系列文章

OpenVINO™ 工具套件

OpenVINO™ 工具套件
头部姿态评估网络应用演示

摘要：点头、摇头、左右转动动作识别应用演示……

姿态评估技术与框架

姿态评估(Pose estimation)是计算机视觉的研究热点之一，姿态评估的算法应用主要可以分为两部完成，第一步是对象检测，对象区域的定位与查找，截取图像ROI，第二步是根据对象检测定位截取ROI区域作为输入，完成姿态评估与预测。当前主要用于姿态评估的深度学习网络框架有以下几种:

OpenPose框架

是一个开源的姿态评估算法框架，支持多任姿态评估，可以实现实时的人体对象检测、身体、头部、手部姿态评估与人脸关机点检测，支持2D与3D，简易的C++接口调用与自定义训练。

DeepCut框架

跟OpenPose类似的姿态评估框，支持多人检测与姿态评估，特别是在图像与视频中运动场景下，包括足球与篮球运动场景。

其它的类似框架还包括有AlphaPose、DeepPose、PoseNet等。姿态评估算法主要应用场景在移动机器人、虚拟现实、人体跌倒检测、危险动作识别、机器人动作训练、活体验证等。

OpenVINO2021.02版本中涉及到姿态评估方面的支持主要有下列几个预训练模型:

模型名称	模型说明
head-pose-estimation-adas-0001	头部姿态评估，简单的全卷积网络
human-pose-estimation-0001	基于OpenPose, MobileNetv1做特征提取，18对关键点
human-pose-estimation-0002	基于EfficientHRNet，17对关键点检测
human-pose-estimation-0003	基于EfficientHRNet，17对关键点检测
human-pose-estimation-0004	基于EfficientHRNet，17对关键点检测

表1

从human-pose-estimation-0002到human-pose-estimation-0004输入图像的分辨率增大，计算复杂度增加，检测精度提升。从human-pose-estimation-0001到human-pose-estimation-0004都支持多人的姿态评估。OpenVINO通过上述五个模型实现基本的人体姿态评估需求。这里我们以头部姿态评估为例来完成姿态评估模型的OpenVINO部署与代码演示。

头部姿态评估模型

OpenVINO支持的头部姿态评估模型head-pose-estimation-adas-0001的输入与输出格式分别如下:
输入格式为NxCxHxW=1x3x60x60，期望的彩色图像通道顺序BGR、大小为60x60
输出层名称与格式如下:
name: "angle_y_fc", shape: [1, 1] - Estimated
name: "angle_p_fc", shape: [1, 1] - Estimated pitch
name: "angle_r_fc", shape: [1, 1] - Estimated roll

在三个维度方向实现头部动作识别，它们分别是:
pitch是俯仰角，是“点头“
yaw是偏航角，是‘摇头’
roll是旋转角，是“翻滚
它们的角度范围分别为：YAW [-90,90]， PITCH [-70,70], ROLL [-70,70]
这三个专业词汇其实是来自无人机与航空领域，计算机视觉科学家一大爱好就是搞新词，就把它们借用到头部姿态评估中，它们的意思图示如下:

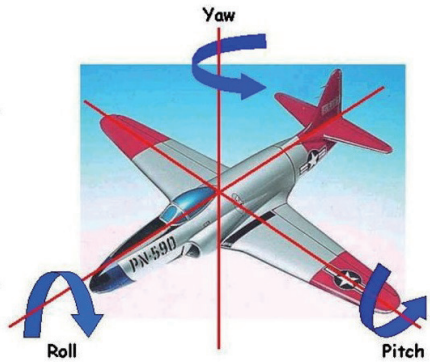

图1（来自网络）

对应到头部姿态评估中

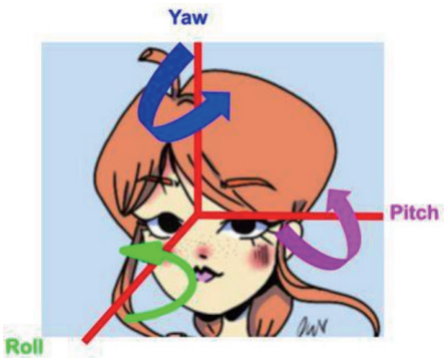

图2（来自网络）

该网络模型的格式通过Netron查看之后你会发现就是一些简单的Conv-ReLU-BN的叠加，最后是一个多任务的分支输出，模型结构跟2017年一篇论文里面的模型有点相似，我大胆猜测一波，OpenVINO头部姿态模型的开发者肯定是了解过此篇论文的，论文地址如下:
https://arxiv.org/pdf/1710.00925.pdf

论文中给出的模型结构如下:

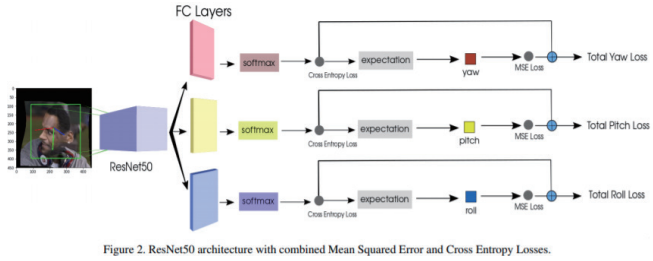

Figure 2. ResNet50 architecture with combined Mean Squared Error and Cross Entropy Losses.

