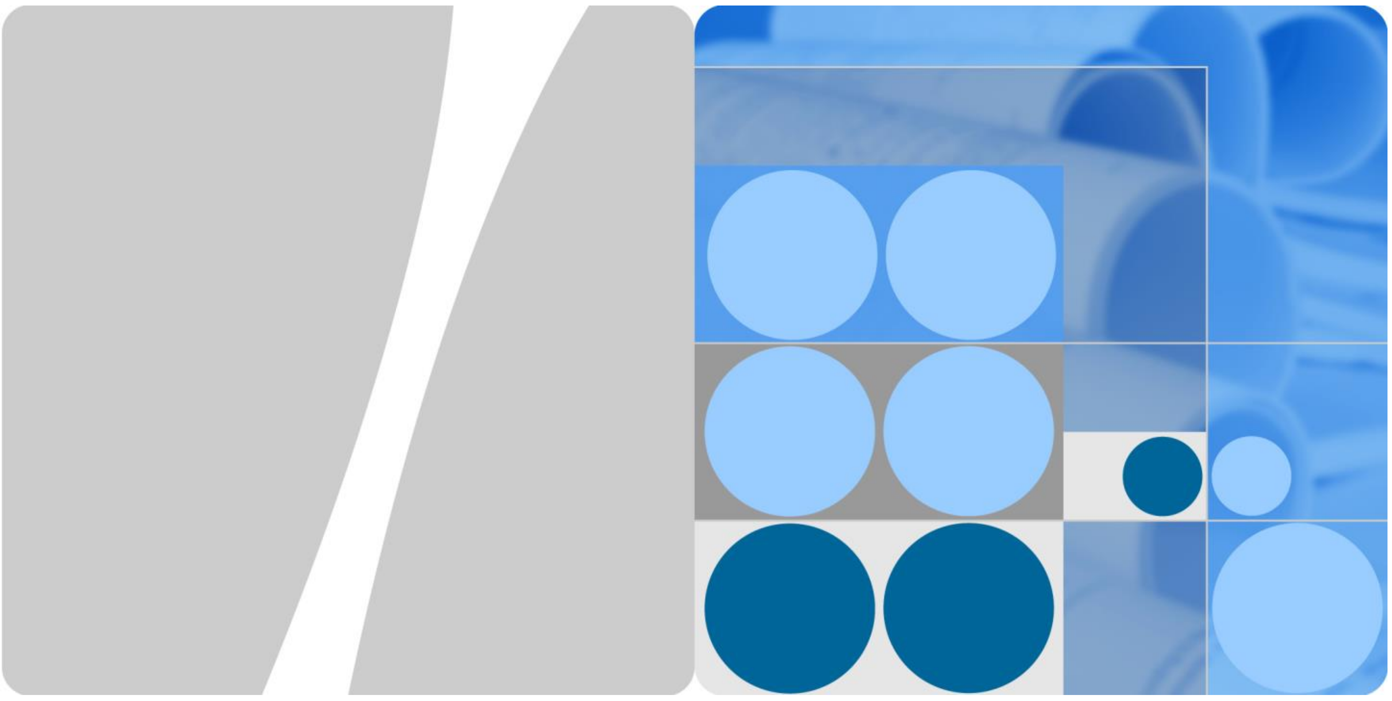


HiAI DDK V320

FAQs

Issue	**03**
Date	**2020-02-28**

HUAWEI TECHNOLOGIES CO., LTD.

Copyright © Huawei Technologies Co., Ltd. 2019. All rights reserved.

No part of this document may be reproduced or transmitted in any form or by any means without prior written consent of Huawei Technologies Co., Ltd.

Trademarks and Permissions

HUAWEI and other Huawei trademarks are trademarks of Huawei Technologies Co., Ltd.

All other trademarks and trade names mentioned in this document are the property of their respective holders.

Notice

The purchased products, services and features are stipulated by the contract made between Huawei and the customer. All or part of the products, services and features described in this document may not be within the purchase scope or the usage scope. Unless otherwise specified in the contract, all statements, information, and recommendations in this document are provided "AS IS" without warranties, guarantees or representations of any kind, either express or implied.

