

华为 HiLens

开发指南

文档版本 01
发布日期 2019-12-05



版权所有 © 华为云计算技术有限公司 2024。保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

商标声明



HUAWEI和其他华为商标均为华为技术有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为云计算技术有限公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为云计算技术有限公司对本文档内容不做任何明示或暗示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

华为云计算技术有限公司

地址：贵州省贵安新区黔中大道交兴功路华为云数据中心 邮编：550029

网址：<https://www.huaweicloud.com/>

目录

1 使用前必读	1
2 HiLens Framework_Python 简介	2
3 初始化	4
3.1 初始化 HiLens Framework	4
3.2 释放 HiLens Framework 资源	6
4 视频输入模块	7
4.1 视频采集器	7
4.2 读取摄像头视频帧	8
4.3 获取视频的宽度	9
4.4 获取视频的高度	9
4.5 示例-输入	9
5 音频输入模块	11
5.1 音频采集器	11
5.2 读取音频数据	12
5.3 音频输入模块示例-输入	13
6 预处理模块	14
6.1 构造图像预处理器	14
6.2 改变图片尺寸	14
6.3 裁剪图片	15
6.4 转换图片颜色格式	15
6.5 示例-预处理	16
7 模型管理模块	17
7.1 模型加密（可选）	17
7.2 创建模型实例	17
7.3 模型推理	18
7.4 示例-模型管理	19
8 输出模块	21
8.1 构造一个用于输出的显示器	21
8.2 输出一帧图片	22
8.3 上传文件	22

8.4 上传缓冲区数据.....	23
8.5 发送消息.....	23
8.6 播放音频文件.....	24
8.7 示例-输出.....	25
9 EIServices 模块.....	26
9.1 模块简介.....	26
9.2 通用接口.....	26
9.3 示例-EIServices 模块.....	27
10 资源管理模块.....	29
10.1 获取模型路径.....	29
10.2 获得技能工作区目录.....	29
10.3 获得技能配置.....	29
10.4 从 OBS 下载文件.....	30
10.5 计算文件的 md5 值.....	30
10.6 示例-资源管理.....	30
11 难例上传模块.....	32
11.1 难例上传介绍及说明.....	32
11.2 初始化难例上传模块.....	33
11.3 检测算法中的难例图片判断.....	33
11.4 分类算法中的难例图片判断.....	34
11.5 难例图片上传.....	34
11.6 获取难例配置.....	35
11.7 更新难例配置.....	36
11.8 示例-难例上传.....	36
12 日志模块.....	38
12.1 设置打印日志的级别.....	38
12.2 打印 Trace 级别的日志.....	38
12.3 打印 Debug 级别的日志.....	39
12.4 打印 Info 级别的日志.....	39
12.5 打印 Warning 级别的日志.....	40
12.6 打印 Error 级别的日志.....	40
12.7 打印 Fatal 级别的日志.....	40
12.8 示例-日志.....	41
13 错误码.....	42

1 使用前必读

在华为HiLens管理控制台上[开发技能](#)时需要在线编辑或上传逻辑代码，而在逻辑代码中需要用到HiLens Framework，本文档针对开发者在开发可运行在HiLens Kit设备的技能的时候，介绍如何在逻辑代码中使用HiLens Framework API，您可以根据[表1-1](#)查找您需要的内容。

表 1-1 文档导读

章节	说明
HiLens Framework_Python 简介	快速了解HiLens Framework接口组成以及接口列表。
初始化 视频输入模块 预处理模块 模型管理模块 输出模块 EIServices模块 资源管理模块 日志模块	HiLens Framework封装的类和函数详细说明。

2 HiLens Framework_Python 简介

HiLens Framework通过封装底层接口、实现常用的管理功能，让开发者可以在华为 HiLens管理控制台上方便地开发技能，培育AI生态。

HiLens Framework的分层结构如图2-1所示，HiLens Framework封装了底层的多媒体处理库（摄像头/麦克风驱动模块Media_mini），以及D芯片相关的图像处理库（DVPP）和模型管理库（ModelManager），另外开发者也可以使用熟悉的视觉处理库OpenCV。在此之上，HiLens Framework提供了以下6个模块供开发者使用，方便开发诸如人形检测、疲劳驾驶检测等技能，模块说明如表2-1所示。用户可以开发Python版本的技能并在HiLens Kit上运行。

图 2-1 HiLens Framework 框架

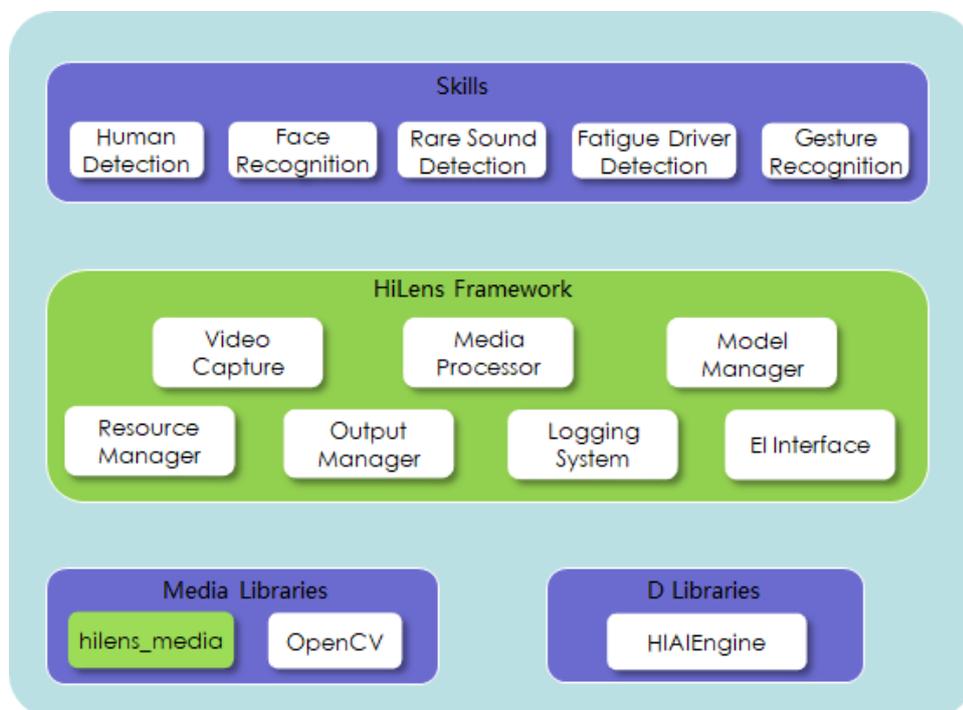


表 2-1 模块说明

序号	模块	功能
1	Input Manager	输入模块：负责视频、音频等输入数据的接入管理。
2	Media Processor	预处理模块：负责视频、音频等媒体数据的处理。
3	Model Manager	模型管理模块：负责模型的初始化与推断任务。
4	Output Manager	输出模块：负责流、文件、消息通知等输出任务的管理。
5	Resource Manager	资源管理模块：负责文件、图片、模型等资源的路径管理。
6	Logging System	日志模块：负责日志系统管理。

