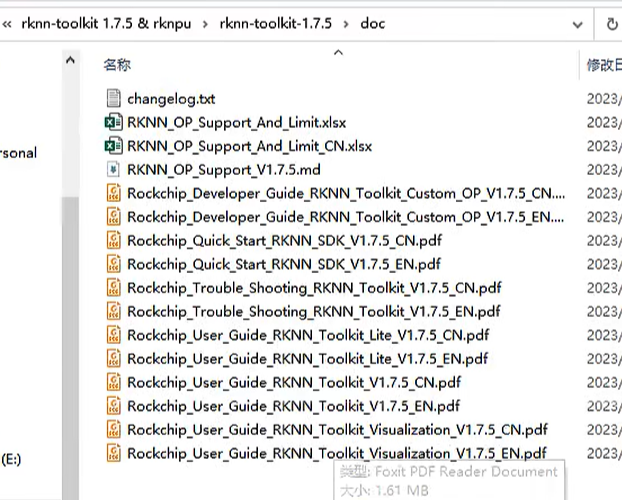
这讲课程我们通过前面的RKNN Toolkit先初步了解模型的转换步骤、操作过程。

也就是调用RKNN Toolkit的api实现模型的转换。由于RKNN Toolkit工具支持的深度学习框架比较多，我们主要以TensorFlow、PyTorch下得到的模型为例子讲解怎么实现模型的初步转换。

如果是其它深度学习模型的话，直接看RKNN Toolkit的手册API的介绍，以及RKNN Toolkit的示例来实现转化模型。Api的介绍就需要看RKNN的手册：

见目录：

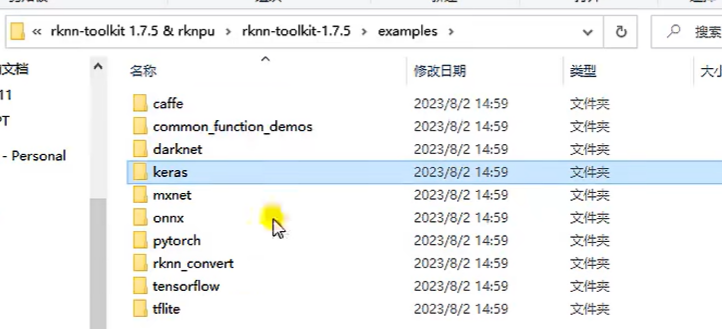
C:\Users\zhongqing\Desktop\笔记\RK3568\_AI\正点原子RK3568 、RK3588 AI开发板\2-基于Anaconda安装RKNN Toolkit\rknn-toolkit2 1.5.2 & rknpu2\rknn-toolkit2-1.5.2\rknn-toolkit2-1.5.2\doc



"C:\Users\zhongqing\Desktop\笔记\RK3568\_AI\正点原子RK3568 、RK3588 AI开发板\2-基于Anaconda安装RKNN Toolkit\rknn-toolkit2 1.5.2 & rknpu2\rknn-toolkit2-1.5.2\rknn-toolkit2-1.5.2\doc\Rockchip\_User\_Guide\_RKNN\_Toolkit2\_CN-1.5.2.pdf"

就是介绍的RKNN 的API.转换其它模型的时候就可以参考这个api介绍。来使用加载模型。示例代码就是看：

C:\Users\zhongqing\Desktop\笔记\RK3568\_AI\正点原子RK3568 、RK3588 AI开发板\2-基于Anaconda安装RKNN Toolkit\rknn-toolkit2 1.5.2 & rknpu2\rknn-toolkit2-1.5.2\rknn-toolkit2-1.5.2\examples

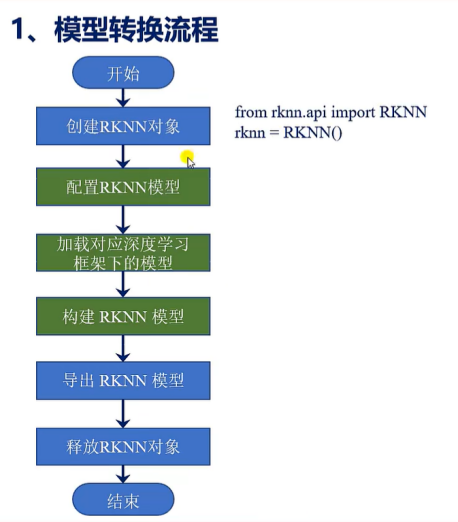


本节课内容：



/\*

1. 模型转换流程：



/\*

首先，在调用RKNN Toolkit的接口之前，我们先通过RKNN的方法来创建一个RKNN对象，后期的API调用都是通过RKNN对象来完成的。创建RKNN对象的方法就是：

from rknnapi import RKNN

Rknn = RKNN()