The information in this document is subject to change without notice. Every effort has been made in the preparation of this document to ensure accuracy of the contents, but all statements, information, and recommendations in this document do not constitute a warranty of any kind, express or implied.

Huawei HiAI Application

Send an application email to developer@huawei.com.

Email subject: HUAWEI HiAI + Company name + Product name

Email body: Cooperation company + Contact person + Phone number + Email address

We will reply to you within 5 working days.

Official website: https://developer.huawei.com/consumer/en/

Change History

Changes between document issues are cumulative. The latest document issue contains all the changes made in earlier issues.

Date	Version	Change Description
2020-02-28	03	Added the description of the .so library compilation mode.
2019-12-31	02	Added the description of HiAI DDK V320.
2019-09-04	01	Added the description of HiAI DDK V310.

Contents

Change History .... **i**

1 HiAI DDK FAQs **1**

1.1 How Do I Check Whether My Smartphone Supports the NPU? 1

1.2 How Do I Check the Time Consumption of Forward Computation? 2

1.3 How Do I Select the Sync or Async APIs? 3

1.4 How the Input Image Format Is Converted Before Model Inference? 3

1.5 What Are the Data Formats Supported by the Inference Function? 6

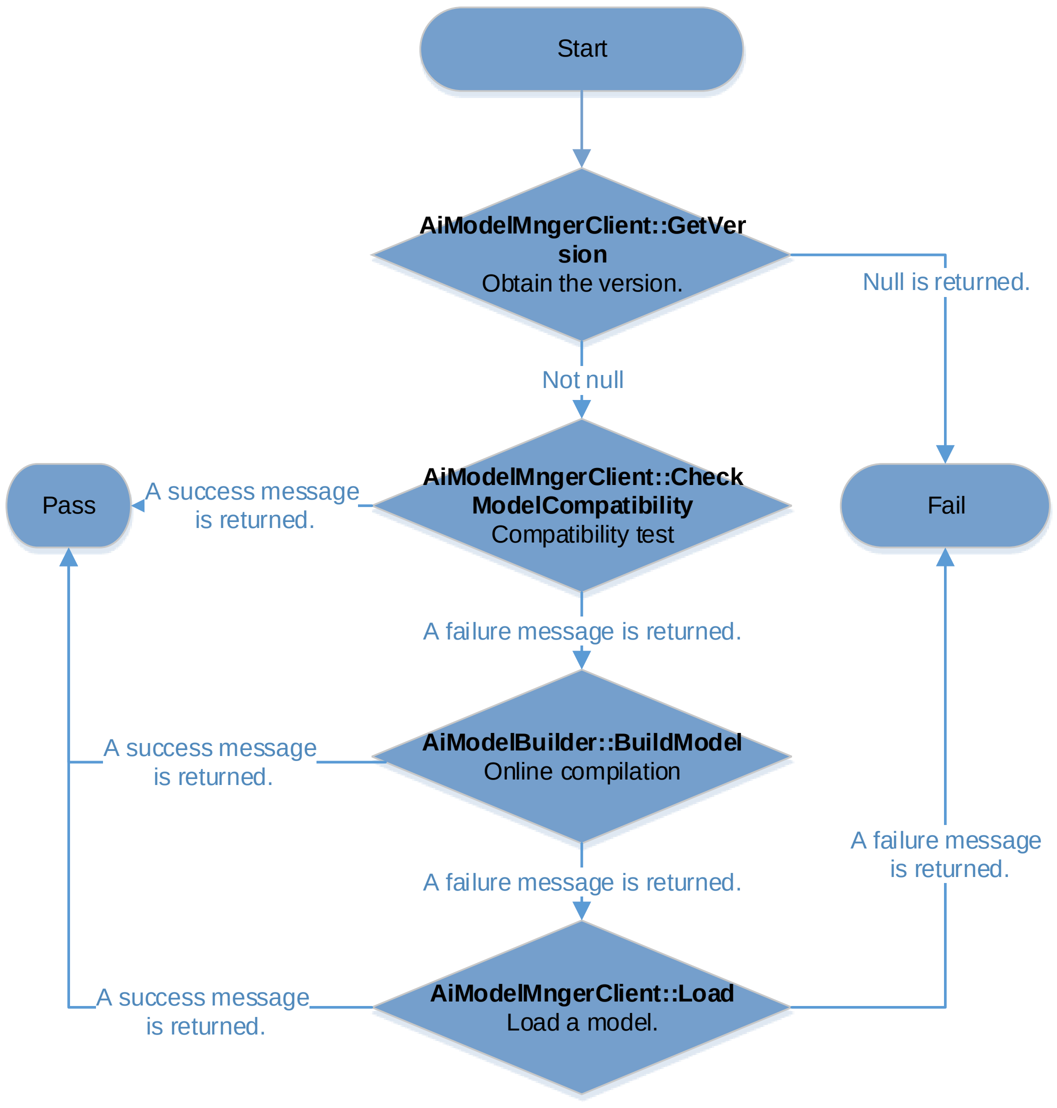
1.6 How Do I Compile .so Libraries? 6

1.7 What Do I Do If Type of the Operator Output by the OMG Offline Model Is Incorrect? 6

1 HiAI DDK FAQs

1.1 How Do I Check Whether My Smartphone Supports the NPU?

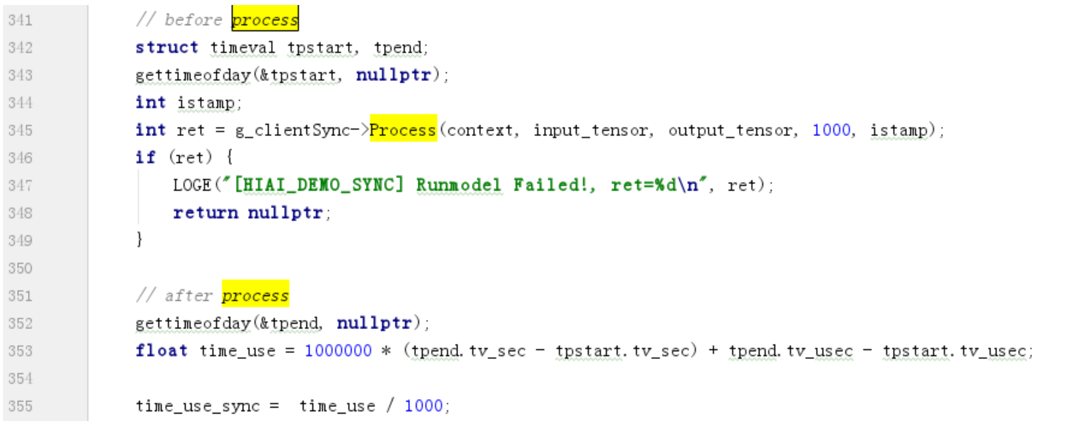
Call the **GetVersion**, **CheckModelCompatibility**, and **Load** methods of class AiModelMngerClient and the **BuildModel** method of class AiModelBuilder as follows.


Start
AiModelMngerClient::GetVersion
Obtain the version.
Null is returned.
Not null
AiModelMngerClient::CheckModelCompatibility
Compatibility test
A success message is returned.
Pass
Fail
A failure message is returned.
AiModelBuilder::BuildModel
Online compilation
A success message is returned.
A failure message is returned.
A failure message is returned.
AiModelMngerClient::Load
Load a model.
A success message is returned.

If running **inference_npu_demo** fails due to a model incompatibility error, inlcude **libcpucl.so** to the demo to run the model on the CPU. For details, see **Android.mk** in **inference_cpu_demo**.

1.2 How Do I Check the Time Consumption of Forward Computation?

- In sync mode, add timestamp print before and after the process API at the JNI layer. For details, see **app_sample\inference_demo\Demo_Soure_Code\app\src\main\jni\classify_jni.cpp**.