3 初始化

3.1 初始化 HiLens Framework

该接口用于初始化HiLens Framework。在调用HiLens Framework的其他接口之前，需要先做全局初始化。

- **接口调用**
hilens.init(verify)
- **参数说明**

表 3-1 参数说明

参数名	是否必选	参数类型	描述
verify	是	字符串	长度0到128的字符串。 应与华为HiLens管理控制台上 新建技能 时，所填写的“基本信息”中的“检验值”一致。如果不一致，HiLens Framework会强制技能停止。

图 3-1 新建技能

基本信息

技能模板 使用空模板 选择已有模板

* 技能名称

* 技能版本

* 适用芯片

* 校验值

* 应用场景

技能图标 Linux Android iOS LiteOS Windows

摘要

1/512

● **返回值**

0为HiLens Framework初始化成功，其他为失败。

该方法还可以用来验证技能是否损坏或被篡改。如果想要使用该功能，参数verify应该是开发者编写的一个函数的返回值，该返回值是实时计算要验证的文件的hash值。开发者在完成技能的开发后，用同样的hash方法计算出hash值，填入控制台**新建技能**的校验值。如下所示：

```
#!/usr/bin/python3.7

import hilens
def verify():
    # 开发者需要实现一个方法，来验证程序身份(以防被损坏、篡改)
    # 例如可以计算技能包中重要文件的Hash值，verify应当返回一个字符串(1~128字节)。
    # 在HiLens平台，技能开发中填入此Hash值。调用init方法后，技能会自动将此Hash值发送到平台上进行比对，并验证技能的使用许可。

    # 调试期间，开发者可以考虑使用一个固定的字符串来进行校验，以便于修改代码。
    # 因为Python脚本源码下发到设备上较容易篡改，对于商用技能，建议开发者使用C++进行开发。
    # 注意：正式发布的技能不应使用硬编码的字符串来校验！
    return "hello"

def main():
    # 初始化HiLens
    rc = hilens.init(verify())
    # 如果在技能开发调试阶段不想用此功能，那么直接填写一个静态字符串即可
    # 如: hilens.init("hello")
    if rc != 0:
        hilens.error("Failed to initialize HiLens")
        return

    # 业务代码
    pass

    # 完成后，清理资源
    hilens.terminate()

if __name__ == '__main__':
    main()
```

3.2 释放 HiLens Framework 资源

该接口用于终止HiLens Framework。在程序结束时，调用此接口用以释放相关资源。

- **接口调用**
hilens.terminate()
- **返回值**
0为资源释放成功，其他为失败。

4 视频输入模块

4.1 视频采集器

该接口用于构造一个视频采集器，用以打开HiLens Kit自带的摄像头、构造一个IPC摄像头视频采集器（目前支持RTSP协议的IPC）或构造一个UVC摄像头（符合USB视频类(USB Video Class)规范的摄像头设备）视频采集器。目前HiLens Kit有两个USB接口，但只能接入一个USB摄像头。

升级到1.0.7及以后的固件版本，本接口支持读取本地MP4文件，并支持设置通过IPC摄像头或者本地MP4文件读取到的视频帧图片宽度、高度。

- **接口调用**
hilens.VideoCapture(camera)
1.0.7及以后固件版本：
hilens.VideoCapture(camera, width, height)
- **参数说明**

表 4-1 参数说明

参数名	是否必选	参数类型	描述
camera	否	<ul style="list-style-type: none"> 字符串 整型0 	<ul style="list-style-type: none"> 如果不输入参数，则构造一个视频采集器以打开HiLens Kit自带的摄像头，一台设备只有一个技能可以使用自带摄像头，不然会资源抢占导致错误。 如果输入设备配置中的摄像头名称，则构造一个IPC摄像头视频采集器。此时优先输入设备配置中的摄像头名称，也可以直接传入形如rtsp://xxx的取流地址。摄像头名称可登录华为HiLens控制台，在“设备管理>设备列表>摄像头配置>摄像头管理”查看。 如果输入整型0，则构造一个UVC摄像头视频采集器（需插入UVC摄像头）。 如果输入本地MP4视频文件路径，则构造一个MP4视频采集器。
width	否，需要和height同时使用	整型	设置读取到的视频帧图片宽度（要求为16的倍数，推荐为32的倍数，且最小为128），仅支持IPC摄像头和MP4视频文件进行设置。如果不输入参数，默认按视频帧原始宽高。
height	否，需要和width同时使用	整型	设置读取到的视频帧图片高度（要求为2的倍数，且最小为128），仅支持IPC摄像头和MP4视频文件进行设置。如果不输入参数，默认按视频帧原始宽高。

- **返回值**
 - 自带摄像头的视频采集器。
 - IPC视频采集器。
 - UVC摄像头视频采集器。
 - MP4视频采集器。
 - 如果创建失败则抛出一个CreateError。开发者可以在查看技能日志输出。

4.2 读取摄像头视频帧

该接口用于读取一帧视频。注意IPC摄像头和MP4视频返回的是YUV_NV21颜色排布的数据，而UVC类型的摄像头返回的是BGR颜色排布的数据。

- **接口调用**
hilens.VideoCapture.read()
- **返回值**
一帧视频数据。参数类型为numpy数组（dtype为uint8），兼容cv2。

4.3 获取视频的宽度

该接口用于获取视频的宽度。

- **接口调用**
hilens.VideoCapture.width
- **返回值**
视频宽度。

4.4 获取视频的高度

该接口用于获取视频的高度。

- **接口调用**
hilens.VideoCapture.height
- **返回值**
视频高度。

4.5 示例-输入

输入模块示例如下所示：

📖 说明

- 在调用视频采集器接口时，此处示例是以固件版本大于等于1.0.7时调用视频采集器接口为例，调用接口支持设置视频大小，即支持调用接口**hilens.VideoCapture(camera, width, height)**时设置视频大小参数“width”和“height”。
- 在HiLens Studio或者固件版本小于1.0.7时，不支持设置视频大小。

```
#!/usr/bin/python3.7

import hilens
import numpy as np

def run():
    # 构造摄像头
    cap0 = hilens.VideoCapture() # 自带摄像头
    cap1 = hilens.VideoCapture("IPC1") # 摄像头配置中name为"IPC1"的IPC。摄像头配置可登录华为HiLens
    # 控制台，在“技能开发>技能管理>新建技能”中的“运行时配置”添加
    cap2 = hilens.VideoCapture("rtsp://192.168.1.1/video") # 地址为rtsp://192.168.1.1/video的RTSP视频流
    cap3 = hilens.VideoCapture("/tmp/test.mp4",1920,1080) # 读取HiLens Kits上/tmp目录下名为test.mp4的视
    # 频文件,并调整视频帧宽高为1920, 1080 (需要固件版本大于等于1.0.7)
    cap4 = hilens.VideoCapture(0) # 目前只支持单路uvc摄像头, 编号为0

    # 获取视频尺寸
    w = cap0.width
    h = cap0.height
    hilens.info("width: %d, height: %d" % (w, h))

    # 读取视频数据
    frame0 = cap0.read()
    # 其他处理
    pass

if __name__ == '__main__':
    hilens.init("hello")
```

```
run()  
hilens.terminate()
```