\*/

/\*

创建好RKNN对象之后，就是配置RKNN模型，配置的目的 就是为了后面构建RKNN模型做准备。 他就是调用rknn.config()这个api来完成模型配置的，在配置RKNN模型的阶段一般就是对模型的输入进行预处理，如：对输入的数据进行归一化操作/标准化操作（这个时候就要确定输入的图片RGB的均值是多少 以及RGB标准差是多少，这些参数确定之后就可以把输入的数据归一化到一定的范围内）、通道顺序的调整（确定是RGB还是BGR）、量化类型选择、量化参数优化算法选择、模型优化等级配置、指定RKNN模型目标运行平台等。

/\*

归一化操作：#新数据＝（原数据-均值）/标准差

归一化的值=(X-mean\_values)/std\_values

通道顺序的调整：0表示R，1表示G，2表示B

‘0 1 2’表示RGB，'2 1 0’表示BGR

rknn-toolkit V1.7.5支持3种量化类型：

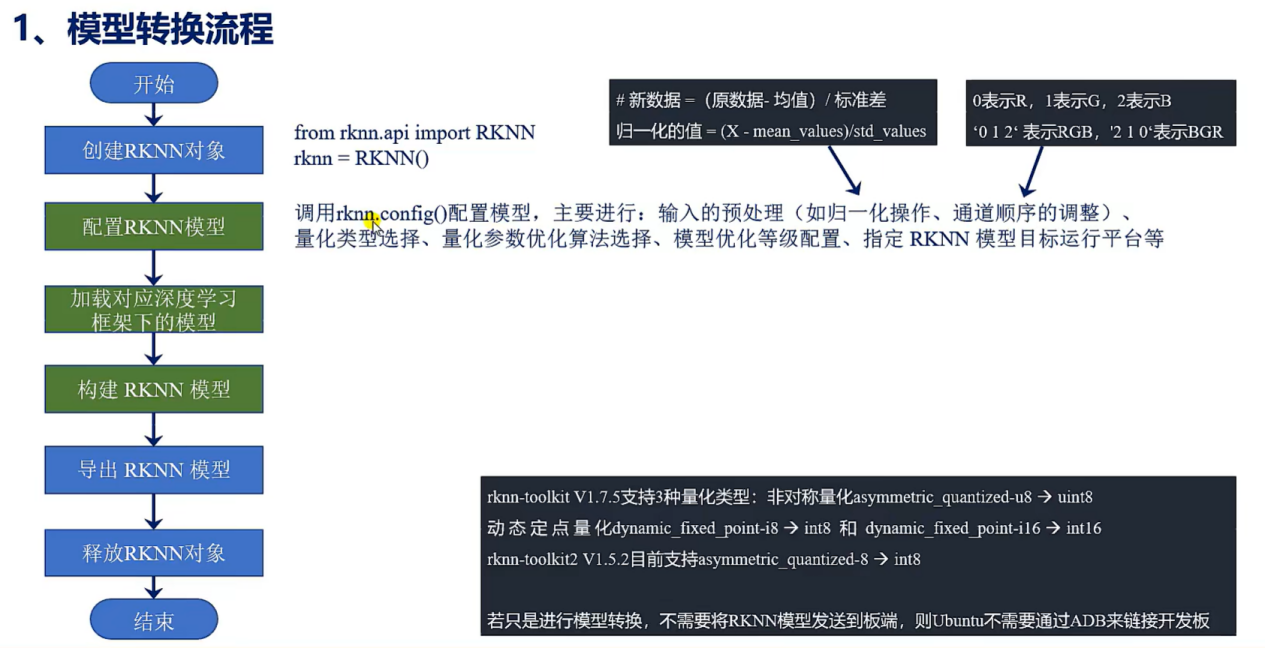
非对称量化asymmetric\_quantized-u8 -> uint8 （默认）

动态定点量化dynamic\_fixed\_point-i8 -> int8

和动态定点量化dynamic\_fixed\_point-i16 -> intl6

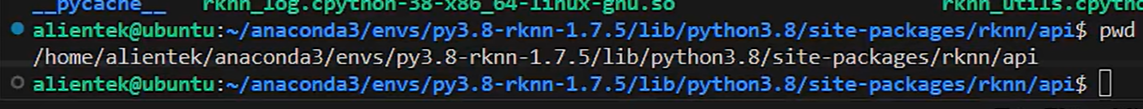
rknn-toolkit2 V1.5.2目前支持asymmetric\_quantized-8 -> int8

