信息。

“转化为静态图像数据”是更为简便的方法。这种方法不追求提取精确的关键点数据特征，遵循传统的图像分类思路将人工智能模型视为“黑箱”，期待人工智能模型通过其数据驱动的学习能力，自主学习数据的特征并实现动作识别。这种方法抽取视频中具有代表性的图像帧作为动作状态的典型特征，使得计算机能够有效识别不同动作状态图像的区别。

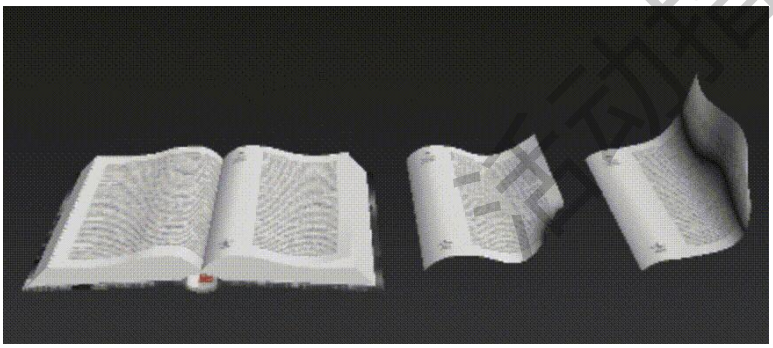
“直接使用视频数据”是一种端到端的解决方案。这种方法无需特别准备数据，就可以直接将视频传入模型中进行推理。正因如此，模型一次性要处理的数据量远大于前两种方法，需要更大的算力和空间，对算法的设计难度要求也更高。

动作分类和姿态分类需要经过数据预处理、模型搭建和模型训练三个步骤，其中数据预处理中的数据准备与动作识别流程中的数据准备具有完全相同的技术路线，均可采用上述三种方法实现。因此，尽管上图看起来复杂，实际上涉及的核心技术并不多。

3. 核心技术概要

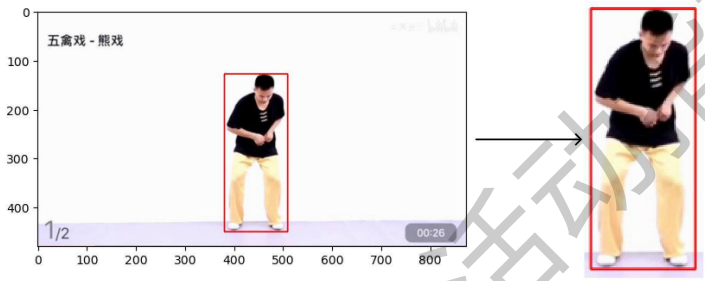
3.1 图像帧提取

计算机视觉是人工智能领域的重要应用之一。为了降低视频处理的难度，需要先将视频拆分为图像帧（frame）。这一过程可以通过多种算法实现，OpenCV是最常用的实现方式，它提供了视频读取、图像帧提取和保存等丰富的视频处理功能。



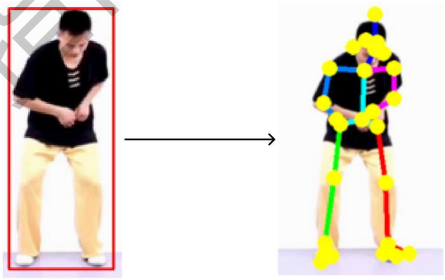
3.2 人体画面提取

在视频画面中，除了人体画面，还有画面背景等其他干扰因素。为了排除干扰因素，我们可以通过提取人体画面来划定要识别的准确区域，帮助模型提升识别能力。XEduHub 中提供了一种简洁的人体画面提取方案，实现效果如下：



3.3 关键点提取

在获取含人体动作状态的图片后，如果想要进一步提取关键点数据，可使用关键点检测技术。XEduHub 提供了人体关键点检测工具，比如选择 task='pose_body26'表示提取每个人体图像 26 个部位的关键点数据，实现效果如下：



3.4 数据清洗

数据清洗是数据预处理的重要步骤，包括识别并纠正错误数据、不一致的数据、重复的数据和不完整的数据。在进行数据清洗时，通常需要执行以下操作：首先，对数据进行探索性分析，了解数据的基本情况和潜在问题；接着，通过删除或填充缺失值、平滑噪声数据、识别和处理异常值、解决重复数据等方法来纠正这些问题；最后，进行数据一致性检查和格式化，确保数据的准确性和可用性。数据清洗能够提高数据的质量，为后续的数据分析和建模奠定良好基础。

在“‘五禽戏’动作识别”任务中，学生可以检查视频数据集的准确性和完备性，对存在错误的数据进行删除或重新归类，也可以对视频进行裁剪、编辑等操作。完成数据预处理之后，可以对数据进行探索性分析，并删除低质量数据。

3.5 数据集制作

制作数据集一定要明确数据集的目标和用途，这将指导数据的收集和选择。当确定好使用的技术路径和人工智能模型工具后，要根据其数据要求和规范来规整数据格式，并确保数据之间的一致性（例如都是用同一种归一化策略进行数据规范）。最后，将处理后的数据按照一定的结构和格式组织起来，形成最终的数

据集，如果是数值型数据，通常以表格或数组的形式存储，如果是图片数据，通常以 ImageNet 格式管理。

3.6 模型搭建和训练

模型搭建是机器学习的核心环节。完成模型搭建后，学生可使用经过预处理的数据集进行模型训练（包括特征选择、特征工程等步骤），以提取有用的信息。在训练模型的过程中，学生可通过超参数调优，即调整模型的参数来优化模型的性能。同时，学生可使用验证集来评估模型的表现，防止过拟合现象。最后，使用测试集对模型进行最终评估，确保模型具有良好的泛化能力。完成以上步骤，学生就可以将模型部署到实际应用中，做出预测或决策。

4. 总结

本篇文档介绍了 AI 算法挑战“‘五禽戏’动作识别”的技术路径，下一篇文档将以“转化为关键点数据”的方法为例，带你了解完成动作识别任务的具体实现形式与核心技术。

附录 4-3

“‘五禽戏’动作识别”核心技术分解

经过前面技术分析篇的学习，相信你熟悉了完成 AI 算法挑战任务的技术方案和核心步骤。本篇文档聚焦“转化为关键点数据”这一技术路径，继续带你详细了解“‘五禽戏’动作识别”的核心技术。

示例项目：

https://openinnolab.org.cn/pjlab/project?id=662f588e514bdf08f291ca26&sc=645caab8a8efa334b3f0eb24#public

1. 图像帧提取

1.1 提取图像帧的意义

实现“数据准备”需要将视频数据降维成图像帧。图像帧是指视频中的图像画面，其意义在于将动态的视频转化为静态的图像，从而使得计算机视觉算法能够在图像层面上进行特征提取和分类。其优点如下：

（1）化简高维数据。视频数据蕴含时间序列信息，意味着图像数据具有第三维度——时间维度。将视频抽帧为图片可以将动作图像处理简化为静态图像处理。我们可以单独对图像进行特征提取，从而更加准确地捕捉动作图像的关键特征。完成图像关键特征提取后，将特征与时间维度相结合，实现高维数据降维。

（2）提高计算效率。直接处理整个视频数据会消耗大量的计算资源。由于视频文件的帧密度比较高，连续的两帧图像之间变化量很小，我们可以通过抽取部分图像帧，提高算法的运行效率和实时性。这也是挑战的重要评估指标之一。

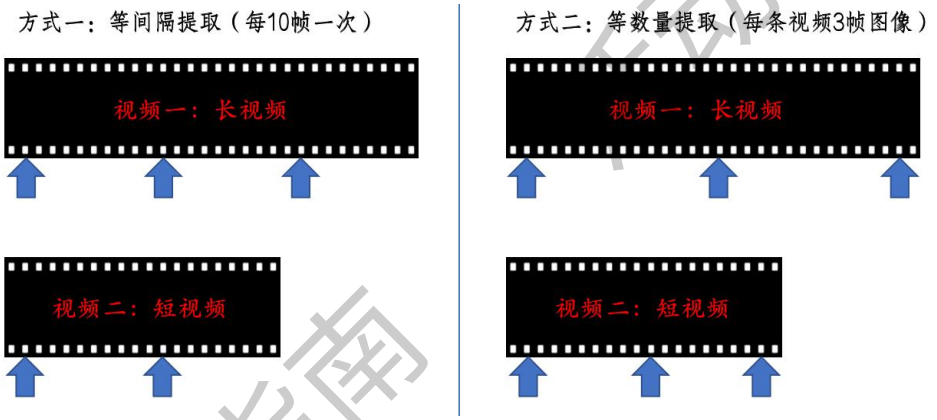
1.2 提取的策略

从视频中提取图像帧有多种策略。从提取方法来看，我们可以通过人工观察，提取最为重要的几帧图像作为图片数据集，但这种方法效率较低。因此，建议增加图像帧的数量，在一个视频中抽取多个图像帧。尽管这中间可能穿插着一些质量不佳的图像（脏数据），但足够多的高质量数据可以弱化低质量数据的影响。

从操作策略来看，我们可以采用从视频中任意选择一帧的方式，但这种策略存在诸多不确定性，图像数据不够井然有序。因此，建议采取均匀提取帧图像的策略，统一对每条视频的操作方式，例如都提取视频中的第 1 帧，或者第 N 帧，

或者最后 1 帧，或者中间帧（类似于中位数）。

均匀提取多帧图像有多种操作方式，比如等间隔提取，等数量提取等，如下图所示。



等间隔提取指指间隔相同帧数提取图像，例如每 10 帧视频提取一次图像；等数量提取是指无论视频长短，都提取同样的帧数量，例如每条视频均匀抽取 3 帧图像，两种方式各有优势。对于一个五禽戏动作而言，先做什么后做什么有比较固定的步骤，但通过我们提供的视频可以发现，人物的动作有快有慢，视频的时长也不一致。因此，如果采用等数量提取方式，可以通过不同视频的同一个动作节点，提取到高度相似的动作。如果采用等间隔提取方式，图像提取具有较大的随机性和不确定性，能提取到什么状态的动作全凭“缘分”。基于此，我们建议学生选择等数量提取方式提取图像帧。

此外，起始帧位置的选择也很重要。五禽戏动作都有一个固定的初始状态，这个状态几乎为所有动作共有。我们可以跳过这个初始动作，将关键帧定在稍后的位置进行图像抽取。

由此可见，在制作训练数据集时，采用均匀提取多帧图像的策略具有诸多优势，特别是通过等数量提取方式，可以保证视频动作状态的一致性。当然，新视频的推理策略也有多种可选方案，例如根据随机的一帧图像结果判定，或者由多帧图像联合投票判定，甚至是使用多个模型联合判定等。

1.3 图像帧提取的实现

1.3.1 图像帧提取核心逻辑

图像帧的提取包括以下 3 个步骤：

（1）视频读取：使用视频处理库打开视频文件。

（2）图像帧提取：根据需求选择合适的提取策略，获取图像。

（3）图像帧保存：将提取的帧作为图像保存到指定的文件路径。

下面讲解如何提取视频最中间的一帧图像：给定一个视频路径 video_path，得到图像帧 frame，并保存到路径 frame_path。为了便于重复使用相同的功能，我们可以使用函数对这个功能进行封装，最终代码如下：

```
Python
def make_image_data(video_path):
    '''
    函数提取视频的中间帧并返回最终的图像
    '''
    cap = cv2.VideoCapture(video_path)
    #  获取视频总帧数
    total_frames = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))
    frame_number = int(total_frames // 2)
    print('一共有',total_frames,'帧，现在读取第',frame_number,'帧。')
    cap.set(cv2.CAP_PROP_POS_FRAMES, frame_number)
    ret, frame = cap.read()
    cap.release()
    return frame

#  调用函数
frame = make_image_data('test_video/熊运_440_1177.avi')

#  保存图像
save_path = 'my_frame.png'
cv2.imwrite(save_path, frame) #  如果不需要保存，可以去掉这一句代码
print('图像已保存至', save_path)
```