5 音频输入模块

5.1 音频采集器

该接口用于构造一个音频采集器，从本地麦克风获取音频或者本地音频文件获取音频数据。

- **接口调用**
1.0.8及以上固件版本
hilens.AudioCapture(file_path)
1.1.2及以上固件版本
hilens.AudioCapture(sample_rate, bit_width, nSamples, sound_mode)
- **参数说明**

表 5-1 参数说明

参数名	是否必选	参数类型	描述
file_path	否	字符串	音频文件路径，从该文件获取音频数据，构造一个音频文件数据的采集器。

参数名	是否必选	参数类型	描述
sample_rate	否	整型	采样率，本地麦克风录音参数。默认值为“AUDIO_SAMPLE_RATE_44100”，可取值： AUDIO_SAMPLE_RATE_8000 AUDIO_SAMPLE_RATE_12000 AUDIO_SAMPLE_RATE_11025 AUDIO_SAMPLE_RATE_16000 AUDIO_SAMPLE_RATE_22050 AUDIO_SAMPLE_RATE_24000 AUDIO_SAMPLE_RATE_32000 AUDIO_SAMPLE_RATE_44100 AUDIO_SAMPLE_RATE_48000 AUDIO_SAMPLE_RATE_64000 AUDIO_SAMPLE_RATE_96000
bit_width	否	整型	位宽，本地麦克风录音参数。默认值为“AUDIO_BIT_WIDTH_16”。
nSamples	否	整型	每帧音频采样点数，本地麦克风录音参数。默认值为1024，取值范围[80, 2048]。
sound_mode	否	整型	声道模式，本地麦克风录音参数。默认值为“AUDIO_SOUND_MODE_MONO”，可取值： AUDIO_SOUND_MODE_MONO AUDIO_SOUND_MODE_STEREO

📖 说明

- 本地麦克风只有一个，不支持多个进程设置不同的录音参数，先设置的生效。
- 麦克风录音接口和[播放音频文件](#)的接口不能同时使用。
- **返回值**
 - 音频数据采集器。
 - 如果创建失败则抛出一个CreateError。开发者可以在查看技能日志输出。

5.2 读取音频数据

该接口用于读取n帧音频数据。仅支持1.0.8及以上固件版本。

- **接口调用**
hilens.AudioCapture.read(nFrames)
- **参数说明**

表 5-2 参数说明

参数名	是否必选	参数类型	描述
nFrames	否	整型	要读取的帧数量，默认值为1。支持最多一次读取 512帧。

- **返回值**
n帧音频数据。参数类型为numpy数组（dtype为int16）。
如果读取失败抛出一个RunTimeError。

5.3 音频输入模块示例-输入

音频输入模块示例如下所示：

```
#!/usr/bin/python3.7

import hilens
import wave

def run():
    # 构造本地音频文件采集器并将解码后数据保存到wav文件
    cap = hilens.AudioCapture("\tmp\test.aac")
    # 构造本地麦克风采集器
    cap2 = hilens.AudioCapture(sample_rate=hilens.AUDIO_SAMPLE_RATE_16000,
bit_width=hilens.AUDIO_BIT_WIDTH_16, nSamples=1000,
sound_mode=hilens.AUDIO_SOUND_MODE_MONO)
    wav = wave.open("test.wav", "wb")
    wav.setnchannels(2) # 设置通道数为2
    wav.setsampwidth(2) # 设置采样率为16Bit
    wav.setframerate(44100) #设置采样率
    for i in range(100): # 读取500帧数据并写到文件（约12S）
        data = cap.read(5)
        wav.writeframes(data.tobytes())
    wav.close() # 当前目录生成test.wav音频文件，可用常见播放器打开

if __name__ == '__main__':
    hilens.init("hello")
    run()
    hilens.terminate()
```

6 预处理模块

6.1 构造图像预处理器

该接口用于构造一个预处理器，用于进行Resize/Crop操作（3559硬件加速）。

- **接口调用**
hilens.Preprocessor()
- **返回值**
返回预处理器实例。
如果失败则抛出一个CreateError。开发者可以在查看技能日志输出。

6.2 改变图片尺寸

该接口用于改变一张图片的尺寸。

- **接口调用**
hilens.Preprocessor.resize(src, w, h, t)
- **参数说明**

表 6-1 参数说明

参数名	是否必选	参数类型	描述
src	是	<class 'numpy.ndarray'>对象	源图，必须为NV21的格式。宽度范围[64, 1920], 2的倍数；高度范围[64, 1080], 2的倍数。
w	是	正整型	缩放后的图片宽度，范围[64, 1920], 2的倍数。
h	是	正整型	缩放后的图片高度，范围[64, 1080], 2的倍数。

参数名	是否必选	参数类型	描述
t	是	整型0或1	目的图片的格式，0为NV21,1为NV12。

- **返回值**
如果成功则返回resize后的图片，<class 'numpy.ndarray'>对象。
失败则抛出一个ValueError。

6.3 裁剪图片

该接口用于裁剪一张图片。

- **接口调用**
hilens.Preprocessor.crop(src, x, y, w, h, t)
- **参数说明**

表 6-2 参数说明

参数名	是否必选	参数类型	描述
src	是	<class 'numpy.ndarray'>对象	源图，必须为NV21的格式。宽度范围[64, 1920], 2的倍数；高度范围[64, 1080], 2的倍数。
x	是	正整型	裁剪区域左上角x坐标，范围[0, 1920], 2的倍数。
y	是	正整型	裁剪区域左上角y坐标，范围[0, 1080], 2的倍数。
w	是	正整型	裁剪宽度，范围[64, 1920], 2的倍数。
h	是	正整型	裁剪高度，范围[64, 1080], 2的倍数。
t	是	整型0或1	目的图片的格式，0为NV21,1为NV12。

- **返回值**
如果成功则返回crop后的图片，<class 'numpy.ndarray'>对象。
失败则抛出一个ValueError。

6.4 转换图片颜色格式

该接口用于转换图片颜色格式。opencv原生未提供RGB/BGR到NV12/NV21的转换选项，故在这里做补充。

- 接口调用
hilens.cvt_color(src, code)
- 参数说明

表 6-3 参数说明

参数名称	是否必选	参数类型	参数描述
src	是	<class 'numpy.ndarray'>对象	源图(GBR888或RGB888)。
code	是	枚举类型, {RGB2YUV_NV12, RGB2YUV_NV21, BGR2YUV_NV12, BGR2YUV_NV21}	指定何种转换类型。

- 返回值
<class 'numpy.ndarray'>对象, 转换后的图片(NV12或NV21), 如果转换失败则返回一个空的numpy.ndarray对象。

6.5 示例-预处理

预处理模块示例如下所示:

```
import hilens
import cv2
import numpy as np

def run():
    # 构造摄像头
    cap = hilens.VideoCapture()
    # 获取一帧画面,自带摄像头获取图像为YUV格式
    # 自带摄像头默认分辨率为720p, 所以YUV图像的大小为(720*3/2,1280)
    frame = cap.read()

    # 转换图片的颜色格式, YUV转BGR需要通过opencv完成
    image_bgr = cv2.cvtColor(image_yuv, cv2.COLOR_YUV2BGR_NV21)

    # 转换图片的颜色格式, BGR/RGB转YUV可通过hilens.cvt_color接口
    image_yuv = hilens.cvt_color(image_bgr, hilens.BGR2YUV_NV21)

    # 构造预处理器,只支持YUV_NV21/NV12格式图片处理
    proc = hilens.Preprocessor()
    # 调整图片大小
    resized = proc.resize(image_yuv, 640, 480,0)
    # 裁剪图片
    cropped = proc.crop(image_yuv, 10, 20, 64, 64, 0)

    # 其他处理
    pass

if __name__ == '__main__':
    hilens.init("hello")
    run()
    hilens.terminate()
```