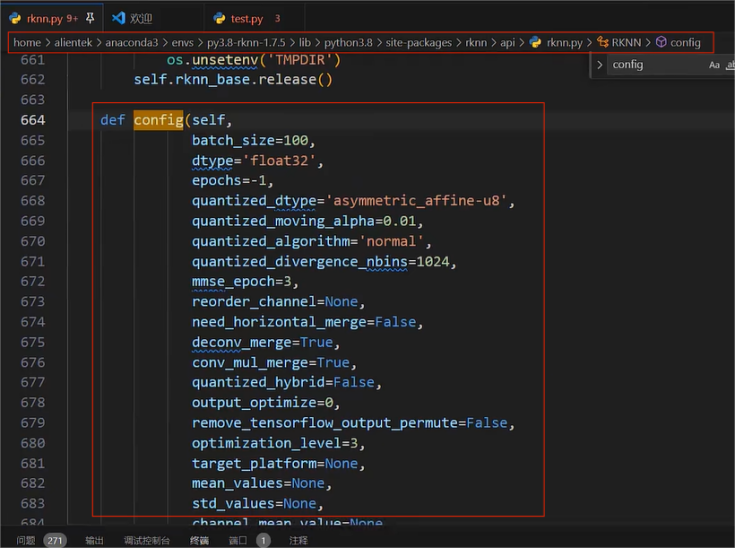
若只是进行模型转换，不需要将RKNN模型发送到板端，则Ubuntu不需要通过ADB来链接开发板.



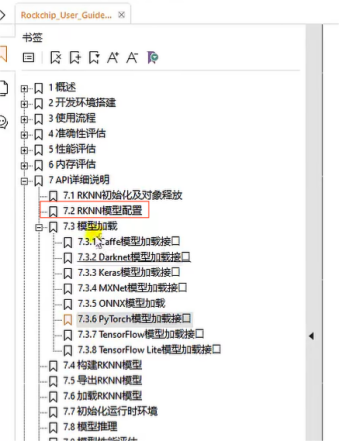
\*/

我们可以查看rknn.config这个api的介绍：我们在vscode中进入目录，然后找到对应的api:

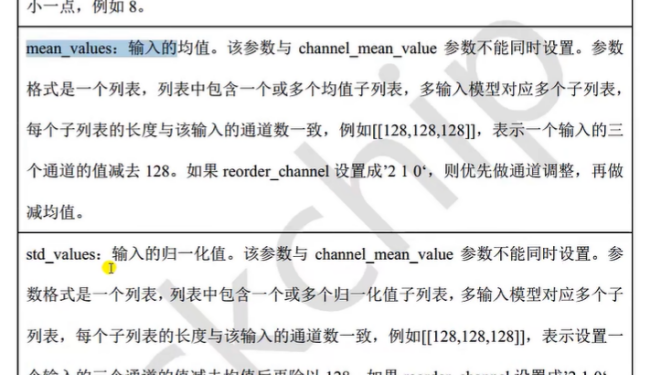




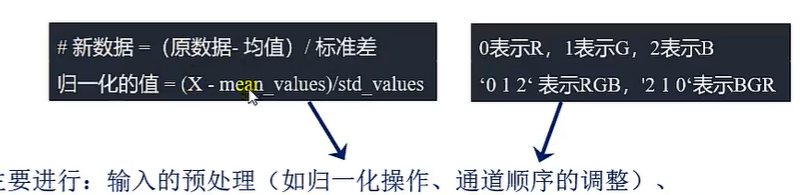
结合RKNN的手册来查看：



例如：



配置了输入图片的均值和标准差之后，输入的Tensor就会被做成如下的处理：

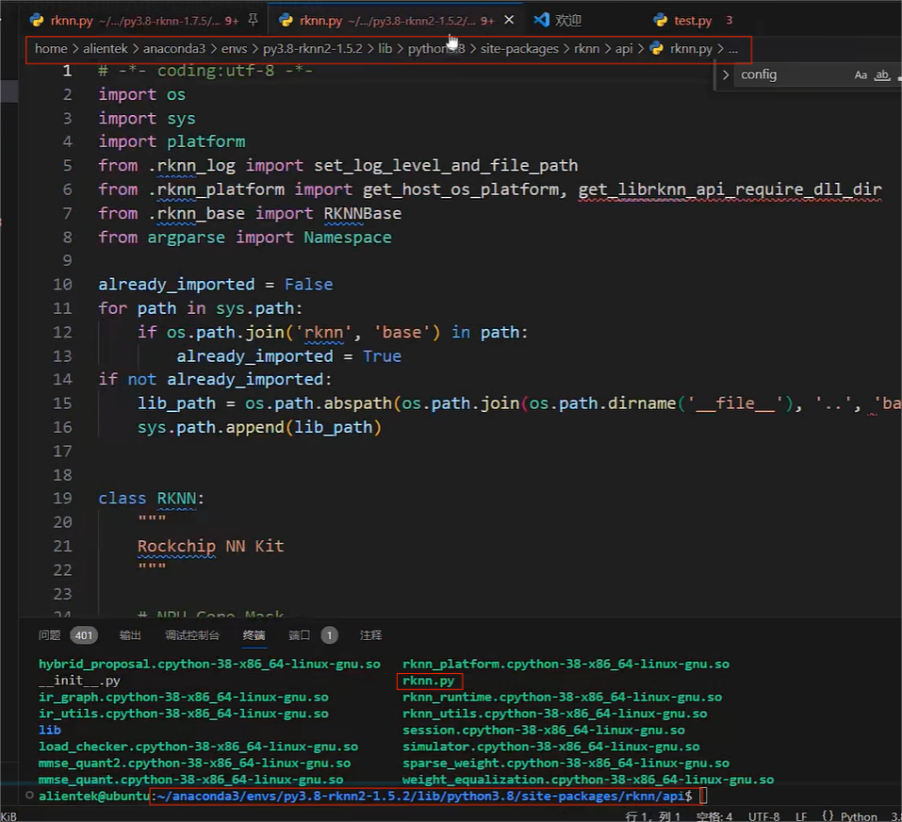


这个新的数据就是最终输入到模型中进行推理的数据。然后reorder\_channel就表示RGB图片的通道顺序，0表示R,1为G，2为B。如果不配置，就会使用默认的None值。就不做通道顺序调整，输入的图片顺序是什么，他就是什么。

