

# 多通道 OpenVINO™ 工具套件

[入门指南](#)  
[操作步骤](#)  
[指南](#)  
[资源](#)  
[性能信息](#)  
[API 参考](#)

## 资源概述

### 示例

- [图像分类 C++ 示例异步](#)
- [图像分类 Python\\* 示例异步](#)
- [Hello 分类 C++ 示例](#)
- [图像分类 Python\\* 示例](#)
- [Hello Reshape SSD C++ 示例](#)
- [Hello NV12 输入分类 C++ 示例](#)
- [Hello 查询设备 C++ 示例](#)
- [Hello 查询设备 Python\\* 示例](#)
- [对象检测 C++ 示例 SSD](#)
- [对象检测 Python\\* 示例 SSD](#)
- [自动语音识别 C++ 示例](#)
- [神经风格迁移 C++ 示例](#)
- [神经风格迁移 Python\\* 示例](#)
- [性能指标评测 C++ 应用](#)
- [性能指标评测 Python\\* 应用](#)

### 演示

- [十字路口摄像头演示](#)
- [交互式面部检测 C++ 演示](#)
- [交互式人脸识别 Python\\* 演示](#)
- [TensorFlow\\* 对象检测 Mask R-CNN 分割演示](#)
- [多通道演示](#)
  - [多渠道面部检测 C++ 演示](#)
  - [多通道人体姿态估计 C++ 演示](#)
  - [多通道对象检测 YOLO\\* V3 C++ 演示](#)
- [使用 Faster R-CNN 的对象检测演示](#)
- [面向 CenterNet Python\\* 的对象检测演示](#)
- [对象检测 SSD C++ 演示，异步 API 性能展示](#)
- [对象检测 SSD Python\\* 演示，异步 API 性能展示](#)
- [安全障碍摄像头演示](#)

- 图像分割 C++ 演示
- 单个人体姿态估计 Python\* 演示
- 图像分割 Python\* 演示
- 智能课堂演示
- 文本检测演示
- 文本辨认 Python\* 演示
- 视线预估演示
- 人体姿态估计演示
- 3D 人体姿态估计 Python\* 演示
- 行人追踪器演示
- 超高分辨率演示
- 对象检测 YOLO\* V3 C++ 演示, 异步 API 性能展示
- 对象检测 YOLO\* V3 Python\* 演示, 异步 API 性能展示
- 动作识别演示
- 实例分割演示
- 3D 分割演示
- 图像检索 Python\* 演示
- 多通道多人追踪 Python\* 演示
- 语音库与语音识别演示
  - 语音库
  - 离线语音识别演示
  - 现场语音识别演示
  - Kaldi\* 统计语言模型转换工具

## 工具

- 精度检查器工具
  - 精度检查器示例
  - 配置 Caffe\* 启动程序
  - 配置 OpenVINO 启动程序
  - 配置 OpenCV\* 启动程序
  - 配置 MxNet\* 启动程序
  - 配置 TensorFlow\* 启动程序
  - 配置 TensorFlow\* Lite 启动程序
  - 配置 ONNX\* Runtime 启动程序
  - 配置 \*PyTorch 启动程序
  - 适配器
  - 注释转换器
  - 预处理程序
  - 后处理程序
  - 指标
  - 面向精度检查器的定制评估器
  - 阅读器
  - Caffe\* 安装技巧

训练后优化

量化

[DefaultQuantization](#) 算法

[AccuracyAwareQuantization](#) 算法

[使用训练后优化工具套件 API](#)

[定义配置文件](#)

[配置文件示例](#)

模型下载器

[交叉检查工具](#)

[编译工具](#)

预训练模型 ( 开放式 Model Zoo )

对象检测模型

[face-detection-adas-0001](#)

[face-detection-adas-binary-0001](#)

[face-detection-retail-0004](#)

[face-detection-retail-0005](#)

