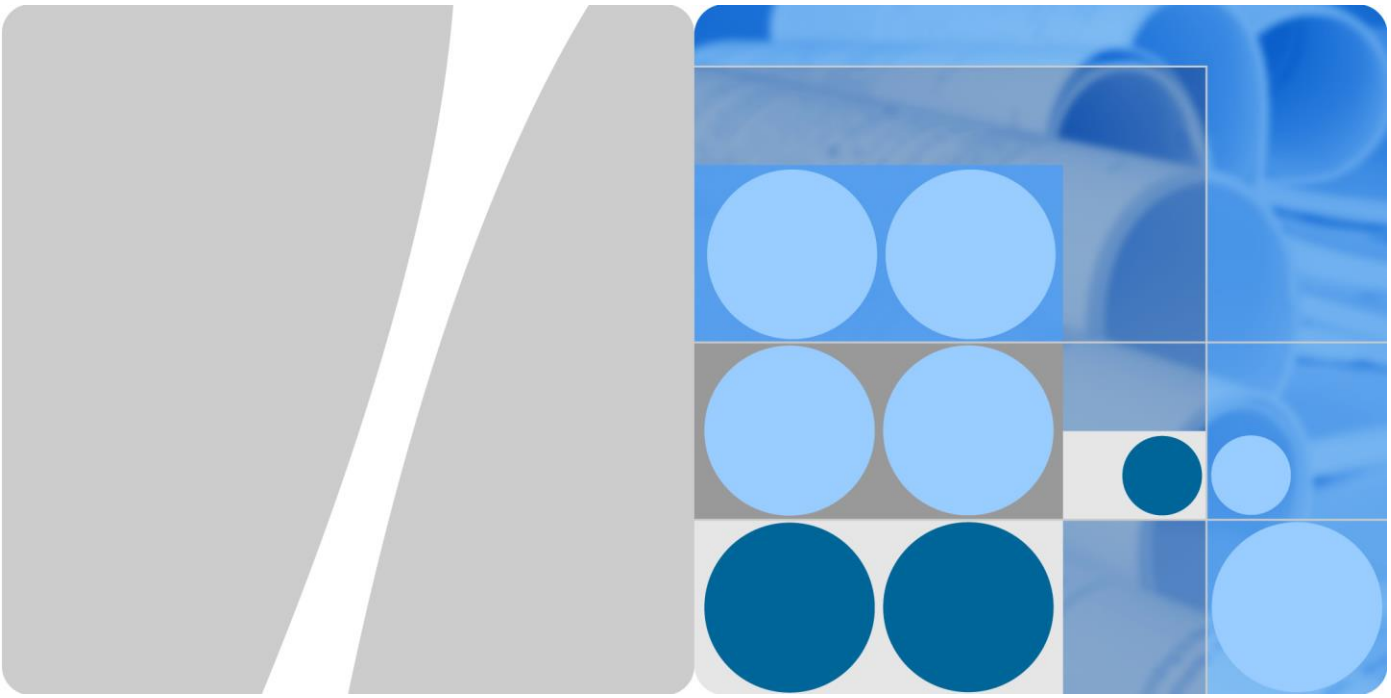


HiAI DDK V320

FAQ

文档版本 03

发布日期 2020-02-28

华为技术有限公司

华为 HiAI 申请方式

发送申请邮件到邮箱：developer@huawei.com

邮件名称：HUAWEI HiAI+公司名称+产品名称

邮件正文：合作公司+联系人+联系方式+联系邮箱

我们将在收到邮件的 5 个工作日内邮件给您反馈结果，请您注意查收。

官网地址 https://developer.huawei.com/consumer/cn/

前言

修改记录

发布日期	文档版本	修改说明
2020-02-28	03	新增 SO 编译方式说明
2019-12-31	02	新增 V320 版本内容
2019-09-04	01	新增 V310 版本内容