1.3.2 图像帧提取函数的实现

为了便于调用，我们封装了多种图像帧提取方式的函数，程序中的 frame_number 是指要提取的帧编号。

方式 1：等间隔提取

下列代码中设置了视频文件路径 video_path、视频文件对应的类别标签 label，以及提取的间隔 interval 三个参数。前两个参数用于读取视频并将图像帧保存为包含类别标签命名的图片。interval 是关键参数，如果要实现从每个视频中每 10 帧提取一帧图像，可将 interval 设置为 10。

```python
import cv2
def make_image_data_by_interval(video_path, label, interval):
    '''
    video_path: 视频文件的路径
    label: 视频文件对应的类别标签
    interval: 提取的间隔
    '''
    cap = cv2.VideoCapture(video_path) # 读取视频
    total_frames = int(cap.get(cv2.CAP_PROP_FRAME_COUNT)) # 获取视频总帧数
    for i in range(int(total_frames//interval)):
        frame_number = i * interval # 将视频指向第 i 个间隔
        cap.set(cv2.CAP_PROP_POS_FRAMES, frame_number)
        ret, frame = cap.read() # 读取图像
        if ret:
            frame_path = f'./%s_%04d.jpg' %(label,frame_number) # 构造图片文件名为标签+帧序号
            cv2.imwrite(frame_path, frame) # 将数据写入磁盘
            print('图像已保存至', frame_path)
    cap.release()
```

方式 2：等数量提取

相比于上一种方法，此种方式需要计算提取的帧间隔。可以通过关键参数 num 设置计划提取的帧数，interval 的计算公式为 `total_frames // num`。

```python
import cv2
def make_image_data_by_num(video_path, label, num):
    '''
    video_path: 视频文件的路径
    label: 视频文件对应的类别标签
    num: 一个视频要提取多少帧图像
    '''
    cap = cv2.VideoCapture(video_path) # 读取视频
    total_frames = int(cap.get(cv2.CAP_PROP_FRAME_COUNT)) # 获取视频总帧数
    interval = max(1,total_frames // num)
    for i in range(num):
```

```
            frame_number = i * interval # 将视频指向第 i 个间隔
            cap.set(cv2.CAP_PROP_POS_FRAMES, frame_number)
            ret, frame = cap.read() # 读取图像
            if ret:
                frame_path = f'./%s_%04d.jpg' %(label,frame_number) # 构造图片文件名为标签+帧序号
                cv2.imwrite(frame_path, frame) # 将数据写入磁盘
                print('图像已保存至', frame_path)
    cap.release()
```

方式 2 的升级：指定起始帧、结束帧的等数量提取

这种方法可以实现视频中部的视频片段提取，从而忽略起始和收尾动作的固定状态。下列代码添加了起始帧序号 begin 和结束帧序号 end 两个参数，interval 的计算公式为 `(end - begin) // num`。

```
Python
import cv2
def make_image_data_by_num_pro(video_path, label, num, begin=0, end=-1):
    '''
    video_path: 视频文件的路径
    label: 视频文件对应的类别标签
    num: 一个视频要提取多少帧图像
    begin: 起始帧序号，默认为 0
    end: 结束帧序号，默认为-1，表示不指定
    '''
    cap = cv2.VideoCapture(video_path) # 读取视频
    total_frames = int(cap.get(cv2.CAP_PROP_FRAME_COUNT)) # 获取视频总帧数
    if end == -1:
        end =total_frames
    interval = max(1,(end - begin) // num)
    for i in range(num):
        frame_number = begin + i * interval # 将视频指向第 i 个间隔
        cap.set(cv2.CAP_PROP_POS_FRAMES, frame_number)
        ret, frame = cap.read() # 读取图像
        if ret:
            frame_path = f'./%s_%04d.jpg' %(label,frame_number)#  构造图片文件名为标签+帧序号
```

```
            cv2.imwrite(frame_path, frame) # 将数据写入磁盘
            print('图像已保存至', frame_path)
    cap.release()
```

1.4 批量提取并制作图像数据集

单条视频可以使用上面的函数进行抽帧。如果数据集中存在大量视频数据，也可以复用上面定义的功能，遍历数据集内的各个视频，逐一完成视频图像帧的提取，完成图像数据集制作。批量提取图像的相关代码和图像数据集已在配套项目中提供。下面详细介绍批量提取并制作图像数据集的步骤。

os 是 Python 的标准库之一，提供了许多与操作系统交互的函数。这些函数可以用来处理文件和目录，运行其他程序，以及获取系统信息等。我们可以使用 os 库来遍历包含动作类别文件夹的数据集，每个类别文件夹中包含多段五禽戏动作视频。

```python
# 遍历文件列表
import os
for video_name in video_path_list:
    # 指定测试集每条视频路径
    video_path = os.path.join(rootpath, video_name)

    # 在这里可以添加处理每个文件的代码
    # 例如：读取、分析、模型推理等
```

以下提供了遍历特定文件夹中的所有文件的示例代码：

Python 基础教程推荐：https://www.runoob.com/python/python-tutorial.html

实际上，前面提供的图像帧提取代码仍存在一些不足，我们可以通过以下方式进行优化：

（1）文件名冲突。为了避免文件名重复，同时更好地区分不同图像帧的来源，可以为每条视频编定唯一序号（global_seq），并设置在文件名中。

（2）文件夹缺失。为了能够按照类别存放文件，存放路径中依据类别名称建立文件夹，可以使用 os.path.exists(dataset_path)方法检查是否已事先手

动创建文件夹。如果事先未创建文件夹，可以使用 os.makedirs(dataset_path) 方法创建。

（3）优化 num 的缺失值。如果不指定 num 参数值，则系统默认提取所有图像帧，意味着无论视频长短如何，interval 都是 1，提取的图像数量为视频的总帧数。

下列代码针对方法 2 的函数进行了简单修改，代码如下（标红部分）：

```
Python
import cv2
def make_image_data(video_path, label, num=-1,global_seq=0):
    '''
    video_path: 视频文件的路径
    label: 视频文件对应的类别标签
    num: 一个视频要提取多少帧图像，默认-1 表示提取所有图像帧
    global_seq: 对每个视频文件，给定一个唯一的编号
    函数会将数据保存至 wuqinxi_datasets_文件夹，并存放在不同类别名称的文件夹内
    '''
    cap = cv2.VideoCapture(video_path) # 读取视频
    print(video_path)
    total_frames = int(cap.get(cv2.CAP_PROP_FRAME_COUNT)) # 获取视频总帧数
    interval = max(1,total_frames // num)
    dataset_path = os.path.join('./wuqinxi_datasets',label)
    if not os.path.exists(dataset_path): # 创建目标图片子文件夹（如果不存在）
        os.makedirs(dataset_path)
    if num == -1:
        interval = 1 # 抽取每一帧图像
        num = total_frames
    for i in range(num):
        frame_number = i * interval # 将视频指向第 i 个间隔
        cap.set(cv2.CAP_PROP_POS_FRAMES, frame_number)
        ret, frame = cap.read() # 读取图像
        if ret:
            frame_path = os.path.join(dataset_path, f'%s_%04d_%04d.jpg' %(label,global_seq,frame_number)) # 构造图片文件名为标签+视频序号+帧序号
```

```
            cv2.imwrite(frame_path, frame) # 将数据写入磁盘
    cap.release()
```

结合下面的遍历视频数据集的代码，再使用上面封装的 make_image_data 函数，我们可以高效地从每个视频中抽取关键帧，并将这些帧按照其对应的动作类别存储到新的图片数据集中。

如果要使用下面的代码批量抽帧，需要将 path 的值修改为实际视频数据集所在的文件夹。

```
Python
import os
path = './video/' # 视频数据集的路径
global_seq = 0      # 唯一确定视频的编号
dirs = os.listdir(path) # 获取类别名称，也是文件夹名称
for d in dirs:  # 遍历每个文件夹
    # 跳过非文件夹项
    if not os.path.isdir(d):
        continue
    video_list = os.listdir(os.path.join(path,d)) # 获取每个视频文件名
    for v in video_list: # 遍历每一条视频，对其抽帧
        if not (v.endswith(('.mp4', '.avi', '.mov'))):
            continue
        make_image_data(os.path.join(path,d,v),d,15,global_seq) # 这里的 d 就是
label，15 是抽 15 帧图像
        global_seq += 1 # 对视频文件唯一编码
```

上述图像数据集经过整理后，便实现将视频数据集转换为图像数据集。我们也为你提供了一个制作好的五禽戏图像数据集（下载链接：https://openinnolab.org.cn/pjlab/dataset/662e1271ae25fb259791e3be）。

小思考：

视频中的哪些问题会造成数据质量不佳？是否可以通过拍摄和剪辑视频的方式优化数据集？

二、关键点检测

获取从视频中批量提取的帧图像后，接下来可以使用这些图片训练一个能识别五禽戏动作的模型。在训练模型的过程中，提取图片特征是关键。我们可以通过关键点检测技术，对这类包含动作或姿势特征的图片进行特征提取。下文将介

绍关键点检测任务，讲解如何巧妙地利用关键点检测技术提取图像的关键特征，使庞大的视频数据集变为主要包含数值型数据的 CSV 表格数据集。

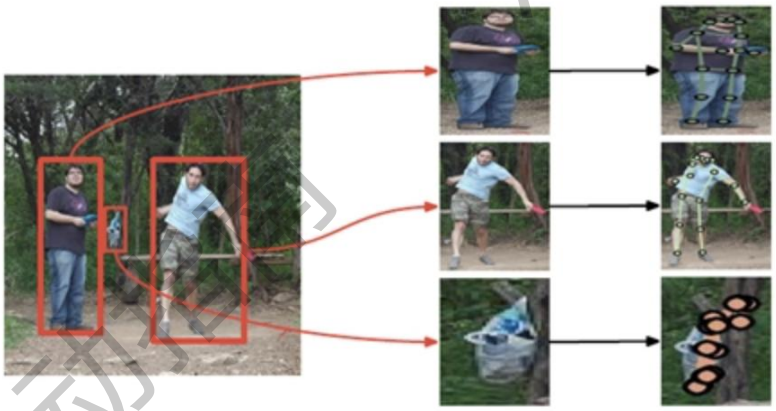
2.1 认识关键点检测

关键点检测是计算机视觉领域的核心任务之一，主要功能是在图片中识别和标注出关键的位置点，例如人脸上的眼睛、鼻子，或者是身体的各个关节位置。这些关键点一旦被检测到，通常会以图像中的坐标点形式进行标记。通过识别和跟踪这些关键点，计算机能够理解图像中的物体形状、姿态和动作，这对实现动作识别、人脸识别、增强现实等任务至关重要。精确的关键点检测技术能够帮助我们更精确、更高效地处理和分析图像数据，尤其是处理用于动作或姿态分类的图像，关键点是这些图像中最具代表性的特征信息。

