# 姿势分类识别图片方案

2024 年 11 月 15 日

# 目录

# 1 数据集要求

## 1.1 图片类别要求

对于正常幼儿：拍摄婴儿自然的动作。

对于异常幼儿：尽量捕捉其异常动作。

每类尽可能多的采集不同幼儿的不同照片。

**总数要求**

正常类别：500-1000 张

异常类别：300-500 张（异常动作相对较难采集）

模糊类别：100-300 张（用作无效数据检测或异常检测模型）

**单个数量要求**

新生儿：100-200 张

婴儿期（3-6 个月）：200-300 张

学步期（7-12 个月）：300-500 张

幼儿期（1-2 岁）：300-500 张

分为**正常、异样、模糊**三个文件夹采集照片。

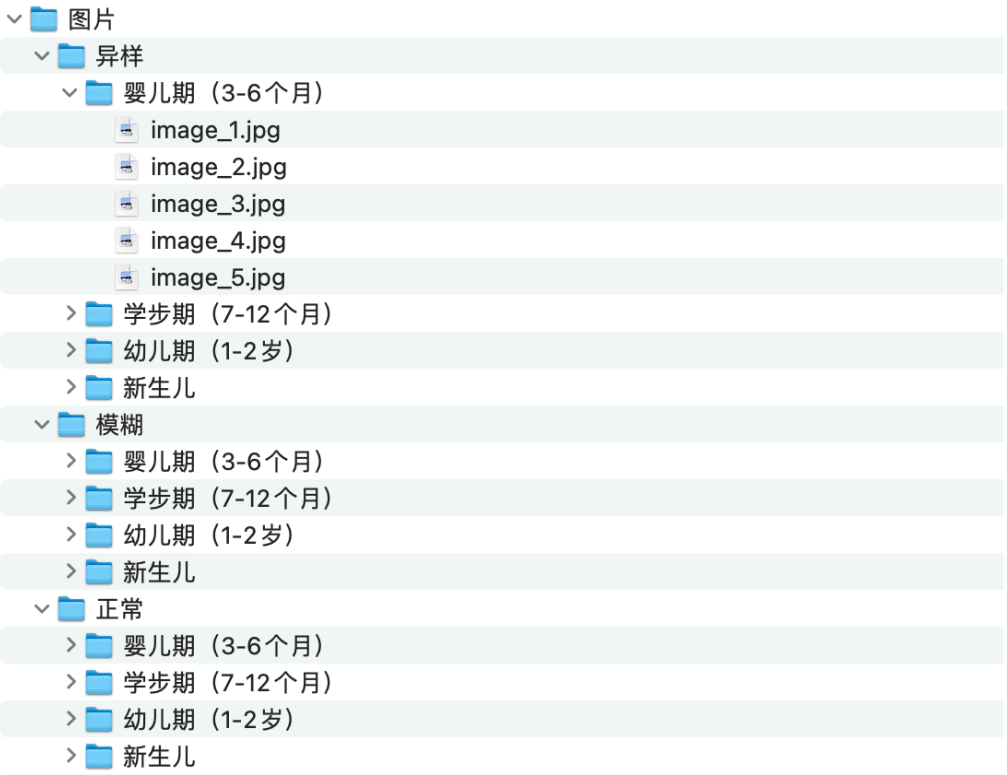


图 1: 结构示例

## 1.2 图片格式要求

1. 图片采用 **16:9** 的格式。

2. 分辨率推荐 **1080p (1920×1080)** 以上。

## 1.3 拍摄要求

### 1.3.1 环境要求

1. 背景应尽量简单，避免复杂的背景干扰。

2. 推荐使用**纯色背景**或室内环境，如床上、垫子上等。

### 1.3.2 光照条件

1. 图像应在良好的光照条件下采集，避免过暗或过曝。

2. 建议采集自然光和室内灯光条件下的图片，确保多样性。

### 1.3.3 婴儿穿着要求

1. **避免**穿着影响婴儿身体骨架特征的**厚衣服**。

2. 婴儿衣服颜色应与背景颜色形成**对比**。

### 1.3.4 婴儿姿势要求

数据集应该尽量包含以下姿势:

1. 睡姿（仰卧、俯卧、侧卧）

2. 爬行（手膝着地移动）

3. 坐姿（双手支撑、稳定坐姿）

4. 站姿（扶站、不扶站）

5. 躺卧翻滚（侧翻或转身）

### 1.3.5 拍摄视角要求

1. 多视角采集，覆盖从**上方、侧面、正面**等不同方向拍摄的图片。

2. 尽量保持婴儿**全身在画面中**，确保关键点（如肘关节、膝关节等）完整。

## 1.4 图片示例


图 2: 正面示例

图 3: 侧面示例

图 4: 顶面示例

# 2 数据预处理与特征提取

## 2.1 图片文件处理

使用 OpenCV 配合 MediaPipePose，生成带有骨架图的儿童图片，统一命名 img_xxx.jpg，统一分辨率以及图片大小。

## 2.2 骨骼数据提取（单摄像头）

使用 MediaPipe 进行人体 33 个关键点的提取，保存为 json 格式，内有 image 和 features 字段。image 字段代表图片的路径，包含图片的名字。features 字段包含 33 个关键点的 keypoint 信息。

## 2.3 数据格式（单摄像头）

建议提取数据放在数据标注操作之后，方便 json 数据的追加 features 属性。1. 单个关键点的数据格式：

```
Keypoint: {
    "keypoint_id": 0,
    "x": 0.5661032795906067,
    "y": 0.3272441029548645,
    "z": -1.3933179378509521,
    "visibility": 0.9999678134918213
}
```