目 录

前言 ii

1 HiAI DDK 使用 FAQ 1

1.1 判断当前手机是否支持 NPU 1
1.2 如何统计前向计算时间 2
1.3 如何选择使用“同步”接口还是“异步”接口 2
1.4 推理时如何将原图转换为模型输入数据 2
1.5 推理函数支持的数据格式 4
1.6 SO 编译方式说明 4
1.7 OMG 离线模型输出算子类型错误处理 4

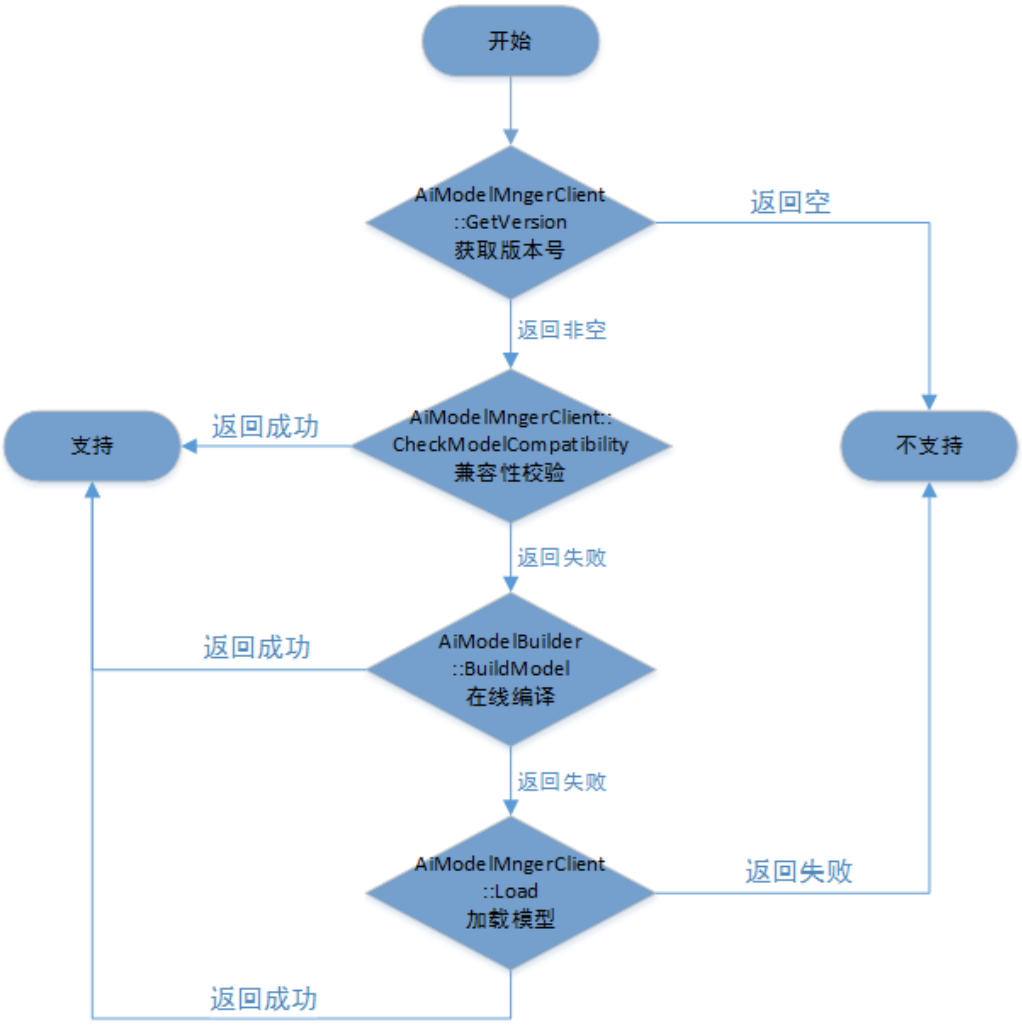
插图目录

图 1-1 输出算子类型修改前（左）和修改后（右） 5

1 HiAI DDK 使用 FAQ

1.1 判断当前手机是否支持 NPU

答：通过调用 AiModelMngerClient 的 GetVersion、CheckModelCompatibility、Load 方法以及 AiModelBuilder 的 BuildModel 方法来判断，流程如下：