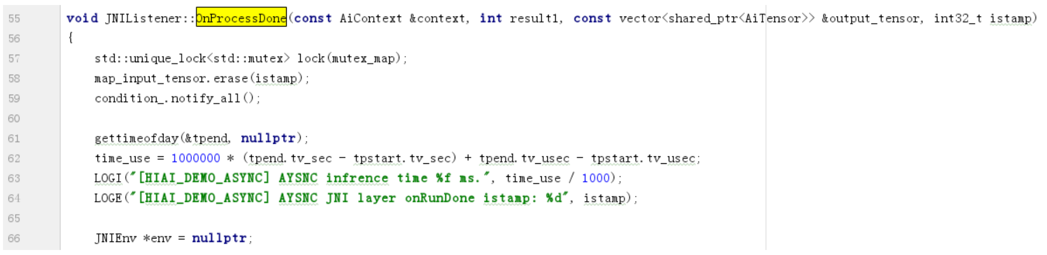


```
// before process
struct timeval tpstart, tpend;
gettimeofday(&tpstart, nullptr);
int istamp;
int ret = g_clientSync->Process(context, input_tensor, output_tensor, 1000, istamp);
if (ret) {
    LOGE("[HIAI_DEMO_SYNC] Runmodel Failed!, ret=%d\n", ret);
    return nullptr;
}

// after process
gettimeofday(&tpend, nullptr);
float time_use = 1000000 * (tpend.tv_sec - tpstart.tv_sec) + tpend.tv_usec - tpstart.tv_usec;

time_use_sync =  time_use / 1000;
```

- In async mode, add timestamp print to the callback function and the OnProcessDone API. If the callback is called only once, the time difference between the callback function and the OnProcessDone API is calculated. If the callback is called more than once, the time difference between the callbacks is calculated. For details, see **app_sample\inference_demo\Demo_Soure_Code\app\src\main\jni\classify_async_jni.cpp**.



```
void JNIListener::OnProcessDone(const AiContext &context, int result1, const vector<shared_ptr<AiTensor>> &output_tensor, int32_t istamp)
{
    std::unique_lock<std::mutex> lock(mutex_map);
    map_input_tensor.erase(istamp);
    condition_.notify_all();

    gettimeofday(&tpend, nullptr);
    time_use = 1000000 * (tpend.tv_sec - tpstart.tv_sec) + tpend.tv_usec - tpstart.tv_usec;
    LOGI("[HIAI_DEMO_ASYNC] AYSNC infrence time %f ms.", time_use / 1000);
    LOGE("[HIAI_DEMO_ASYNC] AYSNC JNI layer onRunDone istamp: %d", istamp);

    JNIEnv *env = nullptr;
```

1.3 How Do I Select the Sync or Async APIs?

Both are supported, depending on the service requirements. However, async APIs are recommended for better performance. (Theoretically, if a callback function is to be called, async APIs should be used.)

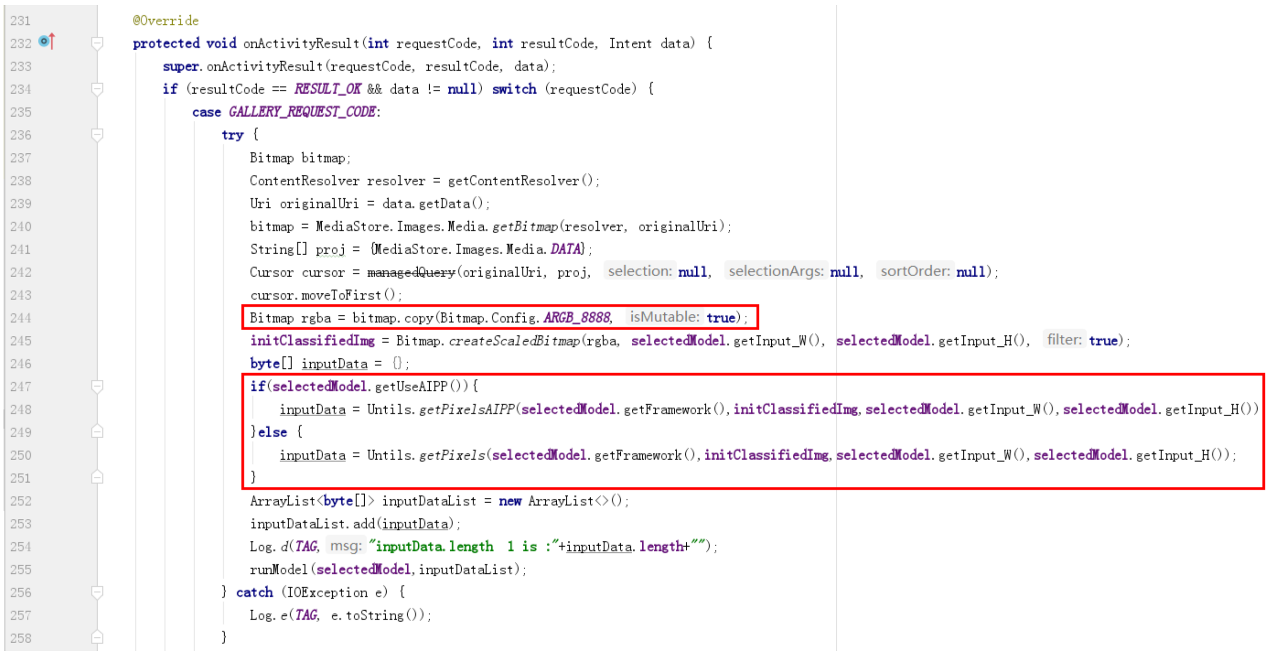
1.4 How the Input Image Format Is Converted Before Model Inference?