此外，还可以进行量化操作，可以在配置模型这里选择量化模型的类型，对于rknn\_toolkit1.7.5版本，它支持的量化类型有3种。

/\*

我们RK3568/88的RKnn\_toolkit使用api见：

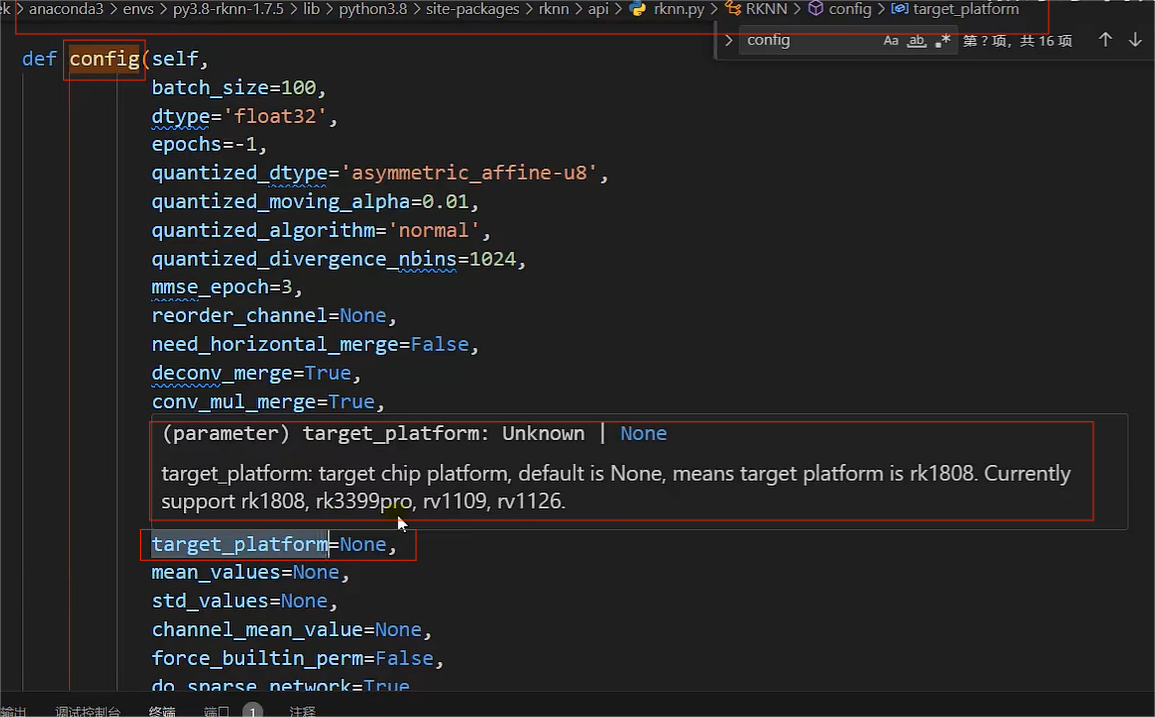


\*/

（rknn\_toolkit 1.7.5版本：）

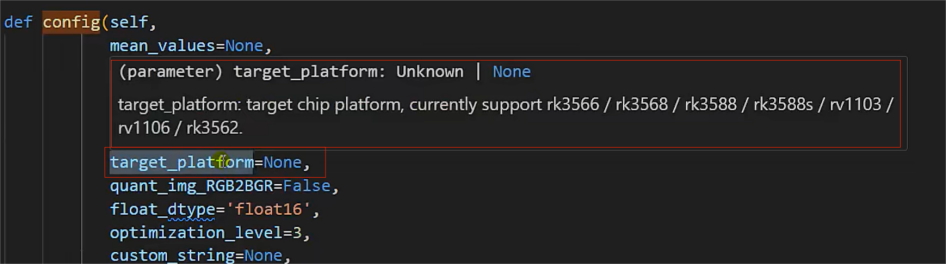
配置rknn模型的目的就是希望后期构建出符合目标运行平台的RKNN模型，所以还需要指定模型的平台。

也就是配置config的target\_platform参数。\*/如果希望得到的RKNN模型后续在RV1126上运行，则这个参数可以配置为RV1126，如果是希望在RK3399pro上运行就配置成rk3399pro:



如果不配置这个参数，对与RKNN Toolkit 1.7.5来说，转换得到的RKNN模型默认是rk1808平台的

如果是rknn\_toolkit2 1.5.2版本，这个参数也是需要配置的：

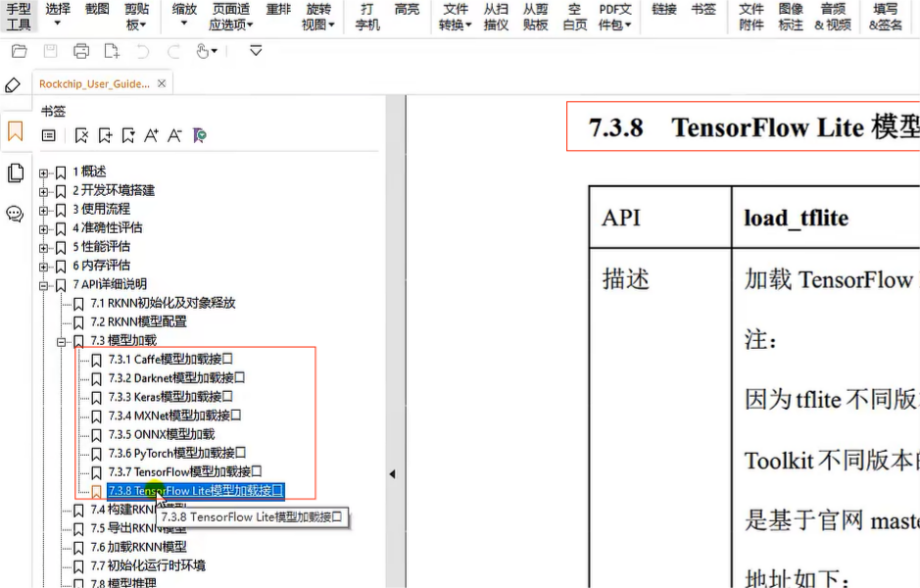


这里只是进行模型的转换而已。没有把RKNN模型发送到开发板端进行推理，所以Ubuntu不需要ADB连接到开发板。

\*/

/\*

配置好要转换的RKNN模型之后，就需要调用对应的api来加载对应的深度学习框架下的模型了，每个深度学习框架模型的加载都有专门的api。（可以参考RKNN手册以及示例代码）：



调用rknn.load\_tensorflow（）、rknn.load\_tflite（）、rknn.load\_pytorch（）、rknn.load\_onnx（）....等等接口，加载对应深度学习框架下的模型