2.2 人体画面提取：关键点检测的前处理

关键点检测任务的模型具有很强的泛化能力，即使人像不在画面中间，也能够发现其位置并识别，给出各个关键点的坐标信息。然而，当画面比较复杂、有前景和背景时，这些干扰因素有时也会被识别为人体的一部分，进而影响识别的效果。另外，关键点模型只适用于画面中只有一个人的场景，如果画面中有多个人，模型经常会把其他人的躯干错误地组合在某个人身上。

因此，有必要使用专门的人体目标检测模型，来帮助计算机定位人体位置，继而针对每一个提取到的画面区域进行关键点检测，或者仅对画面最中央的一个人进行关键点检测。人体画面提取使用的人体目标检测模型，通常标出一个矩形区域，完整包围人体所在的画面，并给出对应边界框的坐标信息（bounding box，bbox）。借助这个信息，我们可以将原图裁剪保存并进行处理，也可以直接在原图上进行新的关键点检测。



2.3 常用的关键点检测技术和方法

如何完成关键点检测呢？目前有越来越多先进的技术和方法支持关键点检测，也有很多开源的模型或易用的商业 API 可以直接调用。下面列举几种常用的关键点检测方法。

（1）基于深度学习的方法。这是目前较为流行且有效的方法，使用深度神经网络（例如卷积神经网络 CNN）从输入的图像中直接学习和预测关键点的位置。常见的深度学习框架有 OpenPose、HRNet、MMPose 等。这类方法能够自动学习特征，无需人工设计，但要自己训练模型，且通常需要大量的标注数据和计算资源。

（2）使用预训练模型的方法。这种方法直接利用已经训练好的模型，如 Google 的 MediaPipe、XEdu 的 XEduHub 等工具，这些工具提供了一系列预训练的深度学习模型，可以直接在用户的设备上运行，进行关键点检测和姿态估计。用户仅需下载并安装相应的库，即可在自己的代码中调用这些模型，相较更为快捷。

（3）通过 API 调用的方法。这种方法通过使用服务提供的 API 进行关键点检测，如百度 API。用户仅需通过网络发送图片，服务端即可返回检测结果，极大简化了关键点检测的复杂度。此外，此类服务通常会提供较为先进的模型和算法，服务提供方负责维护更新，可以省去用户大量的开发和运维工作。然而，这种方法可能会因为网络延迟、服务成本以及请求次数限制等因素，在某些应用场景中受到限制。

2.4 借助 XEduHub 实现对人体关键点的获取

XEdu 工具集中的 XEduHub 是一个专为快速、便捷地利用最先进的深度学习模型完成任务而设计的工具库。XEduHub 内置了大量优质的深度学习 SOTA 模型，包括多个关键点检测模型，如 pose_body26、pose_body17、pose_wholebody133。这些模型都是已经在大型数据集上进行了训练的预训练模型，因此能够直接应用于新的数据上且效果良好。例如，我们常用 pose_body26 这个模型来检测人体的 26 个关键点坐标。

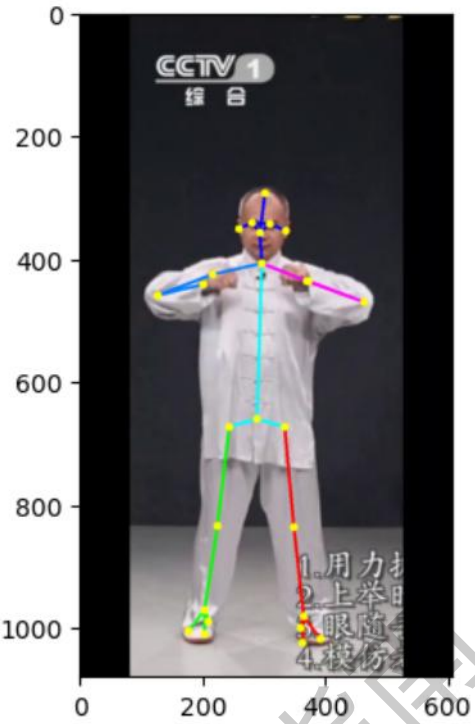
单人关键点检测的示例代码如下，通过选择任务（task）确定实例化的模型并载入，接着指定图片完成模型推理。

```python
from XEdu.hub import Workflow as wf
```

```
body = wf(task='pose_body26' ) # 实例化 pose_body26 模型
img_path = '1.jpg' # 指定一张图片
result,image = body.inference(data=img_path,img_type='cv2')  # 模型推理
re = body.format_output(lang="zh") # 格式化输出结果
body.show(image) # 可视化检测结果图
```

上述代码运行结果如下，输出的分别是使用 format_output 展示格式化后的检测结果和使用 show 展示的检测结果图。

```
pose_body26任务模型加载成功!
{'关键点坐标': [[293.453125, 358.06640625],
            [309.2734375, 344.8828125],
            [280.26953125, 342.24609375],
            [335.640625, 355.4296875],
            [259.17578125, 352.79296875],
            [369.91796875, 437.16796875],
            [216.98828125, 426.62109375],
            [462.203125, 471.4453125],
            [127.33984375, 460.8984375],
            [372.5546875, 437.16796875],
            [201.16796875, 442.44140625],
            [333.00390625, 674.47265625],
            [243.35546875, 674.47265625],
            [348.82421875, 837.94921875],
            [224.8984375, 835.3125],
            [364.64453125, 982.96875],
            [203.8046875, 972.421875],
            [301.36328125, 294.78515625],
            [296.08984375, 408.1640625],
            [288.1796875, 661.2890625],
            [362.0078125, 1025.15625],
            [203.8046875, 1011.97265625],
            [391.01171875, 1019.8828125],
            [177.4375, 1006.69921875],
            [359.37109375, 1001.42578125],
            [209.078125, 990.87890625]],
 '分数': [0.98095703125,
        1.021484375,
        1.005859375,
```



代码运行时如果出现下面的提示，表明 wf()这句代码运行后在完成预训练模型的下载，模型下载完即可使用。

```
本地未检测到pose_body26任务对应模型，云端下载中...
warning: Local config must not be empty before get token via api. Please use the 'openxlab config' command to set the config
100%|■■■■■■■■■■| 27.9M/27.9M [00:01<00:00, 22.2MiB/s]
```

上面的代码针对的是图片中只有一个人打五禽戏的情况，如果图片中有多个人打五禽戏，可以在关键点提取前进行人体画面提取。XEduHub 中提供了一种简洁的人体画面提取方案，通过载入 XEduHub 中内置的预训练模型 det_body，即可先检测出几个人在打五禽戏，然后对每个检测出的人进行关键点检测。实现代码如下：

```Python
from XEdu.hub import Workflow as wf # 导入库
```

```
det    = wf(task='det_body') # 实例化人体目标检测模型
model = wf(task='pose_body26') # 实例化关键点检测模型
img_path = '2.jpg' # 指定进行推理的图片路径
bboxs,new_img = det.inference(data=img_path,img_type='cv2') # 进行人体画面提取
for i in bboxs: # 遍历提取到的人
    keypoints,new_img = model.inference(data=new_img,bbox=i,img_type='cv2') # 进行关键点检测推理
model.show(new_img) # 可视化结果
```

上述代码的运行结果如下，这段代码既支持多人关键点检测，也支持单人关键点检测，可以帮助我们将每张图片中所有人物的关键点信息检测出来。

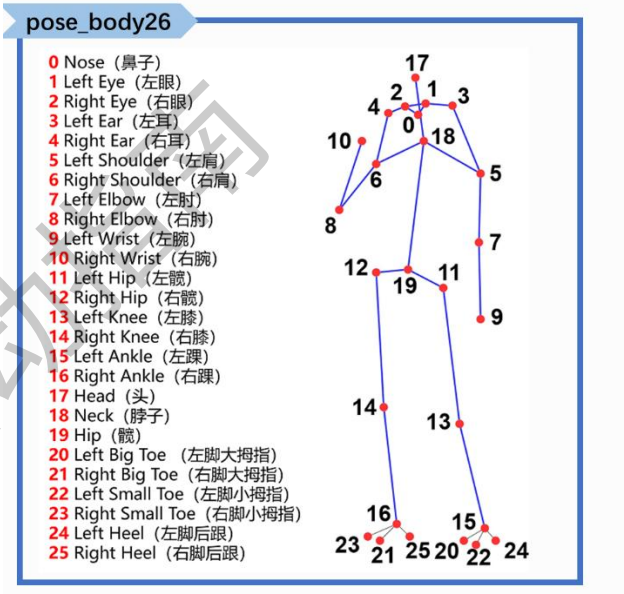
小思考：

如果画面中存在多个人，可能对关键点检测结果产生什么样的影响？如何调整检测策略来优化关键点检测效果？

2.5 关键点的数据结构与简单处理

使用不同的工具和不同的关键点检测模型，检测出的关键点的数量、顺序、数据结构等可能略有差异。因此，当我们选用某种方法后，务必要深入理解该模型输出数据的结构和含义，以便于后续分析和应用关键点信息。

例如，使用 XEduHub 的 pose_body26 模型检测一个人的 26 个关键点，inference 的输出结果是一个二维数组，包含了 26 个关键点的[x,y]坐标信息，每组坐标值以像素点为单位，图片左上角为（0,0）。每个关键点的序号和骨骼位置对应关系如下图所示，假设一组检测结果是 keypoints，那么右肘的关键点是 keypoints[8]，右肘的横坐标和纵坐标分别是 keypoints[8][0]和 keypoints[8][1]。



检测出关键点后，我们还需对关键点进行简单的信息处理，以实现动作判定。例如，可以使用某些关键点计算位置关系，得到一个新的特征，或者直接定义一个判断动作的函数。代码如下：

```python
def clsasify(keypoints):
    #   如果
    if keypoints[9][1] < keypoints[1][1]:
        return "pose1"
    else :
        return "pose2"
```

通过上述的逻辑判断方式，可以实现简单的分类效果。然而，如果想要写出能把五禽戏十个动作都准确识别出来的函数，则难度重重。由此，我们推荐学生通过训练模型的方法，而不是直接通过编写判断逻辑代码来完成算法挑战任务。

2.6 批量完成关键点检测并制作表格数据集

为构建能够满足后续训练需求的数据集，我们可以对从视频中提取的图片进

行进一步处理，通过关键点检测技术将每张图片中的关键信息转化为关键点数据。这一过程涉及到关键点批量检测任务，XEdu 团队已经开发了相应的代码，并完成了批量处理工作。

完成关键点批量检测后，我们可以得到一个新的表格型数据集，包含每张图片的关键点坐标信息以及对应的标签，不仅极大地提高了数据的可用性和处理效率，而且便于存储和传输，能够有效支持后续的机器学习和深度学习任务。

批量完成关键点检测的核心代码如下，先截取人体画面，然后进行关键点检测，并支持判断画面中是否有人。关键点检测的结果都将保存到 keypoints_list 文件中。这段代码可用于模型训练和模型推理阶段，因为两个阶段对数据处理的操作一致。需要说明的是，学生可以自行更换关键点检测模型和检测工具，修改好相关代码即可。

```python
from XEdu.hub import Workflow as wf # 导入库
det   = wf(task='det_body') # 实例化人体目标检测模型
model = wf(task='pose_body26') # 实例化关键点检测模型
img_path = '2.jpg' # 指定进行推理的图片路径
bboxs = det.inference(data=img_path) # 进行人体画面提取

# 检查 bboxs 是否存在（即是否检测到人）
keypoints_list = []
if bboxs.size > 0:
    for i in bboxs:
        # 这里假设每个 bbox 都是合适的格式，您可能需要根据实际情况做调整
        keypoints = model.inference(data=img_path, bbox=i) # 进行关键点检测推理
        keypoints_list.append(keypoints)
    print(keypoints_list)
else:
    print("没有检测到任何边界框")
```