如果 inference_npu_demo 提示模型不兼容无法运行时，可集成 libcpucl.so 将模型运行在 CPU 上，具体可参考 inference_cpu_demo 的 Android.mk。

1.2 如何统计前向计算时间

答：同步情况：在 jni 层 process 接口前后加时间打印。可参考 app_sample\inference_demo\Demo_Soure_Code\app\src\main\jni\classify_jni.cpp

```
341     // before process
342     struct timeval tpstart, tpend;
343     gettimeofday(&tpstart, nullptr);
344     int istamp;
345     int ret = g_clientSync->Process(context, input_tensor, output_tensor, 1000, istamp);
346     if (ret) {
347         LOGE("[HIAI_DEMO_SYNC] Runmodel Failed!, ret=%d\n", ret);
348         return nullptr;
349     }
350
351     // after process
352     gettimeofday(&tpend, nullptr);
353     float time_use = 1000000 * (tpend.tv_sec - tpstart.tv_sec) + tpend.tv_usec - tpstart.tv_usec;
354
355     time_use_sync =  time_use / 1000;
```

异步情况：在回调函数和 OnProcessDone 接口加时间打印，单次计算时间为两者时间差；多次计算时间为回调函数之间的时间差，可参考 app_sample\inference_demo\Demo_Soure_Code\app\src\main\jni\classify_async_jni.cpp

```
55  void JNIListener::OnProcessDone(const AiContext &context, int result1, const vector<shared_ptr<AiTensor>> &output_tensor, int32_t istamp)
56  {
57      std::unique_lock<std::mutex> lock(mutex_map);
58      map_input_tensor.erase(istamp);
59      condition_.notify_all();
60
61      gettimeofday(&tpend, nullptr);
62      time_use = 1000000 * (tpend.tv_sec - tpstart.tv_sec) + tpend.tv_usec - tpstart.tv_usec;
63      LOGI("[HIAI_DEMO_ASYNC] AYSNC infrence time %f ms.", time_use / 1000);
64      LOGE("[HIAI_DEMO_ASYNC] AYSNC JNI layer onRunDone istamp: %d", istamp);
65
66      JNIEnv *env = nullptr;
```

1.3 如何选择使用“同步”接口还是“异步”接口

答：由业务场景决定，目前两者均支持，从性能考虑推荐使用异步接口。

1.4 推理时如何将原图转换为模型输入数据

答：推理时将外部传入图片按照 argb 的格式存储，非 Aipp 场景下，按 NCHW 排序后作为模型输入数据；Aipp 场景下，按 NCHW 排序后并转成 YUV 作为模型输入数据。

input 数据处理代码可参考 app_sample\inference_demo\Demo_Soure_Code\app\src\main\java\com\huawei\hiaidemo\view\NpuClassifyActivity.java