2. 单个图片的数据格式（带有 33 个关键点）：

```
{
    "image": "train_img.png",
    "features": [Keypoint]
}
```

# 3 数据标注

## 3.1 工具采用

使用 doccano 工具进行 Classification 类型标注。使用 docker 部署，暴露 8000 端口，方便多个人员进行数据标注。

## 3.2 标签等级

可以选用 0-9 十个分类或者 a-z 的字母分类作为 label。每个图片可以有多个 label。

### 3.3 数据格式


```
{
    "id": 2,
    "filename": "img_57.jpg",
    "label": "Good",
    "Comments": []
}
```


## 4 数据合并

### 4.1 合并工具选择

使用 Python 的 json 模块进行 json 与 jsonl 数据的合并处理。

### 4.2 预期合并格式


```
Keypoint: {
    "keypoint_id": 0,
    "x": 0.5661032795906067,
    "y": 0.3272441029548645,
    "z": -1.3933179378509521,
    "visibility": 0.9999678134918213
}
{
    "id": 2,
    "filename": "img_57.jpg",
    "features": [Keypoint]
    "label": "Good",
    "Comments": []
}
```


### 4.3 代码片段

## 5 模型训练

### 5.1 工具选择

选用 Tensorflow 进行训练。

## 5.2 传入数据

```
def load_data(json_path):
with open(json_path, 'r') as f:
    data = json.load(f)

inputs = []
labels = []
for sample in data:
    features = sample["features"]
    # Flatten each (x, y, z, visibility) into a single array
    flattened_features = []
    for keypoint in features:
        flattened_features.extend([keypoint["x"], keypoint["y"],
           keypoint["z"], keypoint["visibility"]])

    inputs.append(flattened_features)

    # Multi-label processing: Convert label list to a binary vector
    label_list = sample["label"]
    labels.append(label_list)

return np.array(inputs), labels
```

## 5.3 数据规范

```
# Load data
X, y = load_data('../data.json')
print(f"Initial X shape: {X.shape}")
ALL_LABELS = []
for label in y:
    if label not in ALL_LABELS:
        ALL_LABELS.append(label)
print(f"ALL LABELS: {ALL_LABELS}")
# N categories
N = len(ALL_LABELS)
print(f"N: {N}")
KEY_POINTS = 33
FEATURES = 4
# Reshape data to fit Conv1D input: (samples, steps, features)
```

```
X = X.reshape((X.shape[0], KEY_POINTS, 4))  # 33 keypoints with 4
    features (x, y, z, visibility)
# Checking the shape of the reshaped data
print(X.shape)
```

### 5.4 OneHotEncoding 编码

```
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.utils import to_categorical
# Initialize a LabelEncoder to convert strings to integers
label_encoder = LabelEncoder()
# Fit and transform the labels to integers
y_int = label_encoder.fit_transform(y)
# Now apply to_categorical for one-hot encoding
y_onehot = to_categorical(y_int, num_classes=len(label_encoder.classes_)
    )
print(f"One-hot encoded labels shape: {y_onehot.shape}")
print(y_onehot)
```

### 5.5 模型定义

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv1D, MaxPooling1D, Flatten, Dense
    , Dropout
# Define the model
model = Sequential()
# Add Conv1D layer
model.add(Conv1D(filters=64, kernel_size=3, activation='relu',
    input_shape=(KEY_POINTS, FEATURES)))
# Add MaxPooling1D layer
model.add(MaxPooling1D(pool_size=2))
# Add another Conv1D layer
model.add(Conv1D(filters=128, kernel_size=3, activation='relu'))
# Add another MaxPooling1D layer
model.add(MaxPooling1D(pool_size=2))
# Flatten the output from Conv1D layers
model.add(Flatten())
# Add Dense layer with dropout for regularization
model.add(Dense(128, activation='relu'))
```

```
model.add(Dropout(0.5))
# Output layer with softmax activation (for classification)
model.add(Dense(N, activation='softmax'))  # N is the number of classes
```

### 5.6 模型编译

```
# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics
   =['accuracy'])
# Summary of the model
model.summary()
```

### 5.7 训练模型

```
# Train the Model
history = model.fit(X, y_onehot, epochs=10, batch_size=32,
   validation_split=0.2)
# Evaluate the Model
loss, accuracy = model.evaluate(X, y_onehot)
print(f'Accuracy: {accuracy * 100:.2f}%')
```

## 6 预测图片

### 6.1 提取单张图片的关键点

```
import cv2
from mediapipe_impl.pose_estimation import PoseEstimationModule as pm
detector = pm.PoseDetector()
def extract_keypoints(image_path):
    cap = cv2.VideoCapture(image_path)
    success, img = cap.read()
    img = detector.find_pose(img=img)
    lm_list = detector.find_position(img, draw=False)
    return lm_list
```

### 6.2 预测图片

```
def predict_image(image_path):
    data = []
```

```
    keypoints = extract_keypoints(image_path)
    for keypoint in keypoints:
        data.extend([keypoint["x"], keypoint["y"], keypoint["z"],
           keypoint["visibility"]])
    X = np.array(data).reshape((1, 33, 4))
    print(X.shape)
    # 模型预测
    predictions = model.predict(X)

    # 获取预测类别的索引
    predicted_class = np.argmax(predictions, axis=1)[0]

    print(f"Predicted Class: {predicted_class}")
    print(ALL_LABELS[predicted_class])


# 示例：预测一张新图像
image_path = '../datasets/img.jpg'
result = predict_image(image_path)
```