上述代码的主要作用是遍历每个子文件夹中的图片，以及获取每张图片的类别名。提取的过程耗时较多，如果学生想体验这个过程，建议到本地完成，将数据集和代码文件下载至本地，修改图片路径为自己的路径，甚至可以更换模型，将结果保存到 CSV 表格文件中。为了减小电脑的内存压力，代码中设置了每提取一个文件夹就生成一个中间文件的操作，并生成一个整合版的 CSV 表格文件。

修改关键点检测模型的操作实例代码如下：

```Python
def get_keypoints_from_image_path(class2id,total,image_path):
    det    = wf(task='det_body') # 实例化模型
    body = wf(task='pose_body26') # 实例化模型
    feature_data = []     # 存储特征值
    labels = []           # 存储标签
    filenames = []        # 存储文件名
    error_messages = []   # 收集错误信息

    print("图片姿态关键点提取中")

    error_messages = [] # 创建一个列表来收集错误信息

    with tqdm(total=total_images) as progress_bar:
        for image in image_path:
            bboxs = det.inference(data=image) # 进行人体画面提取
            if bboxs.size > 0:
                keypoints_list = []
                for i in bboxs:
                    keypoints = body.inference(data=image,bbox=i) # 进行关键点检测
                    keypoints_list.append(keypoints)
                if keypoints_list:
                    # 展平数组
                    features = np.concatenate(keypoints_list).reshape(len(keypoints_list), -1)
                    if features.size > 0 and len(features[0]) == 26 * 2:
                        filenames.extend([image] * len(features))     # 因为可能会出现一张图片中有多个姿态，因此在此乘上识别出的姿态数量
# 后面代码省略
```

修改五禽戏图片数据集路径的操作如下：

```Python
# 指定训练集图片对应路径
root_folder = r"datasets"
```

```
# 处理训练数据
subfolders = read_folder(root_folder)
print(subfolders)
# 计算图片数量
for i in range(len(subfolders)):
    print('正在提取第',i,'个文件夹')
    total_images, image_path = collect_image([subfolders[i]])
    feature_data,labels,filenames =
get_keypoints_from_image_path(class2id,total_images,image_path)
    write_to_csv(feature_data,labels,filenames,'workflow_pose-'+str(i)+'.csv')
```

修改保存路径和保存文件名字的操作：

```
Python
# 生成整合版 CSV
import pandas as pd
for i in range(len(subfolders)):
    df = pd.read_csv('workflow_pose-'+str(i)+'.csv')
    if i==0:
        final = df
    else:
        final = pd.concat([final,df])
final.to_csv('workflow_pose-final.csv',index=False) # 可修改保存路径和文件名
```

通过运行上面的代码，可将庞大又复杂的视频数据集转变为图片数据集，图片数据集保存了视频的各帧图像。在此基础上，又将图片数据集转变为只有几十兆内存的表格数据集，保留了视频数据的关键特征。表格数据集的内容为便于计算机识别和处理的数值型数据。由此，我们实现了将视频的视觉内容转化为契合数据处理流程的结构化数据！

1]:

	0	1	2	3	4	5	
0	Path_Filename	Feature 1	Feature 2	Feature 3	Feature 4	Feature 5	
1	wuqinxi15_imagenet/target_path/training_set\0_...	667.4216547182629	266.53638628976694	677.8418667082276	256.1161742998021	658.0434639272945	2
2	wuqinxi15_imagenet/target_path/training_set\0_...	664.2465464345047	243.17285618611743	674.304229617119	234.23269335712712	654.1888632518906	2
3	wuqinxi15_imagenet/target_path/training_set\0_...	662.2697796033963	230.34783752369026	672.4090911474611	221.33511615118812	652.1304680593312	22
4	wuqinxi15_imagenet/target_path/training_set\0_...	663.9589577061789	229.05541573251992	672.7561281408582	222.45753790651042	655.1617872714996	22
...	...	...	...	...	...	...	
13901	wuqinxi15_imagenet/target_path/training_set\9_...	560.9178732761314	268.0992996399956	563.5536339745989	265.1706766416984	557.6963879780045	2
13902	wuqinxi15_imagenet/target_path/training_set\9_...	357.31438212735316	289.0082085345473	360.98333656362126	285.33925409827907	353.978969003473	28
13903	wuqinxi15_imagenet/target_path/training_set\9_...	488.2942451291851	276.93308426333317	491.200520137591	273.3002405028258	484.29811699262683	27
13904	wuqinxi15_imagenet/target_path/training_set\9_...	434.96169523894787	274.4407710007259	437.7263422672238	271.39965926962236	431.9205835078444	27
13905	wuqinxi15_imagenet/target_path/training_set\9_...	151.5016221585018	270.1225826277264	154.825399564313	268.0074515513011	150.2929758291159	2

运行以上示例代码即可得到表格型数据集，实现将对训练集做过批量图像帧

提取生成的图像数据集转换为一个关键点数据集。我们也为你提供了一个我们帮你制作的数据集供你参考（下载地址：https://openinnolab.org.cn/pjlab/dataset/662f67b2ae25fb2597727450）。

三、训练前的数据预处理

数据预处理犹如烹饪前的食材准备，是数据分析和机器学习中非常重要的步骤。类似食材准备的一般步骤，即洗净蔬菜、去掉不好的部分、切好肉块、调配好调料等，数据预处理旨在让原始数据变得干净、有序、易于使用。前面的图像帧提取和关键点检测都可以视为数据预处理的环节，此外些数据增强算法也属于数据预处理的范畴，在此不展开介绍。

接下来，让我们一起了解如何整理、规范数据集的格式，以便于模型训练。

3.1 数据集的认识

进行训练模型需要先掌握数据集的格式和规范。你是否还记得经过图像帧提取、人体画面提取和关键点检测等处理后的表格数据集呢？

五禽戏的图像数据集包含了 10 种不同的五禽戏动作的图像，分别放在 10 个以类别标签命名的文件夹中。关键点表格数据集保存的是数值型数据，学生可以下载后查阅，也可以利用相关 Python 库来辅助分析。例如，可以利用 pandas 等数据分析工具库读取和分析关键点 CSV 表格数据文件的内容，参考代码如下。

```python
import pandas as pd
df = pd.read_csv("feature_data/workflow_pose.csv")
print(df)
```

3.2 常见的数据集格式

不同的 AI 训练工具对数据集的格式有不同的要求。为了满足不同任务、工具的需求，数据集的格式呈现出多样化特征。其中，CSV（Comma-Separated Values）格式以其简洁和通用性广泛应用于数值型数据集中，通过逗号分隔文本文件来存储表格数据，适合机器学习任务。典型的 CSV 文件中，所有特征为数值类型，第一行作为表头，说明各列数据的意义；每行提供一条样本数据，样本的最后一列通常用于存储标签或类别序号。ImageNet 格式通常用于图像分类数据集，它本身为著名的计算机视觉竞赛，提供了统一的数据集让不同算法比较优劣。

3.3 数据集的划分

数据集常被分为训练集（Training set）和验证集（Validation set）两种类别，以便进行模型训练和性能评估。此外，为了公平较量，设置测试集（Test set）用于评估不同模型之间的的最终性能优劣。下图形象比喻了三种数据集的关系，主办方提供了训练集以便学生训练模型，也提供了测试集用于评估和比较模型训练的效果。

在训练模型前，强烈建议学生从训练集中再划分出验证集，以确保模型能够在新的数据上取得良好表现，这种能力叫做模型的“泛化”能力。因此，训练集、验证集和测试集的数据要互不重复、保持独立，尤其不能将验证集和测试集的数据加入到训练集中训练，否则会导致模型评估不准确，甚至出现“过拟合”的情况（类似学习中的“死记硬背，不会变通”），看似效果很好，但不能适应新的数据。所以，需要设置验证集评估模型的性能，帮助我们调整模型训练策略。

3.4 利用代码完成数据集划分

目前我们的数据是有序排列的，训练模型的时候，如果不打乱数据顺序，那么模型也是按照给定数据来学习的。打乱数据顺序是为了确保模型学习到的是数

据的内在规律而非表面的顺序偏差，这有助于提高模型的泛化能力、避免过拟合，并模拟真实世界中数据的随机性，从而提升模型在新数据上的表现。

但是这里我们的数据是由视频提取得到的，针对我们的任务我们还应该希望打乱顺序的同时，同一条视频提取的数据仍然成组，我们可以用 DataFrame 的 groupby 功能对其进行分组，而后进行组间顺序随机打乱。

```
Python
# 依据第一列的文件名字符按照视频来源分组后打乱组间顺序
def df_shuffle(df):
    df['Group'] = df['Path_Filename'].str.slice(start=0, stop=-8)
    grouped = df.groupby('Group')
    # 对分组的索引进行随机打乱，但不改变组内顺序
    indices = grouped.groups.keys()
    indices_shuffled = pd.Series(indices).sample(frac=1).tolist()
    # 使用打乱后的索引重构 DataFrame
    df_shuffled = pd.concat([grouped.get_group(group) for group in indices_shuffled], ignore_index=True)
    # 删除临时的'Group'列
    df_shuffled.drop(columns='Group', inplace=True)
    return df_shuffled
df = pd.read_csv('./ workflow_pose.csv') # 指定原数据
df2 = df_shuffle(df) # 按组打乱顺序
df2.to_csv('./workflow_pose_shuffled.csv', index=False) # 保存为新的数据集，每次均为随机乱序，如重做此步骤建议每次都重命名
```

接着可以进行 CSV 格式数据集划分，即将数据集拆分为训练集和验证集，训练集用于训练模型，验证集用于评估模型的性能。参考代码如下，指定 csv 文件路径以及划分比例，将特征数据集划分为训练集和验证集。

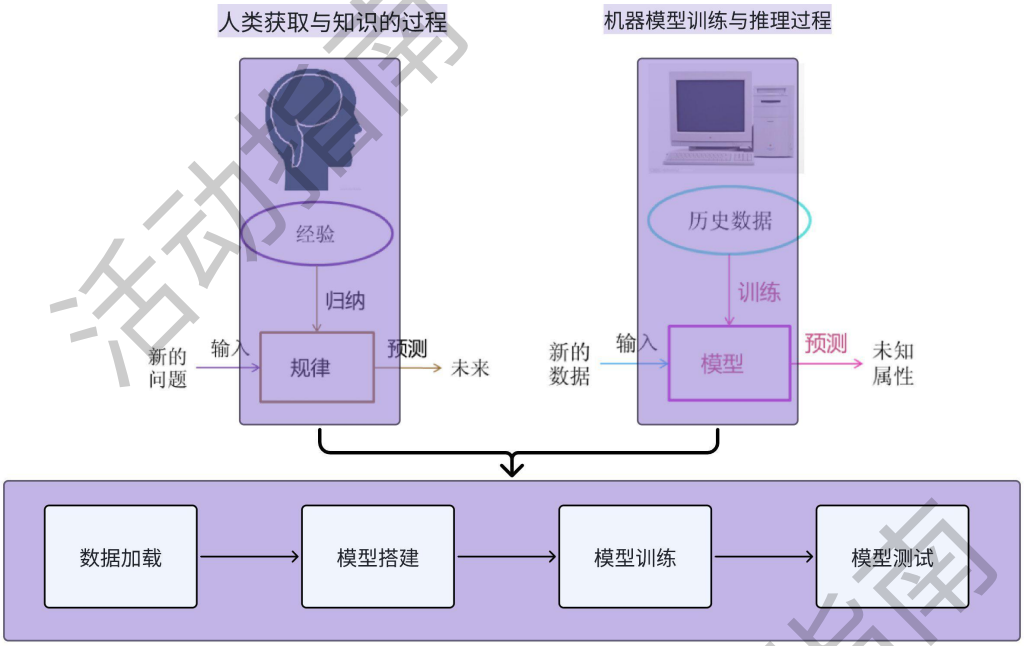
```
Python
# 划分训练集和验证集
ratio = 0.8 # 指定划分比例
df2 = pd.read_csv('./ workflow_pose_shuffled.csv')
df3 = df2.drop(columns='Path_Filename')
df3[:int(len(df3)*ratio)].to_csv('./ workflow_pose_train.csv', index=False)
df3[int(len(df3)*ratio):].to_csv('./ workflow_pose_val.csv', index=False)
```