图3(来自论文
Fine-Grained Head Pose Estimation Without Keypoints)

可以看到它的backbone部分是ResNet50，在OpenVINO中考虑到图像的输入分辨率与速度需求ResNet50的backbone被替换为一些简单的Conv-ReLU-BN叠加。图示如下:

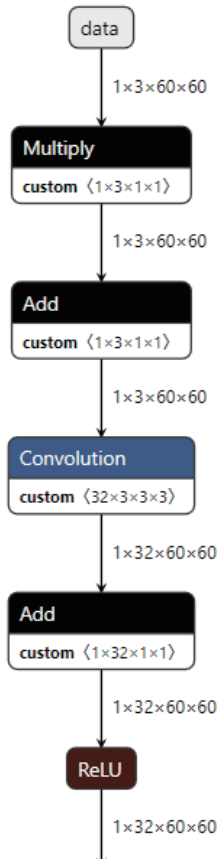

图4

代码实现

首先需要完成人脸检测，然后把对人脸部分ROI区域适当的向上增加大小之后完成截图，重新resize之后再作为输入头部姿态评估模型推理，得到三个角度，根据角度数值的范围获得最终的头部姿态评估结果。代码实现部分的人脸检测作为上一篇的对象检测部分关联知识，这里就不再展开详细说明。主要说明如何通过人脸检测结果截取ROI区域，调用头部姿态评估模型实现姿态评估、预测解析与结果显示，所以头部姿态评估的代码部分可以分为以下几步，它们分别是:

第一步:
加载头部姿态识别模型与设置输入输出格式

```
// load emotion model
InferenceEngine::CNNNetworkhead_pose_net = ie.ReadNetwork(head_model_xml, head_model_bin);
InferenceEngine::InputsDataMaphead_pose_inputs = head_pose_net.getInputsInfo();
InferenceEngine::OutputsDataMaphead_pose_outputs = head_pose_net.getOutputsInfo();
std::string hp_input_name = "";
for (auto item :head_pose_inputs) {
                hp_input_name = item.first;
                auto input_data = item.second;
                input_data->setPrecision(Precision::U8);
                input_data->setLayout(Layout::NCHW);
}

for (auto item :head_pose_outputs) {
                auto output_data = item.second;
                output_data->setPrecision(Precision::FP32);
}
```

第二步：获取可执行网络与推理请求

```
auto executable_hp_network = ie.LoadNetwork(head_pose_net, "CPU");
auto hp_request = executable_hp_network.CreateInferRequest();
```

第三步：截取输入、推理与解析输出

```
// check out of boundary
if (box.x< 0) {
                box.x = 0;
}
if (box.y< 0) {
                box.y = 0;
}
if ((box.width + box.x) >= curr_frame.cols) {
                box.width = curr_frame.cols - box.x;
}
if ((box.height+ box.y) >= curr_frame.rows) {
                box.height = curr_frame.rows - box.y;
}
fetch_head_pose(curr_frame, hp_request, box, hp_input_name);
```

其中fetch_head_pose函数方法的代码实现如下:

```
void fetch_head_pose(cv::Mat &image, InferenceEngine::InferRequest&request, cv::Rect&face_roi,
                std::string &e_input) {
                cv::Mat faceROI = image(face_roi);
                auto blob = request.GetBlob(e_input);
                matU8ToBlob<uchar>(faceROI, blob);

                request.Infer();

                // output prase
                auto output1 = request.GetBlob("angle_y_fc");
                auto output2 = request.GetBlob("angle_p_fc");
                auto output3 = request.GetBlob("angle_r_fc");
                const float* y_pred = static_cast<PrecisionTrait<Precision::FP32>::value_type*>(output1->buffer());
                const float* p_pred = static_cast<PrecisionTrait<Precision::FP32>::value_type*>(output2->buffer());
                const float* r_pred = static_cast<PrecisionTrait<Precision::FP32>::value_type*>(output3->buffer());
```

```
        std::string head_pose = "";
        if (p_pred[0] > 20 || p_pred[0] < -20) {
                head_pose += "pitch, ";
        }
        if (r_pred[0] > 20 || r_pred[0] < -20) {
                head_pose += "roll, ";
        }
        if (y_pred[0] > 20 || y_pred[0] < -20) {
                head_pose += "yaw, ";
        }
        putText(image, head_pose, face_roi.tl(),
cv::FONT_HERSHEY_SIMPLEX, 1.0, Scalar(255, 0, 255), 2,
8);
}
```

最终程序运行结果如下：

图5

如欲了解更多OpenVINO™开发资料，
请扫描下方二维码，我们会把最新资料及时推送给您。