During inference, source images are stored in ARGB format.

In non-AIPP scenarios, the images are laid out according to NCHW before being input to the model.

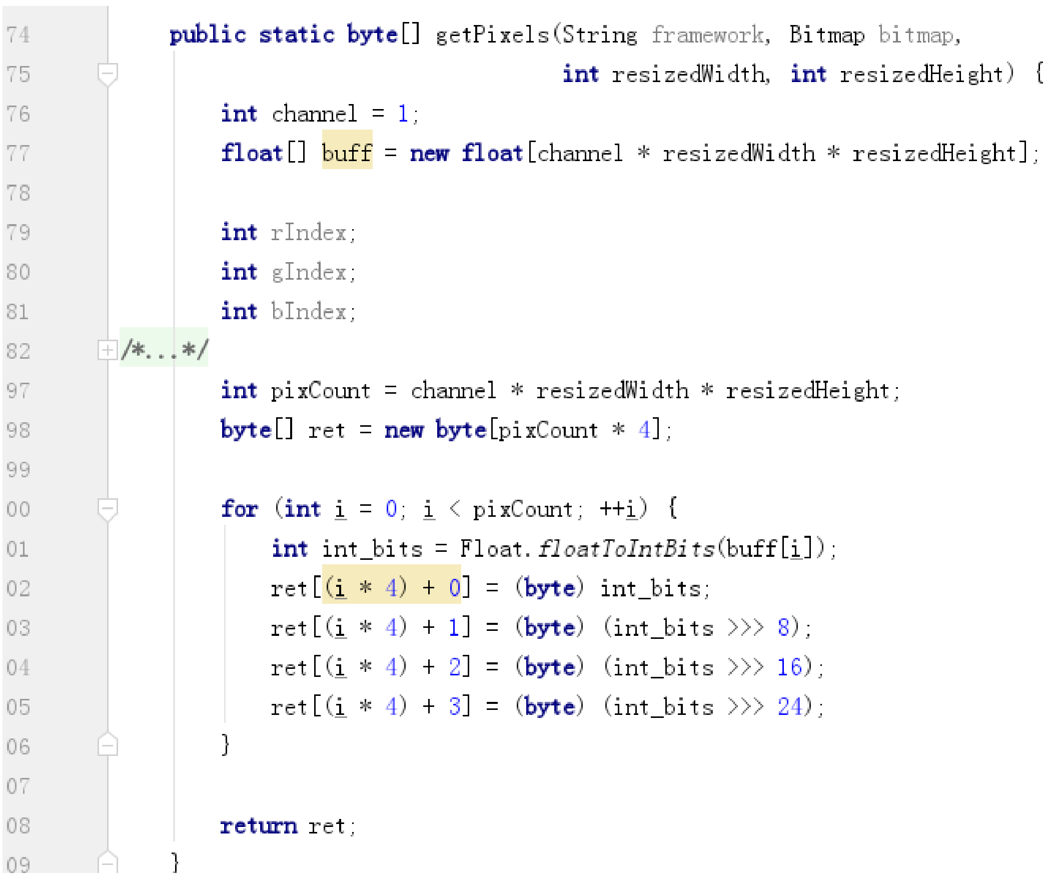
In AIPP scenarios, the images are laid out according to NCHW and then converted into YUV images before being input to the model.