7 模型管理模块

7.1 模型加密（可选）

HiLens Kit支持模型加密，模型加密后，仅支持HiLens Framework接口调用。

模型加密操作

下载加密工具：[crypto_tool](#)，并拷贝到设备系统的“/tmp”目录下，赋予执行权限：

```
chmod +x crypto_tool
```

使用方法请参见工具的帮助信息：

```
./crypto_tool  
./crypto_tool encode --model_file plainModelfile --cipher_file cipherModelfile
```

其中“model_file”为待加密模型文件，“cipher_file”为加密之后的模型文件。

接口调用

和未加密模型一样调用接口，详情请见：

- [创建模型实例](#)
- [模型推理](#)
- [示例-模型管理](#)

7.2 创建模型实例

根据技能的模型，创建一个模型实例。HiLens Kit可以使用昇腾310芯片支持的模型来进行推理，使用此方法来构造一个后续用于推理的模型。

当返回的对象被析构时，对应的模型资源也被释放。

当前支持创建普通模型和加密模型

- **接口调用**
hilens.Model(filepath)

- 参数说明

表 7-1 参数说明

参数名称	是否必选	参数类型	参数描述
filepath	是	字符串	<p>模型文件的路径。</p> <p>可根据自己在新建技能>填写技能内容时上传模型的方式查看模型文件的路径。</p> <ul style="list-style-type: none"> • 方式一：在“模型”字段添加多个模型。此时获取模型路径请参见获取模型路径。 • 方式二：提前将多个模型和代码一起打包上传至OBS，“代码上传方式”选择“从OBS上传文件”。此时模型文件的路径为模型相对于代码所在文件位置的相对路径。

图 7-1 技能内容



- 返回值

<class 'hilens.Model'>模型对象。

模型构造失败则会抛出一个CreateError，并在日志上打印出错误码（例如0x1013011为模型路径错误）。

7.3 模型推理

模型初始化成功后，调用infer接口进行模型推理。灌入一组数据，并得到推理结果。输入数据的类型不是uint8或float32数组组成的list将会抛出一个ValueError。

- 接口调用

```
hilens.Model.infer(inputs)
```

- 参数说明

表 7-2 参数说明

参数名	是否必选	参数类型	描述
inputs	是	列表	推理输入，一组uint8或float32数组组成的list，支持多输入。

- 返回值

模型输出，一组float数组组成的list，支持多输出。

7.4 示例-模型管理

模型管理示例如下：

```
#!/usr/bin/python3.7

import hilens
import numpy as np
def run():

    # 构造摄像头
    cap = hilens.VideoCapture()
    # 获取一帧画面,自带摄像头获取图像为YUV_NV21格式，默认分辨率720p
    frame = cap.read()

    # 加载模型
    # filepath不能只是文件名，如果模型与程序在同一个目录，取相对路径则应当写作"./my_model.om"
    # 如果模型是在技能开发页面中附加进来的，则使用hilens.get_model_dir()可以得到模型所在目录，应当写为：
    # model = hilens.Model(hilens.get_model_dir() + "my_model.om")
    # 如果有多个模型，需要分别加载
    model1 = hilens.Model("./my_model1.om")
    model2 = hilens.Model("./my_model2.om")
    model3 = hilens.Model("./my_model3.om")

    # 假设模型1的输入是一张480*480的YUV_NV21图片，数据类型为uint8
    pro = hilens.Preprocessor()
    input1 = pro.resize(frame, 480, 480, 1)
    input1 = input1.flatten()
    # 进行推理
    output1 = model1.infer([input1])

    # 假设模型2的输入为模型1的输出（已经是list），数据类型为float32
    input2 = output1
    # 进行推理
    output2 = model2.infer(input2)

    # 假设模型3的输入是多输入，数据类型为float32
    ip_0 = (sample_data[0]).transpose(0, 3, 1, 2).astype(np.float32).flatten()
    ip_1 = (sample_data[1]).transpose(0, 3, 1, 2).astype(np.float32).flatten()
    ip_2 = (sample_data[2]).transpose(0, 3, 1, 2).astype(np.float32).flatten()
    ip_3 = (sample_data[3]).transpose(0, 3, 1, 2).astype(np.float32).flatten()
    ip_4 = (sample_data[4]).transpose(0, 3, 1, 2).astype(np.float32).flatten()
    # 进行推理
    output3 = model3.infer([ip_0, ip_1, ip_2, ip_3, ip_4])

    # 其他处理
    pass

if __name__ == '__main__':
    hilens.init("hello")
```

```
run()
hilens.terminate()
```

如果推理的实际输入与模型输入大小不一致，推理将会失败。此时infer的返回值将是一个int的错误码，日志会报出错误信息，开发者可以通过错误信息来定位错误。如下所示：

```
>>> input0 = np.zeros((480*480*3), dtype='uint8')
>>> outputs = model.infer([input0])
2019-09-30 18:44:24,075 [ERROR][SFW] Ascend 310: aiModelManager Process failed, please check your
input. Model info:
inputTensorVec[0]: name=data n=1 c=3 h=480 w=480 size=345600
outputTensorVec[0]: name=output_0_reg_reshape_1_0 n=1 c=6750 h=1 w=1 size=27000
your input size:0: 691200;
>>> outputs
17
>>> type(outputs)
<class 'int'>
```

8 输出模块

8.1 构造一个用于输出的显示器

显示器类，用来构造一个显示器，将视频（图片帧）输出到显示器类。

- **接口调用**
hilens.Display(type, path=None)
- **参数说明**

表 8-1 参数说明

参数名称	是否必选	参数类型	参数描述
type	是	枚举类型，可选 hilens.HDMI、 hilens.RTMP、 hilens.H264_FILE	<ul style="list-style-type: none"> • hilens.HDMI：直接通过设备的HDMI接口输出到显示屏，目前只支持一路数据显示到HDMI。 • hilens.RTMP：实时输出到RTMP服务器供用户查看。 • hilens.H264_FILE：输出到文件(h264编码的裸流)供用户查看。
path	否	字符串	如果类型为HDMI则忽略此参数，如果是RTMP则path为RTMP服务器的URL（rtmp://xxxx），为H264_FILE则path为输出文件的路径（如hilens.get_workspace_path()+” / out.h264”）。

如果是H264_FILE类型的，需要注意，生成的文件仅是h264编码的裸视频流，不含帧率等信息。而且HiLens Framework并未限制文件大小。所以此功能建议只作为调试使用，如果需要保存大文件，建议将文件位置设为/var/lib/docker目录。

- **返回值**
返回一个显示器实例。

如果创建失败则抛出一个CreateError。开发者可以在查看技能日志输出、或 cat /dev/logmpp来定位错误原因。

8.2 输出一帧图片

显示一张图片。在第一次调用该接口时，Display会根据输入的图片尺寸来设置视频尺寸，此后的调用中skill必须保证输入图片的尺寸与之前的一致，待显示的图片，必须为NV21格式，注意HDMI只支持一路输出且输出图像宽高需要大于等于128，否则会导致输出失败。