四、模型训练

模型训练的任务是要让计算机能够识别和分类一段视频中的动作。前面我们对数据进行了简化（将视频数据转化为关键点数据），并完成了数据集的划分和制作，接下来可以使用机器学习方法进行模型训练。

4.1 模型训练的概念

“模型”是一个数学概念，用数学公式、方程等方式来刻画一个问题或事实。在人工智能中，模型使用参数来刻画，模型的参数在解决不同的问题时也有所差异。模型训练是指在已知数据中学习和总结数据蕴含的规律的过程，利用大量数据进行模型参数求解。



模型训练的目标是使得模型能够更好地解决实际问题，在完成模型训练后，需要检验模型的效果。对于未达标的模型需要继续训练，不断优化模型参数，直至得到满意的结果。

4.2 常见的人工智能开发工具

常见的机器学习框架主要有以下 4 种：

（1）scikit-learn。一款用于数据挖掘和数据分析的工具，提供了大量的机器学习算法，如聚类、分类、回归等，并支持模型选择和预处理等任务，具有简单高效的特点。

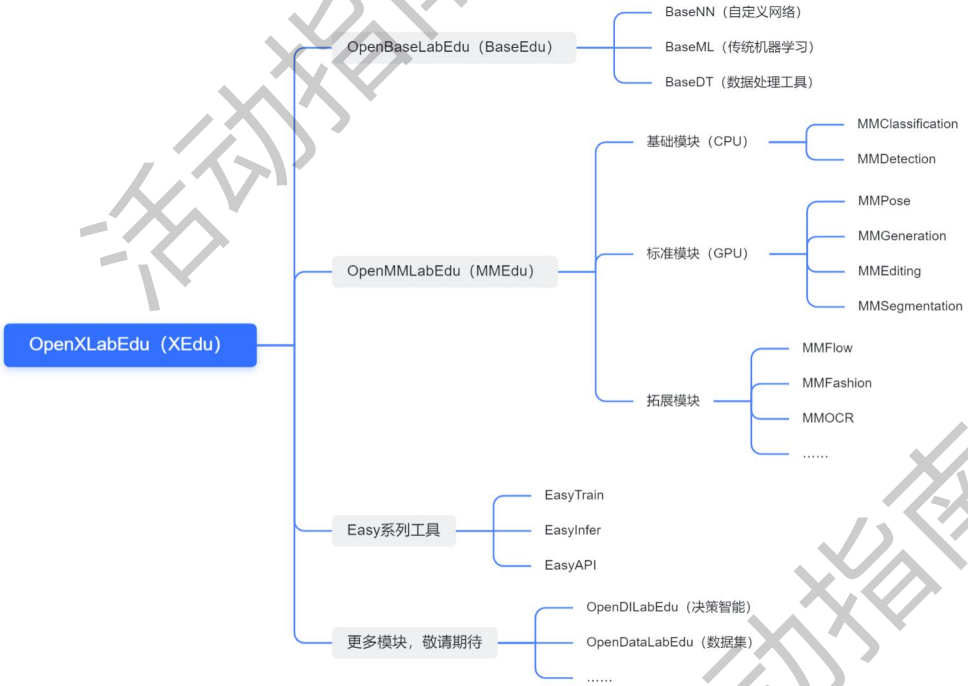
（2）TensorFlow。一个由 Google 开发的开源软件库，用于数值计算，常用于是深度学习应用。

（3）Keras。一个用 Python 编写的开源神经网络库，可以作为 TensorFlow、

Theano 和 CNTK 的高级接口，以简化神经网络的构建和实验。

（4）PyTorch。一个由 Facebook 开发的开源机器学习库，因其能够动态计算图和易用性，受到研究人员的欢迎。

然而，使用上述框架搭建的项目代码往往复杂难懂，更难以后期修改。因此，我们推荐基于 PyTorch 和 scikit-learn 封装的工具箱——XEdu。XEdu 是一款面向中小学 AI 教育的开发和学习工具，对人工智能算法进行了降维处理，通过简洁的代码就能实现人工智能模型的训练与应用。XEdu 提供了传统机器学习库 BaseML、神经网络库 BaseNN、计算机视觉库 MMEdu、深度学习工具集 XEduHub 等工具。



XEdu 提供的 Python 库功能丰富，简单易用，是一款适合中小学生入门的人工智能开发工具。它在精简代码的同时体现了 AI 模型训练和推理的流程化思维，支持学生编写出可以“真正运行”的 AI 代码。

4.3 BaseNN 与神经网络

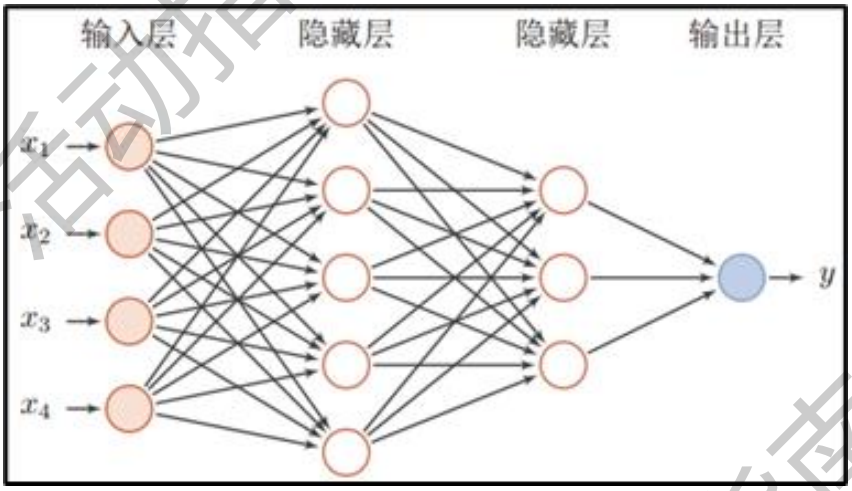
下面介绍如何基于关键点检测的数值型数据训练 AI 模型。前面我们准备好了数值型的表格数据，接下来可以利用各类机器学习方法（如 KNN、SVM 等），或者神经网络方法完成模型训练。针对上述两种方法，XEdu 提供了传统机器学习库 BaseML 和神经网络库 BaseNN 等相关工具。

BaseNN 可通过搭建神经网络、设定训练超参数并得到训练模型。BaseNN 不

仅支持五禽戏动作识等分类任务，还能支持回归任务和生成任务等。

那么，什么是神经网络呢？20 世纪 80 年代，人工神经网络成为人工智能领域的研究热点，它从信号传递的角度对人脑神经元网络进行仿生建模，被称为神经网络。神经网络由大量的神经元节点相互连接构成，每个连接像人脑中的突触一样，可以向其他神经元传递信号。多层神经网络的出现，特别是隐藏层的引入，使得网络具有更强大的潜力，增强了神经网络的非线性表达能力。近年，人工神经网络的研究不断深入，解决了模式识别、智能机器人、自动控制、预测估计、生物、医学、经济等领域的许多难题，取得了较大的进展。

如果你想进一步学习神经网络的相关知识，可以通过 tensorflow 工具（playground.tensorflow.org）进行体验。



4.4 模型搭建和训练

接下来让我们使用 BaseNN 来搭建神经网络训练模型吧！再观察一下我们获得的数值型数据集，可以发现它有 26*2=52 个输入变量，因此设定输入神经元的个数为 52。隐藏层的神经元个数可以自定义，这里设置第一个隐藏层神经元个数为 256，第二个隐藏层神经元个数为 64，最终分类类别数量为 10，故输出层神经元个数设定为 10。此外，设置学习率为 0.0001，训练轮次为 1000 轮。参考代码如下：

```
Python
from BaseNN import nn # 导入库
model = nn('cls') # 模型实例化
model.load_tab_data('wuqinxi.csv') # 指定训练数据集
model.add('linear', size=(52, 256), activation='relu') # 搭建神经网络输入层，输入神经元
```

```
的个数为 52
model.add('linear', size=(256, 64), activation='relu')   # 搭建隐藏层，神经元个数和层数可自行设定
model.add('linear', size=(64, 10), activation='softmax') # 搭建输出层，指定神经元个数为 10
model.train(lr=0.0001, epochs=1000) # 设定训练超参数学习率、训练轮次等
```

在训练模型的过程中，系统会输出各轮的损失值和准确率，我们可以根据这些情况来判断设置的超参数是否合适，以及评估模型的精度是否达到要求。

```
{epoch:994  Loss:2.0165  Accuracy:0.4418}
{epoch:995  Loss:2.0059  Accuracy:0.4536}
{epoch:996  Loss:2.0136  Accuracy:0.4454}
{epoch:997  Loss:2.0012  Accuracy:0.4558}
{epoch:998  Loss:2.0005  Accuracy:0.4597}
{epoch:999  Loss:1.9998  Accuracy:0.4620}
保存模型中...
保存模型checkpoints/BaseNN0410/basenn.pth成功!
```

4.5 模型评估和优化

模型训练过程中输出的 Loss 和 Accuracy 可为模型优化提供有价值的参考。Loss 随着 epoch 减小，而 Accuracy 在提升，这代表模型效果越来越好，如果出现其他情况，情况可能不太理想。当然，此时的 Accuracy 仅是训练集上的准确率，模型评估应该使用没有参与训练的验证集。模型效果验证的参考代码如下：

```
Python
# 计算验证集准确率
def cal_accuracy(y, pred_y):
    res = pred_y.argmax(axis=1)
    tp = np.array(y)==np.array(res)
    acc = np.sum(tp)/ y.shape[0]
    return acc

model = nn('cls') # 声明模型
checkpoint = 'checkpoints/BaseNN0408/basenn.pth' # 现有模型路径

# 读取验证数据
val_path = 'feature_data/workflow_pose_val.csv'
x_val = np.loadtxt(val_path, dtype=float, delimiter=',',skiprows=1,usecols=range(0,52))
y_val = np.loadtxt(val_path, dtype=float, delimiter=',',skiprows=1,usecols=52) # 读取最后一列，标签
```

```
result = model.inference(x_val, checkpoint=checkpoint) # 直接推理
acc = cal_accuracy(y_val, result)
print('验证集准确率: {:.2f}%'.format(100.0 * acc))
```