For details about the format conversion code, see **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 = Untils.getPixelsAIPP(selectedModel.getFramework(),initClassifiedImg,selectedModel.getInput_W(),selectedModel.getInput_H())
                }else {
                    inputData = Untils.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());
            }
```

For details, see the **Utils.java** file in the same directory.

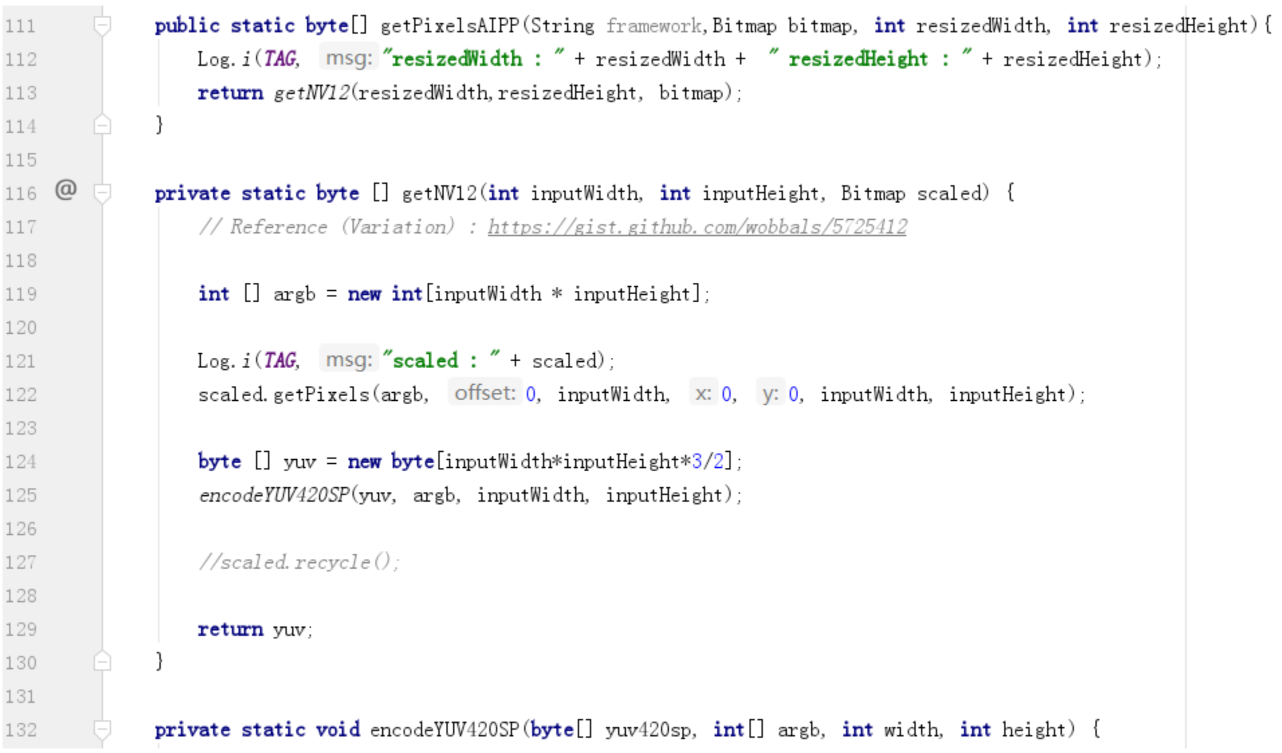


```
    public static byte[] getPixels(String framework, Bitmap bitmap,
                                   int resizedWidth, int resizedHeight) {
        int channel = 1;
        float[] buff = new float[channel * resizedWidth * resizedHeight];

        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;
    }
```



```
    public static byte[] getPixelsAIPP(String framework,Bitmap bitmap, int resizedWidth, int resizedHeight){
        Log.i(TAG, msg: "resizedWidth : " + resizedWidth + " resizedHeight : " + resizedHeight);
        return getNV12(resizedWidth,resizedHeight, bitmap);
    }

    private static byte [] getNV12(int inputWidth, int inputHeight, Bitmap scaled) {
        // Reference (Variation) : https://gist.github.com/wobbals/5725412

        int [] argb = new int[inputWidth * inputHeight];