- **接口调用**
hilens.Display.show(frame)
- **参数说明**

表 8-2 参数说明

参数名	是否必选	参数类型	描述
frame	是	<class 'numpy.ndarray'>类型	待显示的图片，必须为NV21格式。

- **返回值**
成功则返回0，其他为失败。

8.3 上传文件

上传一个文件到OBS，此方法会阻塞线程，直至上传结束。上传的根目录（目标OBS桶的位置）是用户在华为HiLens控制台上针对每个设备配置的（参见[配置数据存储位置](#)），如果用户没有给设备配置这个信息，那么上传文件会失败。

- **接口调用**
hilens.upload_file(key, filepath, mode)
1.0.6固件版本之后请使用hilens.upload_file_to_obs(key, filepath, mode)
- **参数说明**

表 8-3 参数说明

参数名称	是否必选	参数类型	参数描述
key	是	字符串	上传到OBS中的具体路径，不需要网址信息，只需OBS中的文件路径即可，如 'test/output.jpg'。 注意key中，两个目录名不可为 "." 开头或结尾，亦不可出现连续的多个斜杠如"//"。
filepath	是	字符串	待上传文件的绝对路径。

参数名称	是否必选	参数类型	参数描述
mode	是	字符串	上传模式。两种可选：“write”-覆盖方式，“append”-追加方式。

- **返回值**
成功则返回0。否则为失败。

8.4 上传缓冲区数据

上传一个buffer到OBS，此方法会阻塞线程，直至上传结束。上传的根目录（目标OBS桶的位置）是用户在华为HiLens控制台上针对每个设备配置的（参见[配置数据存储位置](#)），如果用户没有给设备配置这个信息，那么上传文件会失败。

- **接口调用**
hilens.upload_bufer(key, buffer, mode)
1.0.6固件版本之后请使用upload_buffer_to_obs(key, buffer, mode)
- **参数说明**

表 8-4 参数说明

参数名称	是否必选	参数类型	参数描述
key	是	字符串	上传到OBS中的具体路径，不需要网址信息，只需OBS中的文件路径即可，如 'test/output.txt'。 注意key中，两个目录名不可为 "." 开头或结尾，亦不可出现连续的多个斜杠如"//"。
buffer	是	字符串或字节数组	需要上传的内容，类型为str或bytes。
mode	是	字符串	上传方式，两种可选：'write'-覆盖方式，'append'-追加方式。

- **返回值**
成功则返回0。否则为失败。

8.5 发送消息

部分场景的技能需要发送消息到用户的手机或邮箱，例如某一技能具备检测陌生人功能，在检测到陌生人后需要发送消息给用户。开发者可调用如下接口实现该功能。

- **接口调用**
hilens.send_msg(subject, message)

仅1.0.7-1.2.2版本的固件提供该接口。

- **参数说明**

表 8-5 参数说明

参数名称	是否必选	参数类型	参数描述
subject	是	字符串，长度不能为0	消息发送的主题名称。如果消息是发送到用户的邮箱，那么该字段是邮件的主题。如果消息是发送到用户的手机，则该字段没有意义。 消息发送方式是邮箱还是手机，用户可在使用技能时，在华为HiLens控制台 配置订阅消息 过程中设置。
message	是	字符串	要发送的消息内容。

- **返回值**

成功则返回0。否则为失败。

8.6 播放音频文件

播放本地AAC格式音频文件。在盒子音频输出口接上耳机或者音箱，调用该接口时可听到声音。

- **接口调用**

```
audio_out = hilens.AudioOutput()
audio_out .play_aac_file(file_path, vol)
```

- **参数说明**

表 8-6 参数说明

参数名	是否必选	参数类型	描述
file_path	是	字符串	本地音频文件绝对路径。
vol	是	整型	播放音频文件音量，取值范围[-121, 6]。

- **返回值**

成功则返回0，其他为失败。

8.7 示例-输出

本示例展示了多种输出端的接口调用，在使用前请确保各种输出端已连接并可用，若您的某种输出端条件不具备，请将示例代码当中相应的代码注释掉或者删除，再运行示例代码。输出模块示例如下所示：

```
#!/usr/bin/python3.7
import hilens
import cv2
import numpy as np
import wave

def run():
    # 显示到HDMI接口的显示器
    # 目前只支持一路数据显示到HDMI，多个技能同时显示到HDMI会报错
    disp0 = hilens.Display(hilens.HDMI)
    # 推流到地址为rtmp://192.168.1.1/stream的服务器
    disp1 = hilens.Display(hilens.RTMP, "rtmp://192.168.1.1/stream")
    # 把视频写到文件,hilens.H264_FILE生成的文件是只包括h264编码的裸视频流文件
    # 并且没有限制文件大小，建议仅作为调试使用
    disp2 = hilens.Display(hilens.H264_FILE, hilens.get_workspace_path() + "video.h264")
    # hilens.get_workspace_path()返回技能工作区目录，详细介绍请参考资源管理模块

    # 构造一个本地摄像头视频采集器
    cap = hilens.VideoCapture()

    # 显示画面到HDMI显示设备
    disp0.show(cap.read())

    # 上传视频video.h264到obs
    # 先生成h264格式的视频文件
    disp2.show(cap.read())
    # 上传到obs
    hilens.upload_file_to_obs("video", hilens.get_workspace_path() + "video.h264", "write")

    # 把1234追加到obs的test4文件
    hilens.upload_buffer_to_obs("test4", "1234", "append")

    # 通过缓存区数据上传图片到obs
    # 转换成BGR
    frame = cap.read()
    img_bgr = cv2.cvtColor(frame, cv2.COLOR_YUV2BGR_NV21)
    # 把当前图片按照jpg格式进行编码
    img_encode = cv2.imencode(".jpg", img_bgr)[1]
    # 通过upload_bufer上传缓存区图片，图片格式与编码格式保持一致
    hilens.upload_bufer("img.jpg", img_encode, "write")

    # 播放音频文件
    audio_out = hilens.AudioOutput()
    audio_out.play_aac_file("test.aac", 6)

if __name__ == '__main__':
    hilens.init("hello")
    run()
    hilens.terminate()
```

9 EIServices 模块

9.1 模块简介

EIServices模块提供便捷接口使得开发者可以快速调用华为云上的各种AI服务，相关的AI服务信息请参考服务的文档说明。

目前提供两种接口供开发者使用：通用接口、常用接口，仅适用于固件 1.0.7及之后版本调用华为云北京四区域接口。

9.2 通用接口

通用接口可以访问华为云上的各种AI服务，是否需要开通及调用具体参数请参考各服务说明。

- **接口调用**
hilens.EIServices.Request(method, host, uri, queryParams, payload, headers)
- **参数说明**

表 9-1 参数说明

参数名称	是否必选	参数类型	参数描述
method	是	枚举	请求方法，可选 hilens.GET,hilens.POST,hilens.PUT,hilens.DELETE。
host	是	字符串	请求域名。host+uri需要是完整的请求url。
uri	是	字符串	请求uri。host+uri需要是完整的请求url。
queryParams	是	字符串	查询字符串。
payload	是	字符串	请求消息体。