验证集准确率：32.47%

从上面的验证效果来看，模型在验证集的效果不如在训练集上的好，这属于正常现象。为解决以上不足，除了在数据预处理的环节增强数据，我们也可以使用一些模型训练优化策略来提升模型的效果，例如修改网络模型和超参数。其中，修改网络模型是指修改神经网络的结构，例如增加隐藏层的层数和神经元的个数、修改激活函数（activation）的类型、修改优化器的类型等。修改超参数是指修改训练的轮次、修改学习率的大小、修改使用的预训练权重文件等。为了保证网络模型和超参数修改前后的可比性，我们通常指定随机数种子，以确保伪随机保持不变，使得训练过程可复现。参考代码如下：

```
Python
model.add(..., activation='relu') # activation 为激活函数类型，可选值包括 ReLU，Softmax，tanh，sigmoid，leakyrelu
model.add(optimizer='SGD') # 指定优化器的名称
checkpoint='checkpoints/basenn.pth' # 指定已有模型的权重文件路径
model.train(lr=0.01, epochs=1000, checkpoint=checkpoint) # 模型基于一个已有的模型继续训练
```

如若了解 basenn 更详细的使用方法，可以参考官方文档的说明：

https://xedu.readthedocs.io/zh/master/basenn/introduction.html

小思考：

除了使用全连接神经网络，模型训练是否可以用其他机器学习方法来实现呢？你能否使用 BaseML 相关工具完成模型训练并对比训练效果？

为便于学生了解从数据预处理到模型训练与推理的全流程，主办方制作了一个项目供学生参考，项目地址预计五月初发布。

附录 4-4

“‘五禽戏’动作识别”项目提交指引

完成五禽戏动作的分类方案后，就可以准备提交项目结果啦。本篇文档讲解项目提交前的相关准备工作——利用训练的模型进行测试集推理并生成结果文件，并介绍继续优化模型或者尝试其他模型训练方案的相关方法。

1. 模型推理与结果提交

经过前面的数据准备和模型训练，此时我们至少可以得到 2 个模型文件。接下来需要利用训练的模型对测试集中的所有视频都进行推理，并按照相关要求编写 CSV 文件的代码，将结果写入一个 CSV 文件中，便于评委复现且能快速完成测试 B 集推理。

1.1 模型推理的概念

模型推理是指将训练好的人工智能模型应用于新的数据以做出预测或决策的过程。在这个阶段，模型将利用在训练阶段学到的知识，对新输入的数据进行分析，输出预测结果。根据前面选择的模型训练路径，模型推理的主要流程如下：

在实际应用中，模型的运作可以被简化为一个直观的过程：用户仅需输入新的数据，模型便会自动进行复杂的计算并输出预测结果。这种操作方式无需用户深入了解模型内部的复杂机制和算法细节，只需关注如何准备和提供合适的输入数据。由此，即使非专业人士也能够便捷地利用模型来解决实际问题，极大地扩展了人工智能技术的可用性和普及度。

1.2 模型推理初体验

先来测试一下我们训练的模型吧，注意保持输入数据和训练集数据一致。

```python
from BaseNN import nn
model = nn('cls') #  声明模型
checkpoint = 'basenn.pth' #  指定现有模型路径
```

```
data = [[34.69628906, 52.27832031, 40.66308594, 53.65527344, 32.63085938,
         52.04882813, 47.54785156, 42.18066406, 31.02441406, 43.09863281,
         50.99023438, 36.21386719, 22.53320313, 36.90234375, 46.62988281,
         52.73730469, 15.41894531, 55.49121094, 41.3515625 , 72.93261719,
         16.33691406, 74.99804688, 31.71289063, 14.87109375, 14.04199219,
         15.33007813, 36.30273438, 48.37695313, 14.04199219, 48.8359375 ,
         38.59765625, 74.76855469, 15.87792969, 75.91601563, 41.12207031,
         50.44238281, 38.13867188, 33.45996094, 23.22167969, 13.953125    ,
         42.26953125, 86.47265625, 16.10742188, 86.93164063, 45.94140625,
         85.09570313, 10.59960938, 86.01367188, 33.77832031, 77.52246094,
         22.53320313, 79.81738281]]   # 指定一组新数据
result = model.inference(data, checkpoint=checkpoint) # 直接推理
model.print_result(result) # 输出推理结果
```

1.3 模型格式转换与应用

模型的格式决定了它是否能在不同的平台和环境中运行，例如.pth 文件是 PyTorch 的模型保存格式，.pb 文件是 TensorFlow 的模型保存格式，文件中都包含了模型的结构和参数。

为了能够在多个平台上进行模型部署，我们建议通过模型转换，将模型格式转换为通用格式，如.onnx 格式的模型。ONNX（Open Neural Network Exchange）是一个开放的深度学习模型交换格式，目标是使得不同的深度学习框架（如 PyTorch、TensorFlow、Keras 等）可以共享模型，从而使得模型的训练和推理更加灵活和高效。目前 BaseNN 已经内置了模型转换函数，可直接支持模型转换。

BaseNN 模型转换的示例代码如下：

```
Python
from BaseNN import nn
model = nn('cls')
model.convert(checkpoint="basenn_cd.pth",out_file="basenn_cd.onnx")
```

当模型转换为 ONNX 格式后，可以直接使用 XEdu 的 XEduHub 库完成推理。XEduHub 库支持经 BaseNN 转换的 ONNX 模型的推理，且没有过多的依赖库，安装速度快速。

使用 XEduHub 完成 BaseNN 模型推理应用的示例代码如下：

```
Python
```

```
from XEdu.hub import Workflow as wf
basenn = wf(task="basenn",checkpoint="basenn.onnx")# 指定使用的 onnx 模型
result = basenn.inference(data=data)# 进行模型推理
format_result = basenn.format_output()
```

1.4 完整推理的程序

前面我们进行了和训练集数据一致的单条数据的测试和应用，接下来需要考虑如何处理测试集中存放的多条待标注的视频，以完成所有视频的推理并将写入结果文件中。这一过程涉及测试集数据处理、模型推理，以及视频读取、批量处理、CSV 文件读写等步骤。同时模型推理速度纳入活动评分范畴也需考虑。

1.4.1 推理流程分析

需要强调的是，每条视频的推理需要经过图像帧提取、关键点检测、维度处理等操作，以获得与训练时相同的输入维度。此外，完成测试集中所有视频的推理后，需要将视频分类结果写入一个 CSV 文件中。

（1）单条视频推理

在测试模型时，可以利用一组数据模拟关键点提取结果，推理出动作类型。因此要实现单条视频推理需要将新视频也转换为关键点数据，以真正实现某帧图像帧的推理。

（2）批量推理

在单条视频推理的基础上，需要批量处理测试集中所有视频数据。

（3）生成结果文件

得到每条视频的分类结果后，将其写入一个 CSV 文件中。

1.4.2 思路点拨

（1）维度处理

不同的模型可能会要求不同的数据处理方法，因此在进行模型推理时，需要核对新数据的维度与模型输入维度是否一致（参照模型训练时的数据处理方式）。

例如，在模型训练前，通过批量关键点检测得到了每行 52 特征维度的关键点数据集，那么在模型推理时也需对事先提取的图像帧作同一种关键点检测。在对单张图片进行关键点检测时，生成的是一个包含多组关键点[x,y]坐标信息的二维数组，所以检测后还需经过展平操作 pose_features = np.concatenate(keypoints_list).reshape(len(keypoints_list), -1)，将其调整为一维数组。此外，如果新的数据

在原有特征的基础上增加了新的复合特征，我们也要对新的图片或视频进行同样的维度处理，保证模型输入维度一致。

（2）图像帧提取

在模型训练前，我们进行了图像帧提取，模型推理阶段也要进行类似操作，但是提取的策略有所不同。在提取图像帧时，选取帧的多样性和信息含量会对模型的性能产生重要影响。我们可以运用先前学习的图像帧提取策略和实现方法，对原有函数进行适当的调整（不必拘泥于与训练集完全一致的抽帧策略），但必须确保所采用的方法能够有效提取出具有高度代表性和丰富信息的图像帧，同时兼顾模型的推理效率。最为简单的例子就是从视频中抽出一帧图像并推理出该图像所属的类别，比如提取视频中间的一帧。

（3）CSV 文件读写

使用 NumPy 库从 CSV 文件中读取数据只需要一行代码，下面这行代码的作用是从一个 CSV 文件中读取第一列数据（跳过标题行）。

```
Python
video_path_list = np.loadtxt(csv_name, delimiter=",", skiprows=1, usecols=[0], dtype=str)
```

参数说明：

“csv_name”是指要读取的 CSV 文件的名称或路径；

“delimiter”是指文件中数据的分隔符，CSV 文件中常用逗号（“,”）作为分隔符；

“skiprows=1”表示跳过第一行，值为 0 表示不跳过；

“usecols”是指从 CSV 文件中读取哪些列，例如“[0]”表示只读取第一列的数据；

“dtype”是指返回数组的数据类型；

如果你想继续探索 NumPy 库相关知识，可通过“NumPy 快速入门教程（中文）（https://www.numpy.org.cn/）”进行体验。

1.4.5 参考程序

在活动的配套项目中，主办方提供了模型推理和生成结果文件的程序，学生可以直接使用。程序功能包含遍历测试集的所有视频，并对视频进行相应的数据处理，以及按照要求将所有视频的推理结果写入 CSV 文件中。

```python
import cv2, os, csv
import numpy as np
from XEdu.hub import Workflow as wf

# 定义图像帧提取的函数
def get_frame_from_video(video_path):
    '''
    函数提取视频的中间帧并返回最终的图像
    '''
    cap = cv2.VideoCapture(video_path)
    # 获取视频总帧数
    total_frames = int(cap.get(cv2.CAP_PROP_FRAME_COUNT))
    frame_number = int(total_frames // 2)
    print('一共有',total_frames,'帧，现在读取第',frame_number,'帧。')
    cap.set(cv2.CAP_PROP_POS_FRAMES, frame_number)
    ret, frame = cap.read()
    cap.release()
    return frame

# 定义提取关键点的函数
def get_keypoints_from_frame(det_model, pose_model, image_path):
    bboxs = det_model.inference(data=image_path)
    keypoints_list = []
    if len(bboxs)>0:
        keypoints = pose_model.inference(data=image_path, bbox=bboxs[0]) # 提取检测出的第一个人的关键点
        keypoints_list.append(keypoints)
    if len(keypoints_list)>0:
        # 展平数组
        keypoints_list = np.concatenate(keypoints_list).reshape(len(keypoints_list), -1)
    return keypoints_list

# 定义分类模型推理的函数
def get_cls_result(model, data):
    result = model.inference(data=data)
    format_result = model.format_output(lang='zh')
```

```
    res = []
    for key in format_result:
        res.append(format_result[key]['预测值'])
    return res[0]


# 设置路径和模型参数
csv_path = 'test_video/filename_list.csv'
# 读取视频文件名列表
video_path_list = np.loadtxt(csv_path, delimiter=',', skiprows=1, usecols=[0], dtype=str)

# 初始化关键点提取和分类模型
det_model = wf(task='det_body')
pose_model = wf(task='pose_body26')
cls_model = wf('basenn',checkpoint='checkpoints/BaseNN0410/basenn_cd.onnx')

# 存储所有推理结果
predictions = []
 # 遍历测试集视频文件名列表
for video_path in video_path_list:
    # 提取视频帧
    frame = get_frame_from_video(os.path.join('test_video/', video_path))
    print(f'正在处理视频：{video_path} ...')
    # 提取关键点
    keypoints_list = get_keypoints_from_frame(det_model, pose_model, frame)
    # 分类模型推理
    if len(keypoints_list)>0:
        final_result = get_cls_result(cls_model, keypoints_list)
        # 输出该视频分类结果
        print(f'推断 {video_path} 的姿势序号为 {final_result}')
        predictions.append(int(final_result))
print('所有推断结果如下：', predictions)

# 指定生成的 csv 文件，可修改路径和名字
result_csv_path = 'submission.csv'

# 设定标签
```

```
label = ['0_TigerLift', '1_TigerPounce', '2_DeerButt', '3_DeerGallop', '4_BearSwing',
            '5_BearWave', '6_MonkeyLift', '7_MonkeyPick', '8_BirdStretch', '9_BirdFly']

with open(result_csv_path,"w", newline='') as csvfile: # 将预测结果写入新的 csv 文件，注意命名
    csv_writer = csv.writer(csvfile)
    csv_writer.writerow([f'filename','prediction', 'pre_class'])
    for index,image_name in enumerate(video_path_list):
        if predictions[index] is not None:
            csv_writer.writerow([image_name, predictions[index] , label[predictions[index]]] )
        else:
            csv_writer.writerow([image_name])
    csvfile.close()
```