[person-detection-retail-0002](#)

[person-detection-retail-0013](#)

[person-detection-action-recognition-0005](#)

[person-detection-action-recognition-0006](#)

[person-detection-action-recognition-teacher-0002](#)

[person-detection-raisinghand-recognition-0001](#)

[person-detection-asl-0001](#)

[pedestrian-detection-adas-0002](#)

[pedestrian-detection-adas-binary-0001](#)

[pedestrian-and-vehicle-detector-adas-0001](#)

[vehicle-detection-adas-0002](#)

[vehicle-detection-adas-binary-0001](#)

[person-vehicle-bike-detection-crossroad-0078](#)

[person-vehicle-bike-detection-crossroad-1016](#)

[product-detection-0001](#)

[vehicle-license-plate-detection-barrier-0106](#)

对象识别模型

[age-gender-recognition-retail-0013](#)

[head-pose-estimation-adas-0001](#)

[asl-recognition-0003](#)

[license-plate-recognition-barrier-0001](#)

[vehicle-attributes-recognition-barrier-0039](#)

[emotions-recognition-retail-0003](#)

[landmarks-regression-retail-0009](#)

facial-landmarks-35-adas-0002  
person-attributes-recognition-crossroad-0230  
gaze-estimation-adas-0002

#### 重识别模型

person-reidentification-retail-0031  
person-reidentification-retail-0103  
person-reidentification-retail-0107  
person-reidentification-retail-0200  
face-reidentification-retail-0095

#### 语义分割模型

road-segmentation-adas-0001  
semantic-segmentation-adas-0001

#### 实例分割模型

instance-segmentation-security-0050  
instance-segmentation-security-0083  
instance-segmentation-security-0010

#### 人体姿态估计模型

human-pose-estimation-0001

#### 图像处理

single-image-super-resolution-1032  
single-image-super-resolution-1033  
text-image-super-resolution-0001

#### 文本检测

text-detection-0003  
text-detection-0004

#### 文本识别

text-recognition-0012  
handwritten-score-recognition-0003

#### 文本辨认

text-spotting-0001-detector  
text-spotting-0001-recognizer-decoder  
text-spotting-0001-recognizer-encoder

#### 动作识别模型

driver-action-recognition-adas-0002-encoder  
driver-action-recognition-adas-0002-decoder  
action-recognition-0001-encoder  
action-recognition-0001-decoder

#### 图像检索

image-retrieval-0001

#### 压缩模型

resnet18-xnor-binary-onnx-0001  
resnet50-binary-0001

## 动作识别 Python\* 演示

### 本文档

[操作步骤](#)

[运行](#)

[演示输出](#)

[另请参阅](#)

这是一款面向动作识别算法的演示应用，可对输入视频中执行的动作进行分类。该产品提供了以下预训练模型：

- `driver-action-recognition-adas-0002-encoder + driver-action-recognition-adas-0002-decoder`，这些模型用于驾驶员监控场景。它们可识别诸如安全驾驶、打电话等动作
- `action-recognition-0001-encoder + action-recognition-0001-decoder`，它们是面向 Kinetics-400 数据集的通用动作识别（400 个动作）模型。

有关预训练模型的更多信息，请参阅[模型文档](#)。

### 操作步骤

演示管道包括多个帧，即数据、编码器、解码器和渲染。每个步骤通过创建一个来源于 `PipelineStep` 基类的类，从而实现 `PipelineStep` 接口。请参阅 `steps.py` 了解实现详情。

- `DataStep` 从输入视频中读取帧。
- `EncoderStep` 预处理帧，并将其提供给编码器模型，以生成帧嵌入。
- `DecoderStep` 将 `EncoderStep` 生成的嵌入内容反馈给解码器模型，并生成预测。
- `RenderStep` 渲染预测结果。