参数名称	是否必选	参数类型	参数描述
headers	是	字符串	请求消息头。提供hilens.EIHeaders()对象方便添加请求头，使用示例： headers = hilens.EIHeaders() headers.add("Content-Type: application/json")

- **返回值**

EIResponse结构体，包含requestState和responseBody两个成员，详情请参见[表 9-2](#)。

表 9-2 返回值说明

名称	参数类型	参数说明
EIResponse.requestState	布尔类型	表示请求状态。True表示成功，False表示失败。
EIResponse.responseBody	字符串	请求响应体。

9.3 示例-EIServices 模块

EIServices模块输出示例代码如下：

```
import hilens
import cv2
import numpy as np
import base64
import json

def run():
    # 使用图片作为输入
    f=open('/tmp/dengchao.jpg','rb')
    base_f=base64.b64encode(f.read())
    f_string=base_f.decode('utf-8')

    # 使用Mat格式或者直接从摄像头输入
    #img = cv2.imread("/tmp/dengchao.jpg")
    cap = hilens.VideoCapture()
    frame = cap.read()
    img = cv2.cvtColor(frame, cv2.COLOR_YUV2BGR_NV21)
    img_str = cv2.imencode('.jpg', img)[1].tostring() # 将图片编码成流数据，放到内存缓存中，然后转化成string格式
    b64_code = base64.b64encode(img_str) # 编码成base64
    f_string1=b64_code.decode('utf-8')
```

```

headers = hilens.EIHeaders()
body = {"image_base64": f_string}
json_str = json.dumps(body)
response5 = hilens.EIServices.Request(hilens.POST, "hilens-api.cn-north-4.myhuaweicloud.com", "/v1/
human-detect", "", json_str, headers)
print(response5.requestState)
print(response5.responseBody)
body1 = {"face_set_name": "ei_test"}
json_str1 = json.dumps(body1)
response6 = hilens.EIServices.Request(hilens.POST, "face.cn-north-4.myhuaweicloud.com", "/v1/
fc3bc995e9c441369d71159c67404e88/face-sets", "", json_str1, headers)
print(response6.requestState)
print(response6.responseBody)
response7 = hilens.EIServices.AddFace("ei_test", f_string, "")
print(response7.requestState)
print(response7.responseBody)
response8 = hilens.EIServices.SearchFace("ei_test", f_string, 1, 0.93, "")
print(response8.requestState)
print(response8.responseBody)
response9 = hilens.EIServices.Request(hilens.POST, "face.cn-north-4.myhuaweicloud.com", "/v1/
fc3bc995e9c441369d71159c67404e88/face-sets/ei_test/search", "", json_str, headers)
print(response9.requestState)
print(response9.responseBody)
response10 = hilens.EIServices.Request(hilens.DELETE, "face.cn-north-4.myhuaweicloud.com", "/v1/
fc3bc995e9c441369d71159c67404e88/face-sets/ei_test", "", "", headers)
print(response10.requestState)
print(response10.responseBody)
response11 = hilens.EIServices.Request(hilens.GET, "face.cn-north-4.myhuaweicloud.com", "/v1/
fc3bc995e9c441369d71159c67404e88/face-sets/ei_test", "", "", headers)
print(response11.requestState)
print(response11.responseBody)

if __name__ == '__main__':
    hilens.init("hello")
    run()
    hilens.terminate()

```

10 资源管理模块

10.1 获取模型路径

获得技能模型所在目录的路径（末尾带”/“）。适用于创建技能时从模型管理页面选择模型并下发的情况，模型文件会被下载至一个专门存储模型的位置，通过此函数来获取模型所在目录。如果HiLens Framework没有获取模型所在目录，则返回当前路径（即代码所在目录）。

- **接口调用**
hilens.get_model_dir()
- **返回值**
返回字符串，技能模型所在目录路径，失败返回空字符串。

10.2 获得技能工作区目录

获得技能工作区目录的路径（末尾带”/“）。设计上不推荐在技能安装目录下写操作，故需要指定各技能可写的工作区位置。如果HiLens Framework没有获取到工作区位置，则返回当前路径。

- **接口调用**
hilens.get_workspace_path()
- **返回值**
返回字符串，工作目录的绝对路径"/.../data/"，失败返回空字符串。

10.3 获得技能配置

获得技能配置。如果没有成功获取到则返回None。

- **接口调用**
hilens.get_skill_config()
- **返回值**
成功返回一个技能配置的dict，失败返回None。

10.4 从 OBS 下载文件

从OBS下载文件。

- **接口调用**
hilens.download_from_obs(url, download_to)
- **参数说明**

表 10-1 参数说明

参数名	是否必选	参数类型	描述
url	是	字符串	OBS资源的链接。资源链接获取详情请参见 OBS控制台指南>通过对象URL访问对象 。
download_to	是	字符串	指定文件下载后放置的目录，建议使用已经存在的目录。路径长度最大支持256。

- **返回值**
0为成功，其他为失败。

10.5 计算文件的 md5 值

计算文件的md5值。

- **接口调用**
hilens.md5_of_file(file)
- **参数说明**

表 10-2 参数说明

参数名	是否必选	参数类型	描述
file	是	字符串	被计算的文件的路径。

- **返回值**
返回文件的md5值。

10.6 示例-资源管理

资源管理示例如下所示：

```
#!/usr/bin/python3.7
import hilens
import os
```

```
def run():
    # 获得技能工作区目录的路径（末尾带"/"）
    skill_path = hilens.get_workspace_path()

    # 获得技能模型所在目录的路径（末尾带"/"）
    model_path = hilens.get_model_dir()

    # 获得技能配置。如果没有成功获取则返回None
    skill_config = hilens.get_skill_config()
    # 假设技能配置中有名为face_dataset的配置项，其值为obs中的人脸库文件face_dataset.zip的地址
    # 设置技能配置参数可参考《用户指南》相关操作
    face_dataset_url = skill_config["face_dataset"]["value"]
    # 从OBS下载该文件到技能工作区目录，并通过返回值判断是否下载成功
    ret = hilens.download_from_obs(face_dataset_url, hilens.get_workspace_path())
    if ret != 0:
        hilens.error("Failed to download from obs")
        return
    # 在技能工作区目录新建文件夹并解压
    os.system('mkdir '+hilens.get_workspace_path()+face_dataset')
    os.system('unzip '+hilens.get_workspace_path()+face_dataset.zip+' -d '+hilens.get_workspace_path()+face_dataset')

    # 计算文件的md5值
    md5 = hilens.md5_of_file(hilens.get_workspace_path()+face_dataset.zip)

if __name__ == '__main__':
    hilens.init("hello")
    run()
    hilens.terminate()
```

说明

设置技能配置参数的相关步骤如下：

1. 登录华为HiLens控制台，在开发技能时填写运行时配置，即技能在运行时用户需要配置的参数，详情请参见[新建技能](#)。
2. 技能开发完成后，将技能部署至您的设备上，详情请参见[部署和调试技能](#)。
3. 将技能部署至设备上后，可在“技能管理”中设置“运行时配置”的参数，详情请参见[添加运行时配置](#)。