学生可对以上代码进行调整，相关说明如下：

（1）模型调整

对于使用了不同训练工具的同学，主要调整 def cls_inference 处的推理代码，并导入 import 对应的库，指定自己训练的模型路径。在完成模型推理后，注意检查输出结果 predictions 的格式是否与预期一致。

（2）测试集调整

学生需按照要求提供一个 inference.py 文件，评委将会调试该程序，将测试集路径修改为测试集 B 集，并根据该程序的运行结果进行最终评分。

为了便于复制出要提交的程序，我们已经将教材中提及的有关模型训练和推理的代码存放在了项目中的完整训练推理代码.ipynb 文件中。在提交项目结果前，我们也可以自行**调试**这个文件，修改测试集的路径，并对模拟的测试文件夹进行推理与调试，避免可能出现的各类报错。

1.5 结果提交与查看

至此，你应该已经获得了一个符合要求的 submission.csv 文件！那么可以去提交你的 A 榜作品了，按照视频步骤在测评平台（届时发布于报名页面）上传你的结果文件。如果你还没有报名信息的话，抓紧完成报名并获取报名信息，完成填写即可获取你的 A 榜成绩啦。

如果你对 A 榜成绩不够满意，可以仔细回顾前面学习过的知识点，想办法

提升你的成绩，也可以直接去项目提升区继续学习进阶知识，加油！

2.算法优化与项目提升

在模型的开发过程中，算法优化是提高模型性能的关键环节。以下提供了几种重要的优化策略，可以帮助我们提升模型的效果，并确保模型在实际应用中的准确性和可靠性。同时，展示几种其他动作识别实现路径的策略。

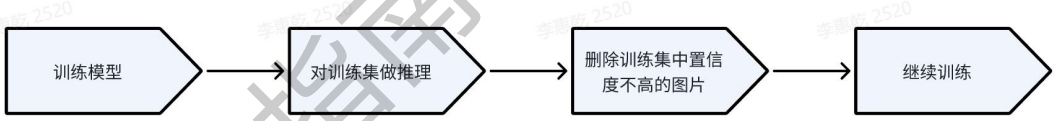
2.1 数据优化

2.1.1 数据集清洗

数据集的质量是构建高效模型的基石，对模型训练效果有着决定性的影响。原始数据集中可能包含一些低质量或无关的数据，如在视频截取过程中可能不小心包含了背景中的无关人员、图像帧提取得到的图片为休息动作、csv 数据没有与标签相对应或存在错误的数值等，这些数据都会对模型训练产生负面影响。也就是，如果原始视频数据的质量较低，那么模型训练效果势必不不够理想。

我们可以通过人工浏览数据集，粗略检查和排除问题数据，例如**精细剪辑视频、挑选图片**等，从而提高数据集的整体质量。需要注意高质量的数据集是模型训练成功的前提，投入在数据清洗上的时间和精力将在模型性能上得到回报。

此外，我们也可以使用自己训练的模型，将训练集的数据也推理一次，查看哪些图片的“置信度”特别低。如果这些图片果真存在分类错误的问题，那么可以断删除掉，并重新训练模型。上述方法通过选择继续训练（迁移学习）即可实现，具体工作流程如下图所示。



另外，我们也可以直接对 CSV 格式的训练集（训练模型时载入的数据）进

行模型推理，删除那些置信度低的数据行。参考代码如下：

```python
import numpy as np
from BaseNN import nn

# 加载分类模型，此处使用的是 BaseNN 训练的模型
model = nn('cls')
checkpoint = 'checkpoints/BaseNN/basenn.pth'

# 定义置信度阈值
confidence_threshold = 0.7
# 指定训练集
train_path = 'data/workflow_pose_train.csv'

# 首先，读取 CSV 文件的标题行
with open(train_path, 'r') as file:
    header = file.readline().strip()  # 读取第一行即标题行

# 使用 numpy 读取 CSV 文件，skip_header=1 跳过标题行
data =  np.loadtxt(train_path, dtype=float, delimiter=',',skiprows=1)

# 用于存储要保留的行
to_keep = []

# 对数据中的每一行进行处理
for index, row in enumerate(data):
    # 提取除最后一列标签列以外的所有特征列
    features = np.array(row[:-1], dtype=float)
    # 如有其他处理可继续写入

    # 进行模型推理
    result = model.inference(features.reshape(1, -1), checkpoint=checkpoint)
    # 输出推理结果
    res = model.print_result(result)
    # 获取置信度
    confidence = res[0]['置信度']
```

```
        # 检查置信度是否高于阈值
        if confidence >= confidence_threshold:
            to_keep.append(index)

# 保留置信度高的行
data_kept = data[to_keep, :]
# 将数据保存到新的 CSV 文件中
np.savetxt('new_dataset.csv', data_kept, delimiter=',', fmt='%s', header=header,
comments='')
```

2.1.2 特征提取优化

特征提取是机器学习中的重要步骤，好的特征可以显著提升模型的性能。例如，确定将五禽戏动作的手部姿态作为重要分类依据后，可以使用 XEduHub 中的 pose_wholebody144 模型提取关键点。

（1）数据增强-视频滑动窗口

在实际应用中，用于推理的视频可能并不总是理想地从动作的起点开始，或者由于各种原因我们无法仅通过特定的帧数来判断视频的类别。这种情况下，视频增强技术就显得尤为重要，它可以帮助模型更好地理解和识别视频中的动作。

通过在原始视频序列上应用一个“窗口”，沿着视频的时间轴滑动，可以截取不同时间段的连续帧作为新的视频片段。通过改变窗口的大小和滑动的步长，我们可以从同一视频源生成多个不同长度和内容的视频片段。

例如，针对一个包含了一系列动作的长视频，我们可以设置一个 5 秒钟的窗口，并以 1 秒的步长移动这个窗口。这样我们就可以从原始视频中生成多个 5 秒长的视频片段，每个片段都包含不同的动作序列。

（2）数据归一化

归一化是数据预处理中的一种常用技术，目的是将数据按比例缩放，使之落入一个特定的小区间，通常是[0, 1]或[-1, 1]。这样可以消除不同特征之间的量纲影响，使得数据更加平稳，算法的学习和处理。

例如，在处理包含熊猫和人体的图像时，因为熊猫和人体的身体比例差异较大，如果不进行归一化处理，可能会导致模型对某些特征（如特定身体部位的距离）过于敏感，而忽略了其他重要的特征。通过归一化处理，可以确保模型对所有特征都给予适当的重视，从而提高识别的准确性。

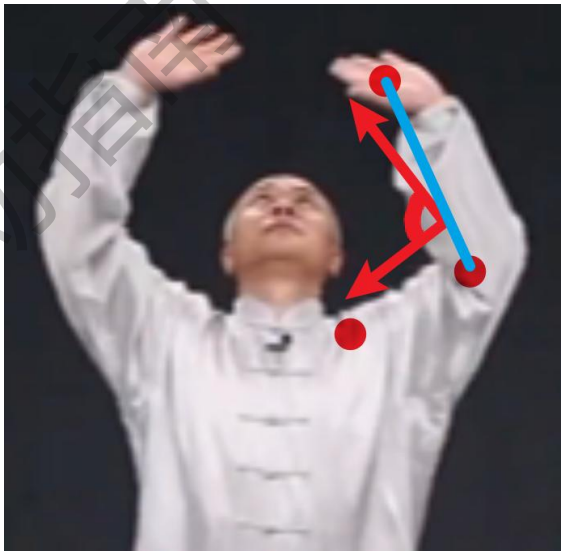
值得注意的是，如果在模型训练时对数据进行了归一化处理，在测试集推理时也需进行同样的归一化处理操作。Pandas 库提供了简便的数据处理流程，可以方便地实现数据归一化，参考代码如下：

```python
import pandas as pd
import numpy as np

df = pd.read_csv('./workflow_pose_val.csv')

headers = df.columns
arr = df.values

# 对每列特征进行归一化，忽略最后一列类别标签

for i,pose_features in enumerate(arr[:,:-1]):
    arr[i,:-1] = (pose_features - pose_features.min()) / (pose_features.max() - pose_features.min())

# 写入新的 CSV 文件

df = pd.DataFrame(data=arr,columns=headers)
df.to_csv('./normalized_workflow_pose_val.csv', index=False)
```

（3）增加复合特征

除了人体识别得到的坐标点，我们还可以增加复合特征，如通过计算关键点间的角度或距离，让计算机更好地理解不同动作间的差异。例如，通过计算坐标点，计算某个关节的角度；通过坐标计算两个部位间的横坐标距离（x1-x2）以及直线距离 $\sqrt{((x1-x2)^2+(y1-y2)^2)}$ 。

计算得到这些特征后，将其加入待识别的 csv 文件或图像中，调整模型的输入维度，神经网络就会自动学习新的特征，并根据其对预测结果的重要性与相关性调整网络内部的权重。



2.2 算法优化

在深度学习中，我们可以设置一些对训练策略的约束，这些设置并不直接决定参数，因此我们常把它们称为“超参数”。这个过程可以类比为在阅读一本书时需要做出的一些选择，如阅读速度、阅读顺序等，这些选择会影响我们理解和掌握书中的内容。与超参数类似，这些选择需要我们自己决定，并且需要根据自己的经验和目的来进行调整以获得最佳的阅读效果。

2.2.1 优化学习结果

“过拟合”发生在模型对训练数据学习过度的情况下，表现为噪声和异常值，会导致模型对新数据的泛化能力下降。“过拟合”通常由于模型过于复杂、参数多于数据信息量所导致。解决“过拟合”现象可以采用正则化、交叉验证、扩充数据集或简化模型等策略。

“欠拟合”是由于模型过于简单，未能捕捉数据的关键模式，导致模型在训

练集和新数据上都表现不佳。通过提升模型复杂度、增加数据量或训练时间，以及使用更复杂的模型结构有助于缓解“欠拟合”问题。

模型的训练轮次对避免欠拟合和过拟合至关重要。设置合理的轮次可以确保模型在训练过程中既能够充分学习数据的特征，又不至于对训练数据过度学习。我们可以通过观察系统输出的 loss 值，及时调整学习率、优化器和训练轮次，找到最佳的训练策略，以最大化模型的性能。

2.2.2 优化 BaseNN 自建网络

如果选择自建神经网络进行模型训练，我们需要根据问题的特性来设计网络结构，包括选择合适的层数和激活函数。

（1）层数。对于简单的图像分类任务，可以从 3-5 层卷积层（Convolutional Layers）开始尝试。对于更复杂的图像，如高分辨率图像或更复杂的分类任务，需要采用更深的网络，如使用 ResNet50 等预训练模型。