管道步骤包含在 `AsyncPipeline` 中。使用 `parallel=True` 选项将步骤添加至管道后，便可以在单个线程上运行每个步骤。如果两个后续步骤出现在不同的线程上，它们将通过消息队列（例如，提供步骤结果或中止信号传输）进行通信。

为了确保最高的性能，推理引擎被包装在 `AsyncWrapper` 中，`AsyncWrapper` 通过以周期性顺序（异步启动对每个新输入的推理，返回运行时间最长的推理请求结

果) 调度推理请求, 使用推理引擎异步 API。您可以在 `action_recognition.py` 中更改 `num_requests` 的值, 以便为您的推理加速器找出并行运行推理请求的最佳数量 ( 计算棒和 GPU 可出色运行更多的推理请求 )。

**NOTE:** 默认情况下, Open Model Zoo 演示希望输入采用 BGR 通道顺序。如果经过训练的模型采用 RGB 顺序, 您需要手动重新排列演示应用中的默认通道顺序, 或者使用模型优化器工具 ( 指定 `--reverse_input_channels` 参数 ) 重新转换模型。有关参数的更多信息, 请参阅“何时反转输入通道顺序”一节, [该节使用通用转换参数转换模型](#)。

## 运行

使用 `-h` 选项运行应用可生成下列使用信息:

```
1 usage: action_recognition.py [-h] -m_en M_ENCODER -m_de M_DECODER -i INPUT
2           [-l CPU_EXTENSION] [-d DEVICE] [--fps FPS]
3           [-lb LABELS]
4
5 Options:
6 -h, --help            Show this help message and exit.
7 -m_en M_ENCODER, --m_encoder M_ENCODER
8           Required.Path to encoder model
9 -m_de M_DECODER, --m_decoder M_DECODER
10          Required.Path to decoder model
11 -i INPUT, --input INPUT
12          Required.Id of the video capturing device to open (to
13          open default camera just pass 0), path to a video or a
14          .txt file with a list of ids or video files (one
15          object per line)
16 -l CPU_EXTENSION, --cpu_extension CPU_EXTENSION
17          Optional.For CPU custom layers, if any.Absolute path
18          to a shared library with the kernels implementation.
19 -d DEVICE, --device DEVICE
20          Optional.Specify a target device to infer on.CPU,
21          GPU, FPGA, HDDL or MYRIAD is acceptable.The demo will
22          look for a suitable plugin for the device specified.
23          Default value is CPU
24 --fps FPS            Optional.FPS for renderer
25 -lb LABELS, --labels LABELS
26          Optional.Path to file with label names
27 --no_show            Optional.Don't show output
```

使用空选项列表运行应用程序将生成上述使用信息和错误消息。

如欲运行演示, 您可以使用公共或预训练模型。如欲下载预训练模型, 请使用 OpenVINO [模型下载器](#) 或访问 <https://download.01.org/opencv/>。

注：使用经过训练的模型运行演示之前，请确保已使用[模型优化器工具](#)将模型转换为推理引擎格式 (\*.xml + \*.bin)。

例如，若要运行车内驾驶员监控场景的演示，请提供编码器和解码器的路径、输入视频和包含标签名称的文件：

```
1 python3 action_recognition.py -m_en models/driver_action_recognition_tsd_0002_encoder.xml \  
2 -m_de models/driver_action_recognition_tsd_0002_decoder.xml \  
3 -i <path_to_video>/inputVideo.mp4 \  
4 -lb driver_actions.txt
```

## 演示输出

该应用使用 OpenCV 显示实时结果和当前的推理性能 (FPS)。

## 另请参阅

- [使用 Open Model Zoo 演示](#)
- [模型优化器](#)
- [模型下载器](#)

有关编译器优化的更多完整信息，请参阅我们的[优化声明](#)

## 支持

[英特尔® OpenVINO™ 工具套件分发版的英特尔® 开发人员专区论坛](#)

## Cookie

[英特尔 Cookie 和类似技术声明](#)