11 难例上传模块

11.1 难例上传介绍及说明

1.1.2固件版本开始支持边缘AI难例发现算法，如果要使用难例上传相关接口，请先升级固件版本到1.1.2，详情请见[升级固件版本](#)。

当前主要支持的难例发现算法如下。

- **图片分类**

CrossEntropyFilter(threshold_cross_entropy)

原理：根据推理结果的交叉熵，判断熵是否小于交叉熵，小于则为难例。

输入：推理结果**prediction classes list**，例如[**class1-score, class2-score, class2-score,....**]，class-score表示类别得分，其范围为[0,1]。

输出：True or False，**True**是难例，**False**是非难例。

- **目标检测**

IBT (image-box-thresholds)

原理：**box_threshold**框阈值用于计算图片难例系数，推理结果的置信度得分小于阈值的数量占总输出推理框的百分比；**img_threshold**图阈值用于判断该图片是否是难例。

输入：**prediction boxes list**，例如[**bbox1, bbox2, bbox3,....**]，其中**bbox = [xmin, ymin, xmax, ymax, score, label]**，x和y为框的坐标，**score**表示置信度得分，**label**表示类别标签，**score**的范围需要为[0,1]。

输出：True or False，**True**是难例，**False**是非难例。

CSF(confidence score filter)

原理：**box_threshold_low**和**box_threshold_up**框阈值用于判断该图片是否是难例，方法是只要有一个输出框置信度得分在区间[**box_threshold_low, box_threshold_up**]，就判断该图片是难例。

输入：**prediction boxes list**，例如[**bbox1, bbox2, bbox3,....**]，其中**bbox = [xmin, ymin, xmax, ymax, score, label]**，x和y为框的坐标，**score**表示置信度得分，**label**表示类别标签，**score**范围为[0,1]。

输出：True or False，**True**是难例，**False**是非难例。

11.2 初始化难例上传模块

构造一个难例过滤器。

- **接口调用**
hilens.HardSample(threshold_one, threshold_two, filter_type)
- **参数说明**

表 11-1 参数说明

参数名	是否必选	参数类型	描述
threshold_one	是	float	阈值。filter_type取不同值时取值不同，详见“filter_type”描述。
threshold_two	是	float	阈值。filter_type取不同值时取值不同，详见“filter_type”描述。
filter_type	是	int	<p>难例过滤器的类型。目前取值支持0, 1, 2, 分别对应难例发现算法“CrossEntropyFilter”、“IBT”和“CSF”，详情请见难例上传介绍及说明。</p> <ul style="list-style-type: none"> • 取0的时候threshold_one为算法“CrossEntropyFilter”的参数“threshold_cross_entropy”，threshold_two可为任意值。 • 取1的时候threshold_one和threshold_two分别对应“IBT”算法的参数“box_threshold”和“img_threshold”。 • 取2的时候threshold_one和threshold_two分别对应“CSF”算法的“box_threshold_low”和“box_threshold_up”。

- **返回值**
返回一个难例过滤器，构造失败的话会抛出异常。

11.3 检测算法中的难例图片判断

对检测结果进行判断。

- 接口调用
hard_sample_detection_filter(inputs)
- 参数说明

表 11-2 参数说明

参数名	是否必选	参数类型	描述
inputs	是	list	检测框，例如[bbox1, bbox2, bbox3,...]，其中bbox = [xmin, ymin, xmax, ymax, score, label]，其中xmin、ymin、xmax、ymax、score为float类型，score取值范围为[0,1]，label为int类型。

- 返回值
返回Bool值，True或False，True表示是难例，False表示不是难例。

11.4 分类算法中的难例图片判断

对分类结果进行判断。

- 接口调用
hard_sample_classification_filter(inputs,input_size)
- 参数说明

表 11-3 参数说明

参数名	是否必选	参数类型	描述
inputs	是	list	类别得分，例如[class1-score, class2-score, class2-score,...]，class-score取值范围为[0,1]。
input_size	是	int	类别数。

- 返回值
返回Bool值，True或False，True表示是难例，False表示不是难例。

11.5 难例图片上传

对识别为难例的图片进行上传。上传的文件名为"model_name-camera_name-index.jpg"，如果当前文件夹有同名文件，将会覆盖，可以在model_name或者camera_name中加入时间戳实现不覆盖。注意：obs文件名长度限制为1024个字符。

- 接口调用
upload_jpeg(upload_url, index, model_name, camera_name, frame)

- 参数说明

表 11-4 参数说明

参数名	是否必选	参数类型	描述
upload_url	是	string	上传难例集的url。 获取难例集的url首先需要在难例上传界面配置相应的数据集，然后通过 get_hard_sample_config 获取难例配置，详情请见 获取难例配置 ，返回值中的参数 dataset_path 对应的是数据集的url。
index	是	int	上传图片的序号。
model_name	是	string	上传图片对应的模型名称。
camera_name	是	string	上传图片对应的摄像头名称。
frame	是	mat	要上传的图片，必须为NV21格式。

- 返回值

返回Bool值，**True**或**False**，**True**表示成功上传。

11.6 获取难例配置

读取难例配置文件，难例配置文件路径为相应技能“data”目录，如“/home/hilens/skills/***/skill_path/data/hardsample_config.json”。

- 接口调用

hilens.get_hard_sample_config()

- 返回值

json格式的难例配置。样例如下。

```
{
  "hard_sample_setting":
  [
    {
      "camera_names": ["123"],
      "data_count": 100,
      "datacur_count": 100,
      "dataset_name": "dataset-a3ae",
      "dataset_path": "https://a.b.csss.obs.cn-north-7.ulanqab.huawei.com/nali/",
      "model_algorithm": "image_classification",
      "model_id": "073c4c8674164307ae300b713a4a050c",
      "model_name": "model-framework5",
      "setting_config":
      {
        "thr": 0.5
      }
    }
  ]
}
```

表 11-5 参数说明

参数名	描述
camera_names	摄像头名称。可在上传的时候区分不同摄像头的的数据。
data_count	总上传图片数。
datacur_count	已经上传的图片数。
dataset_name	要上传到的数据集名称。
dataset_path	要上传的路径，对应的是数据集的url。
model_algorithm	模型的算法类别，一般为分类或检测。
model_id	模型ID。
model_name	模型名称。可在上传的时候区分不同模型的数据。
setting_config	其他设置。可以自定义一些配置，比如下面的阈值。
thr	阈值，可以通过难例上传配置页面下发阈值实现不同场景使用不同的阈值。