（2）激活函数。ReLU（Rectified Linear Unit）是最常用的激活函数，用于卷积层和全连接层。Softmax 激活函数通常用于输出层的多类分类问题。

（3）正则化技术。正则化是用于防止过拟合，提高模型泛化能力的一系列方法。过拟合发生在模型对训练数据学习过度，以至于模型在新的数据上表现不佳。应用正则化技术（如 L1、L2 正则化或 Dropout）可以减少模型过拟合的风险。这些技术通过在模型训练过程中添加一定的约束或者通过随机丢弃神经网络中的一部分连接，来提高模型的泛化能力。

以下是利用 BaseNN 库解决猫狗分类问题的网络搭建示例。

```
Python
from BaseNN import nn
```

```
model = nn()
model.load_img_data('CatsDogs/training_set', batch_size=32)
model.add('mobilenet_backbone') # MobileNet 主干网络
model.add('Linear', size=(1280,1000), activation='relu')
model.add('Dropout', p=0.2)
model.add('Linear', size=(1000,2),activation='Softmax')
model.add(optimmizer='Adam')
model.save_fold = 'mobilenet_ckpt'
model.train(lr=1e-3, epochs=20,metrics=['acc']) # 模型训练
```

2.3 推理结果优化

前面我们提到可以在测试集中使用不同的抽帧方案，接下来让我们了解更进一步的算法思路。例如，某位同学在训练阶段选择了每隔一定时间抽取 5 帧图像进行训练，所训练出的模型能够根据单帧图像输出其对应的类别；而在推理阶段，他决定在测试集中的每个视频中等间隔地抽取 20 帧图像用于推理。这样做相当于为每个视频提供了更多的评估点，虽然牺牲了一定的速度，但提高了分类的精确度。另一方面，为了加快处理速度并提升推理代码的执行效率，有些同学可能会选择在每个视频中仅抽取 3 帧图像，还可能会移除提取函数中的保存功能，以便直接返回图像数据，从而进一步提高处理速度。

在模型推理时，如果从视频中抽取了多帧图像，如何综合各图像帧推理结果判断出这个视频的分类呢？比较好理解的方式是“投票”，通过比较每帧图像帧或每组关键点数据的推理结果，找到出现频率最高的类别，确认视频的分类结果。借助 Python 标准库中的 collections.Counter 类，能够计算元素在列表中出现的次数，示例代码如下：

```
Python
# 导入库
from collections import Counter
# 使用 Counter 来计算每个元素的出现次数
```

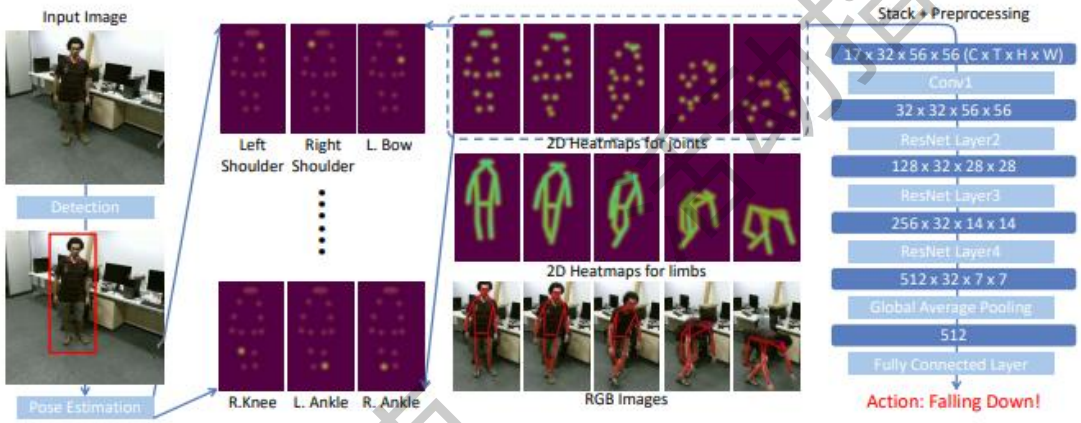
```
counter_results = Counter(all_results)
# 找到出现频率最高的元素
most_common = counter_results.most_common(1)[0]  # 返回格式 [(元素, 频率)]
most_common_class, frequency = most_common
print(f"出现频率最高的类别是 {most_common_class}，出现了 {frequency} 次")
```

首先，代码通过 Counter(all_results)创建了一个 Counter 对象，遍历结果后，内部结构可以看作一个字典，键是 all_results 中的元素，值是相应元素出现的次数。其次，counter_results.most_common(1)方法能获取出现次数最多的元素及其频率，参数为 1 表示只返回出现次数最多的一个元素，通过索引操作[0]获取最频繁出现的元素及它的频率。

2.4 其他优化策略

动作识别任务是一个较为复杂的计算机视觉任务。我们可以把视频问题简化为**图像问题**，利用卷积神经网络提取每个图像帧的特征，然后进行图像分类。更具有技巧性的做法是使用**关键点姿态分类**，例如视频方法、图像方法、关键点检测方法，或是上述方法的组合。

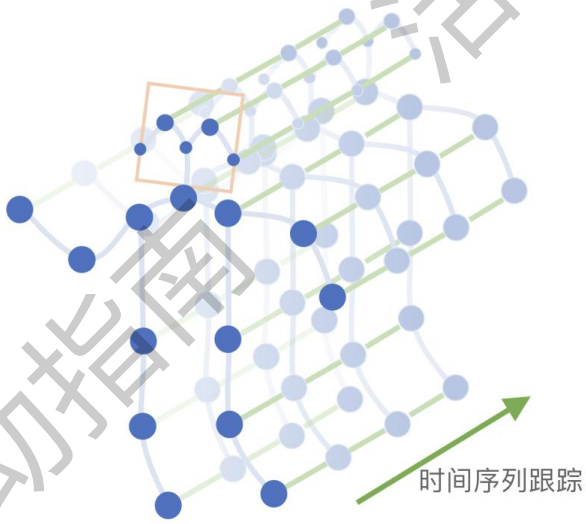
然而，前面的方法忽视了“动作”这一概念的时序特性。动作是有先后的，因此数据之间存在一定的关联。我们可以尝试将数据连接，将多帧图像提取的数据联合进行训练。循环神经网络、三维卷积和双流法等都是利用视频时序信息进行模型优化的好方法。



2.4.1 构建循环神经网络

循环神经网络（RNN）及其变种（如 LSTM 和 GRU）是一种特别适合处理序列数据的神经网络结构。在动作识别任务中，RNN 可以有效地处理时间序列数据，如视频帧或特征数据，从而识别和分类人体动作。它通过记住前面的信息，

并结合当前的输入，逐步构建出一个能够识别特定动作的模型。BaseNN 也支持构建简单的循环神经网络模型，支持的数据集格式为 npz 格式。活动为学生提供了一个搭建循环神经网络训练姿态识别模型的参考项目，该项目涉及了数据集制作、模型搭建、模型训练和应用的全流程，可以举一反三完成五禽戏动作识别。



项目地址：

https://openinnolab.org.cn/pjlab/project?id=64daed3eafeb1059822a1578&sc=635638d69ed68060c638f979#public

参考论文：https://arxiv.org/pdf/1801.07451.pdf

2.4.2 分类的细化

前面我们指出了图像分类数据集的类内差异较大，这是因为一个图片对应的动作有很多，并不是只有一种动作状态，由此导致无论是图片数据还是关键点数据，在同一个类别中的数据都不一定相同。因此，我们可以将动作分段，按照节拍将动作拆分为若干类别，例如“抬手”、“扭头”等，当一段视频按照一定顺序完成了前面规定的若干个动作，那么才判定为该动作。

这样，一方面细化了分类，减小了类内的差距，优化了识别效果；另一方面，能够适应动作快和动作慢的多种视频情况，实现更好的解决效果。具体来说，我们可以对视频进行更详细的标注，利用视频剪辑软件将每个细节动作切分，通过视频图像帧提取、关键点检测等步骤制作新的分类数据集并进行训练。

2.4.3 图像分类算法提升

三维卷积（3D Convolution）能够同时处理空间和时间信息的卷积操作，通

过在三维空间（宽度、高度和时间）上应用卷积核，能够捕捉到视频中的时空特征。与传统的二维卷积（只处理图像的宽度和高度）相比，3D 卷积增加了时间维度的处理，使得模型能够学习到动作随时间的演变。

参考项目：

https://openaccess.thecvf.com/content_iccv_2015/papers/Tran_Learning_Spatiotemporal_Features_ICCV_2015_paper.pdf

https://openaccess.thecvf.com/content_cvpr_2018/papers/Wang_Non-Local_Neural_Networks_CVPR_2018_paper.pdf

双流法（Two-Stream）也是一种在动作识别领域中非常流行的方法，它通过分别处理空间信息和时间信息来提高识别的准确性。这种方法的核心思想是，动作识别可以从两个不同的角度来理解，一个是空间角度，关注动作发生的物体的形状和外观；另一个是时间角度，关注动作随时间的变化。

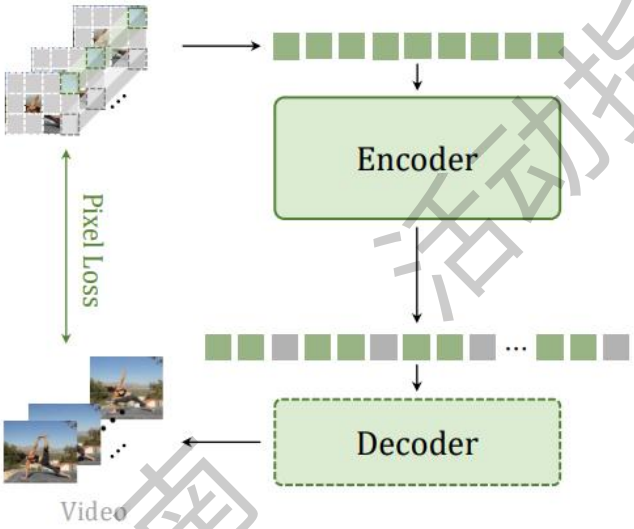
参考项目：

https://proceedings.neurips.cc/paper/2014/file/00ec53c4682d36f5c4359f4ae7bd7ba1-Paper.pdf

https://openaccess.thecvf.com/content_cvpr_2016/papers/Feichtenhofer_Convolutional_Two-Stream_Network_CVPR_2016_paper.pdf

2.4.4 大模型方法

视觉转换器（Vision Transformer，ViT）是一种结合了卷积神经网络（CNN）和 Transformer 架构的新型视觉模型，最初是为图像分类任务设计的，但也可以被应用于动作识别。ViT 模型主要包含卷积神经网络（CNN）和 Transformer 两个部分。CNN 负责从视频帧中提取空间特征，并将特征向量被送入 Transformer。Transformer 是一种基于自注意力机制的架构，能够处理序列数据，负责处理 CNN 传送的特征并捕捉它们之间的关系。



参考项目：

https://github.com/mx-mark/VideoTransformer-pytorch

https://github.com/sallymmx/ActionCLIP

除了上述方法，目前更多基于大模型的动作识别方案正在涌现，由于技术过于复杂，这里仅提供链接供学生参考：

（1）InternVideo2: https://github.com/OpenGVLab/InternVideo2

（2）Hulk: https://github.com/OpenGVLab/Hulk