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (resultCode == RESULT_OK && data != null) switch (requestCode) {
        case GALLERY_REQUEST_CODE:
            try {
                Bitmap bitmap;
                ContentResolver resolver = getContentResolver();
                Uri originalUri = data.getData();
                bitmap = MediaStore.Images.Media.getBitmap(resolver, originalUri);
                String[] proj = {MediaStore.Images.Media.DATA};
                Cursor cursor = managedQuery(originalUri, proj,  selection: null,  selectionArgs: null,  sortOrder: null);
                cursor.moveToFirst();
                Bitmap rgba = bitmap.copy(Bitmap.Config.ARGB_8888,  isMutable: true);
                initClassifiedImg = Bitmap.createScaledBitmap(rgba,  selectedModel.getInput_W(),  selectedModel.getInput_H(),  filter: true);
                byte[] inputData = {};
                if(selectedModel.getUseAIPP()){
                    inputData = Utils.getPixelsAIPP(selectedModel.getFramework(),initClassifiedImg,selectedModel.getInput_W(),selectedModel.getInput_H());
                }else {
                    inputData = Utils.getPixels(selectedModel.getFramework(),initClassifiedImg,selectedModel.getInput_W(),selectedModel.getInput_H());
                }
                ArrayList<byte[]> inputDataList = new ArrayList<>();
                inputDataList.add(inputData);
                Log.d(TAG,  msg: "inputData.length  1 is :"+inputData.length+"");
                runModel(selectedModel,inputDataList);
            } catch (IOException e) {
                Log.e(TAG,  e.toString());
            }
```

处理细节可参考同层目录下 Utils.java，如下：

```
    public static byte[] getPixels(String framework, Bitmap bitmap,
                                   int resizedWidth, int resizedHeight) {
        int channel = 1;
        float[] buff = new float[channel * resizedWidth * resizedHeight];

        int rIndex;
        int gIndex;
        int bIndex;
/*...*/
        int pixCount = channel * resizedWidth * resizedHeight;
        byte[] ret = new byte[pixCount * 4];

        for (int i = 0; i < pixCount; ++i) {
            int int_bits = Float.floatToIntBits(buff[i]);
            ret[(i * 4) + 0] = (byte) int_bits;
            ret[(i * 4) + 1] = (byte) (int_bits >>> 8);
            ret[(i * 4) + 2] = (byte) (int_bits >>> 16);
            ret[(i * 4) + 3] = (byte) (int_bits >>> 24);
        }