11.7 更新难例配置

更新难例配置conf到难例配置文件，并根据输入更新云侧难例上传状态。

- 接口调用
hilens.set_hard_sample_config(conf)
- 参数说明

表 11-6 参数说明

参数名	是否必选	参数类型	描述
conf	是	json	要更新的难例配置。

- 返回值
HiLensEC错误码，0为成功，其他为失败，可参考[错误码](#)。

11.8 示例-难例上传

难例上传示例如下所示：

```
import hilens
import cv2
import numpy as np
```

```

def run():
    # 构造摄像头
    cap = hilens.VideoCapture()

    disp = hilens.Display(hilens.HDMI)

    hard_sample = hilens.HardSample(0.5,0.5,1) # 1,2为检测模型使用的算法
    #hard_sample = hilens.HardSample(0.5,0.5,0) # 0为分类模型使用的算法

    hard_sample_flag = False # 是否存在难例上传配置

    hard_sample_config = hilens.get_hard_sample_config() # 获取难例配置
    if not hard_sample_config:
        hilens.warning("hardSampleConfig is empty")
    else:
        hard_sample_flag = True
        data_count = hard_sample_config["hard_sample_setting"][0]["data_count"]
        data_current_count = hard_sample_config["hard_sample_setting"][0]["datacur_count"]
        upload_jpeg_url = hard_sample_config["hard_sample_setting"][0]["dataset_path"]
        model_name = hard_sample_config["hard_sample_setting"][0]["model_name"]
        camera_name = "default"
        if data_count > data_current_count:
            upload_flag = True # upload_flag是否继续上传
        else:
            upload_flag = False

    while True:
        # 获取一帧画面
        frame = cap.read()
        if hard_sample_flag:
            if upload_flag:
                if hard_sample.hard_sample_detection_filter([[0,0,1280,720,0.4,1]]): # 检测算法的输入为后处理
                    之后的检测框,每个检测框包括[xmin, ymin, xmax, ymax, conf, label] (包括置信度和类别标签)
                    #if hard_sample.hard_sample_classification_filter([0.2, 0.2, 0.2, 0.2, 0.2],5): # 分类算法的输入为各
                    类别的概率,即模型的输出
                        hard_sample.upload_jpeg(upload_jpeg_url, data_current_count, model_name, camera_name,
frame)
                            data_current_count += 1
                            hard_sample_config["hard_sample_setting"][0]["datacur_count"] = data_current_count
                            if data_current_count == 1 or data_current_count == data_count:
                                if data_current_count == data_count:
                                    upload_flag = False
                                hilens.set_hard_sample_config(hard_sample_config) # 更新端侧难例配置文件

            #输出到HDMI
            disp.show(frame)

if __name__ == '__main__':
    #参数【hello】要与基本信息的【检验值】一致。详情请查看开发指南
    hilens.init("hello")
    run()
    hilens.terminate()

```

12 日志模块

12.1 设置打印日志的级别

设置打印日志的级别，默认只打印Info及以上级别的日志。日志级别：Trace -> Debug -> Info -> Warning -> Error -> Fatal

- 接口调用
hilens.set_log_level(level)
- 参数说明

表 12-1 参数说明

参数名称	是否必选	参数类型	参数描述
level	是	枚举类型	hilens.TRACE：打印Trace及以上级别的日志。 hilens.DEBUG：打印Debug及以上级别的日志。 hilens.INFO：打印Info及以上级别的日志。 hilens.WARNING：打印Warn及以上级别的日志。 hilens.ERROR：打印Error及以上级别的日志。 hilens.FATAL：打印Fatal级别的日志。

- 返回值
None

12.2 打印 Trace 级别的日志

输出Trace级别的日志信息并保存至日志文件。

- 接口调用
hilens.trace(msg)
- 参数说明

表 12-2 参数说明

参数名称	是否必选	参数类型	参数描述
msg	是	字符串	Trace级别的日志信息。单条日志支持最大255个字符。

- 返回值
None

12.3 打印 Debug 级别的日志

输出Debug级别的日志信息并保存至日志文件。

- 接口调用
hilens.debug(msg)
- 参数说明

表 12-3 参数说明

参数名称	是否必选	参数类型	参数描述
msg	是	字符串	Debug级别的日志信息。单条日志支持最大255个字符。

- 返回值
None

12.4 打印 Info 级别的日志

输出Info级别的日志信息并保存至日志文件。

- 接口调用
hilens.info(msg)
- 参数说明

表 12-4 参数说明

参数名称	是否必选	参数类型	参数描述
msg	是	字符串	Info级别的日志信息。单条日志支持最大255个字符。

- 返回值
None

12.5 打印 Warning 级别的日志

输出Warn级别的日志信息并保存至日志文件。

- **接口调用**
hilens.warning(msg)
- **参数说明**

表 12-5 参数说明

参数名称	是否必选	参数类型	参数描述
msg	是	字符串	Warning级别的日志信息。单条日志支持最大255个字符。

- **返回值**
None

12.6 打印 Error 级别的日志

输出Error级别的日志信息并保存至日志文件。

- **接口调用**
hilens.error(msg)
- **参数说明**

表 12-6 参数说明

参数名称	是否必选	参数类型	参数描述
msg	是	字符串	Error级别的日志信息。单条日志支持最大255个字符。

- **返回值**
None

12.7 打印 Fatal 级别的日志

输出Fatal级别的日志信息并保存至日志文件。

- **接口调用**
hilens.fatal(msg)
- **参数说明**

表 12-7 参数说明

参数名称	是否必选	参数类型	参数描述
msg	是	字符串	Fatal级别的日志信息。单条日志支持最大255个字符。

- 返回值
None

12.8 示例-日志

打印日志示例如下所示：

```
#!/usr/bin/python3.7

import hilens

def run():
    # 设置日志级别
    hilens.set_log_level(hilens.DEBUG)

    # 打印一条trace级别的日志
    hilens.trace("trace")

    # 打印一条debug级别的日志
    hilens.debug("debug")

    # 打印一条info级别的日志
    hilens.info("info")

    # 打印一条warning级别的日志
    hilens.warning("warning")

    # 打印一条error级别的日志
    hilens.error("error")

    # 打印一条fatal级别的日志
    hilens.fatal("fatal")

if __name__ == '__main__':
    hilens.init("hello")
    run()
    hilens.terminate()
```

13 错误码

HiLens Framework以枚举类型返回错误码，当调用接口发生错误并返回错误码时，可以查看以下枚举类来获取错误信息：

表 13-1 错误码

错误码	说明
UNKNOWN_ERROR	未知错误。
INIT_CURL_ERROR	初始化CURL错误。
CREATE_DIR_FAILED	创建文件夹失败。
OPENFILE_FAILED	打开文件失败。
RENAME_FAILED	重命名失败。
ACCESS_FILE_FAILED	文件不存在或无文件访问权限。
INVALID_BUF	无效的BUF。
COULDNT_RESOLVE_HOST	无法解析，查看网络是否通畅。
WRITE_ERROR	写错误，检查下载目录是否有写权限，空间是否足够。
TIMEOUT	请求超时。
AUTH_FAILED	认证信息错误，检查ak, sk, token是否有效。
NOT_FOUND	没有这个对象。
SERVER_ERROR	服务端内部错误。
OBJECT_CONFLICT	对象冲突。
APPEND_FAILED	追加失败（比如追加到不可追加的对象上）。
HIAI_SEND_DATA_FAILED	hiai engine发送数据失败，请根据日志来分析具体情况。

错误码	说明
HIAI_INFER_ERROR	hiai engine推理错误，请根据日志来分析具体情况（可能是实际输入大小与模型的输入大小不匹配）。
INVALID_SRC_SIZE	图片处理，src尺寸不符合约束条件。
INVALID_DST_SIZE	图片处理，dst尺寸不符合约束条件。
MPP_PROCESS_FAILED	mpp处理图片失败。
WEBSOCKET_ERROR	WebSocket错误。
CONFIG_FILE_ERROR	配置文件错误。
INVALID_PARAM	参数有误。