        Log.i(TAG, msg: "scaled : " + scaled);
        scaled.getPixels(argb, offset: 0, inputWidth, x: 0, y: 0, inputWidth, inputHeight);

        byte [] yuv = new byte[inputWidth*inputHeight*3/2];
        encodeYUV420SP(yuv, argb, inputWidth, inputHeight);

        //scaled.recycle();

        return yuv;
    }

    private static void encodeYUV420SP(byte[] yuv420sp, int[] argb, int width, int height) {
```

1.5 What Are the Data Formats Supported by the Inference Function?

The data format of **input_tensor** configured in the **process** API must be NCHW.

1.6 How Do I Compile .so Libraries?

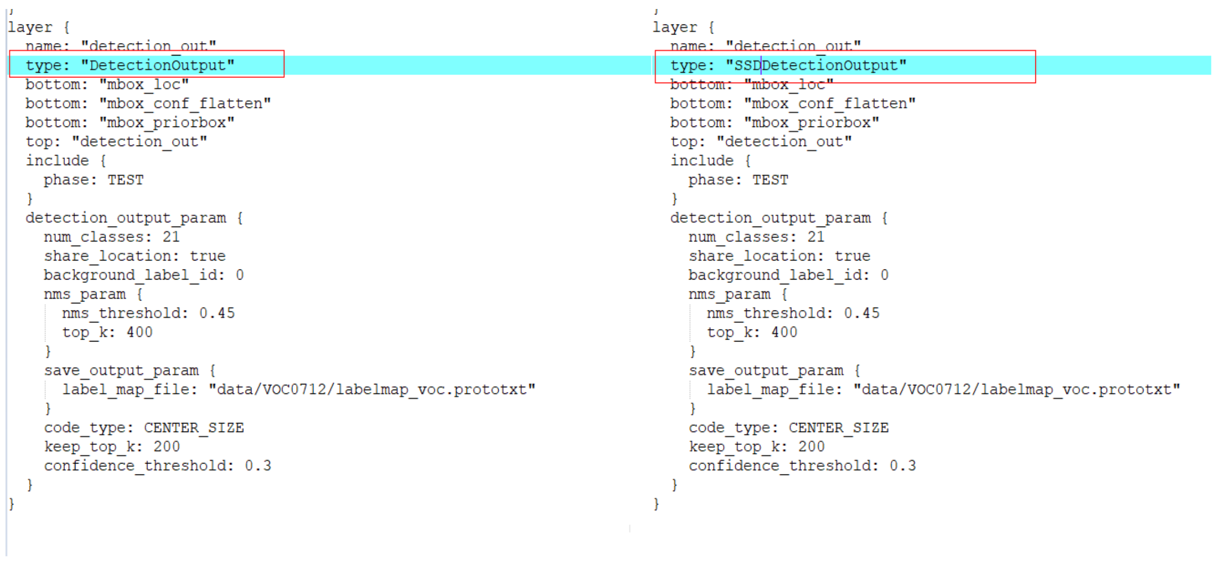
The DDK HiAI external APIs are encapsulated in the dynamic library **libhiai*.so**.

The standard library **c++_shared** is a dynamic dependency (**APP_STL := c++ _shared**). To avoid problems caused by version differences, you are advised to compile the standard library as a dynamic library.

1.7 What Do I Do If Type of the Operator Output by the OMG Offline Model Is Incorrect?

In the Caffe network, some layers of the same type serve different computing purposes. For example, at the DetectionOutput layer, you need to explicitly specify the detection operator types such as FSRDetectionOutput and SSDDetectionOutput through operator mapping. Otherwise, the OMG fails to generate the offline model. You can include the **--op_name_map** parameter to the OMG command. For details, see section "General Parameters" in *Huawei HiAI DDK V320 OMG Tool Instructions*. Alternatively, you can explicitly specify the output operator type in the .proto model file of the original network, for example, SSDDetectionOutput, as shown in the following figure.

Figure 1-1 Output operator type before and after modification



```
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
  }
}
```