        return ret;
    }
```

```
111     public static byte[] getPixelsAIPP(String framework,Bitmap bitmap, int resizedWidth, int resizedHeight){
112         Log.i(TAG, msg: "resizedWidth : " + resizedWidth + " resizedHeight : " + resizedHeight);
113         return getNV12(resizedWidth,resizedHeight, bitmap);
114     }
115
116 @   private static byte [] getNV12(int inputWidth, int inputHeight, Bitmap scaled) {
117         // Reference (Variation) : https://gist.github.com/wobbals/5725412
118
119         int [] argb = new int[inputWidth * inputHeight];
120
121         Log.i(TAG, msg: "scaled : " + scaled);
122         scaled.getPixels(argb, offset: 0, inputWidth, x: 0, y: 0, inputWidth, inputHeight);
123
124         byte [] yuv = new byte[inputWidth*inputHeight*3/2];
125         encodeYUV420SP(yuv, argb, inputWidth, inputHeight);
126
127         //scaled.recycle();
128
129         return yuv;
130     }
131
132     private static void encodeYUV420SP(byte[] yuv420sp, int[] argb, int width, int height) {
```

1.5 推理函数支持的数据格式

答：process 推理时传入的 input_tensor 的数据格式当前只支持 nchw。

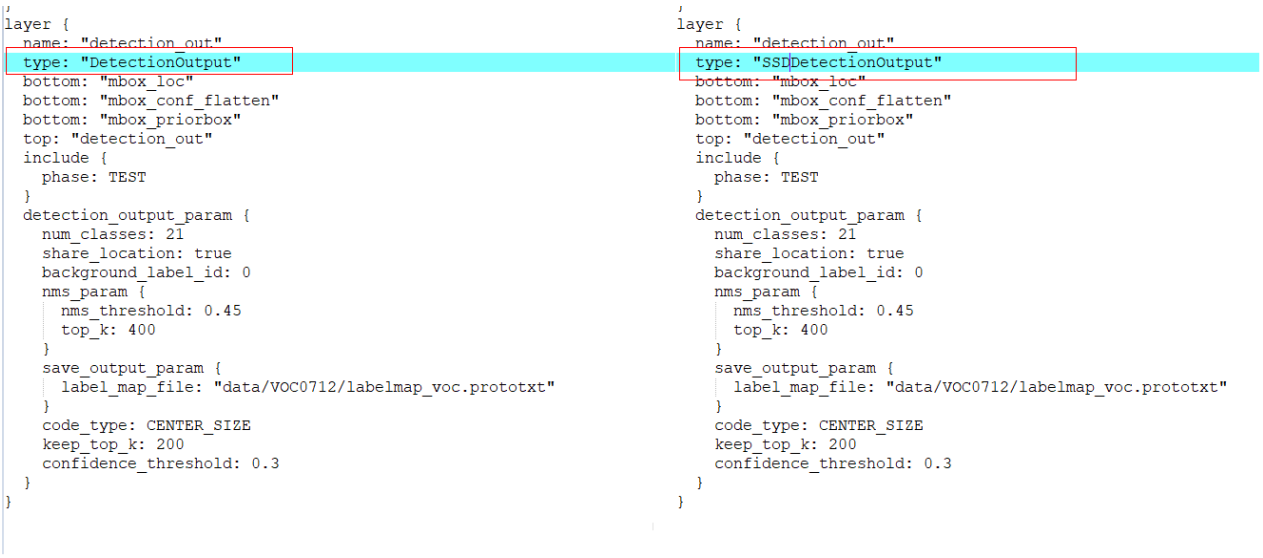
1.6 SO 编译方式说明

答：DDK hiai 对外接口的实现封装在动态库 libhiai*.so 中，对于标准库 c++_shared 均为动态依赖（APP_STL := c++_shared）,为了避免因标准库版本不同引起的问题，建议用户在编译自己所需的 so 时，对于标准库请使用动态依赖的方式进行编译。

1.7 OMG 离线模型输出算子类型错误处理

答：Caffe 网络中具有相同类型名但计算功能不同的层。比如 DetectionOutput 层，需要使用算子映射显式的指明为 FSRDetectionOutput、SSDDetectionOutput 等检测算子类型，否则 OMG 生成离线模型会执行失败。用户可以在 omg 命令中加入--op_name_map 参数指定，参考《华为 HIAI_DDK_V320_OMG 工具使用说明》文档中“整体参数 --op_name_map 参数设置”，也可以直接在原始网络 proto 模型文件中将输出算子类型显式指定为 SSDDetectionOutput 等算子类型，如下图所示。以上两种方案二选一即可。

图1-1 输出算子类型修改前（左）和修改后（右）



```
layer {
  name: "detection_out"
  type: "DetectionOutput"
  bottom: "mbox_loc"
  bottom: "mbox_conf_flatten"
  bottom: "mbox_priorbox"
  top: "detection_out"
  include {
    phase: TEST
  }
  detection_output_param {
    num_classes: 21
    share_location: true
    background_label_id: 0
    nms_param {
      nms_threshold: 0.45
      top_k: 400
    }
    save_output_param {
      label_map_file: "data/VOC0712/labelmap_voc.prototxt"
    }
    code_type: CENTER_SIZE
    keep_top_k: 200
    confidence_threshold: 0.3
  }
}
```

```
layer {
  name: "detection_out"
  type: "SSDDetectionOutput"
  bottom: "mbox_loc"
  bottom: "mbox_conf_flatten"
  bottom: "mbox_priorbox"
  top: "detection_out"
  include {
    phase: TEST
  }
  detection_output_param {
    num_classes: 21
    share_location: true
    background_label_id: 0
    nms_param {
      nms_threshold: 0.45
      top_k: 400
    }
    save_output_param {
      label_map_file: "data/VOC0712/labelmap_voc.prototxt"
    }
    code_type: CENTER_SIZE
    keep_top_k: 200
    confidence_threshold: 0.3
  }
}
```