\*/

/\*

构建RKNN模型：根据加载模型的结构、网络的层数、通道数、权重参数等等来进行编译。编译的目的就是要把算法部署到对应的硬件平台上。

构建（或者说编译）就是通过软件程序，如：c/python ,把这些参数编译为2进制代码，那么得到的模型/算法就可以在AI处理器（NPU上运行了）。

这就是构建RKNN模型的目的，构建就是我们常说的编译。

\*/

/\*

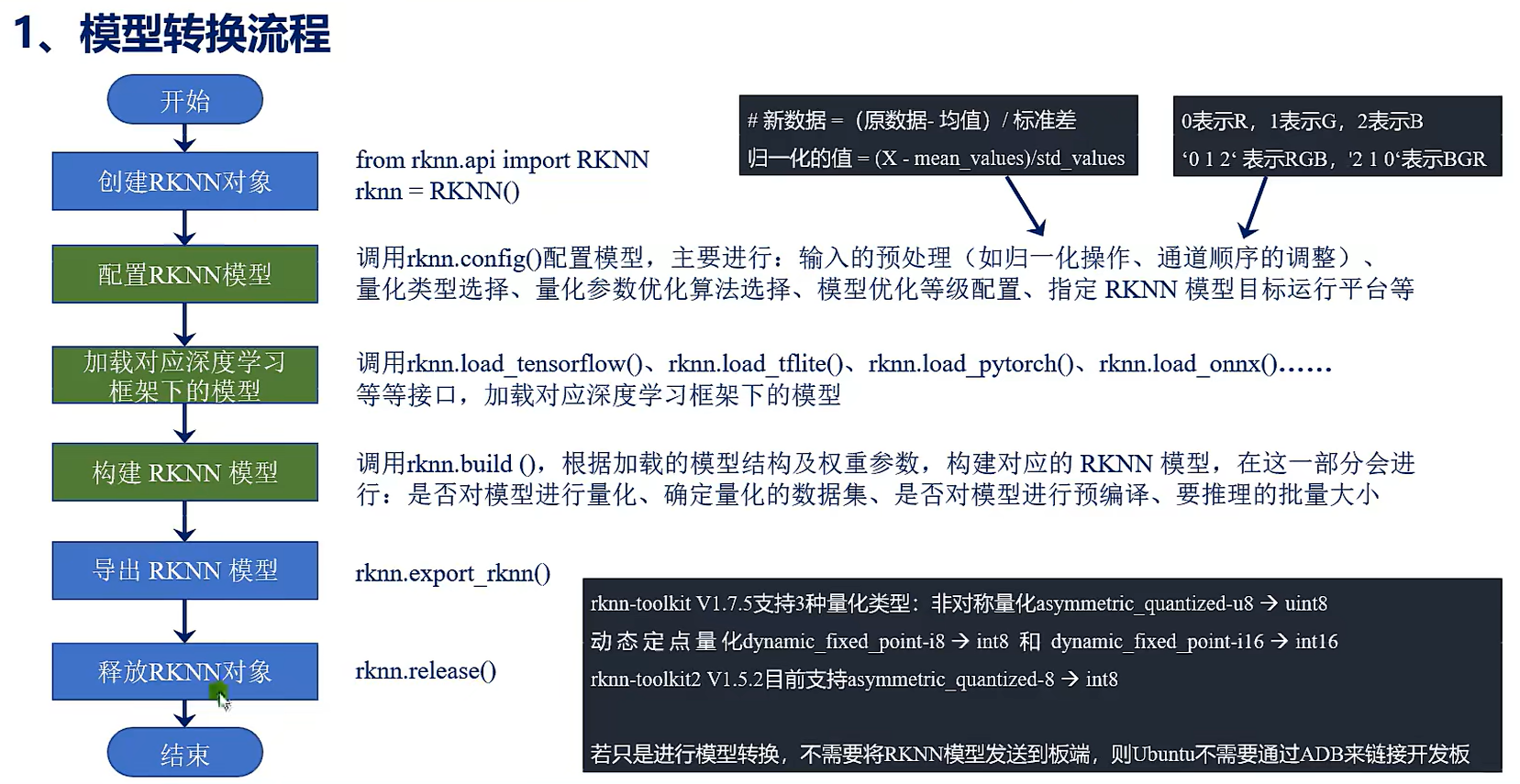
我们构建得到一个RKNN模型后，就可以把RKNN模型导出来使用了，所以直接调用rknn.export\_rknn()这个API来完成模型的导出。

\*/

/\*

导出RKNN模型之后，再调用rknn.release()这个API来释放RKNN对象

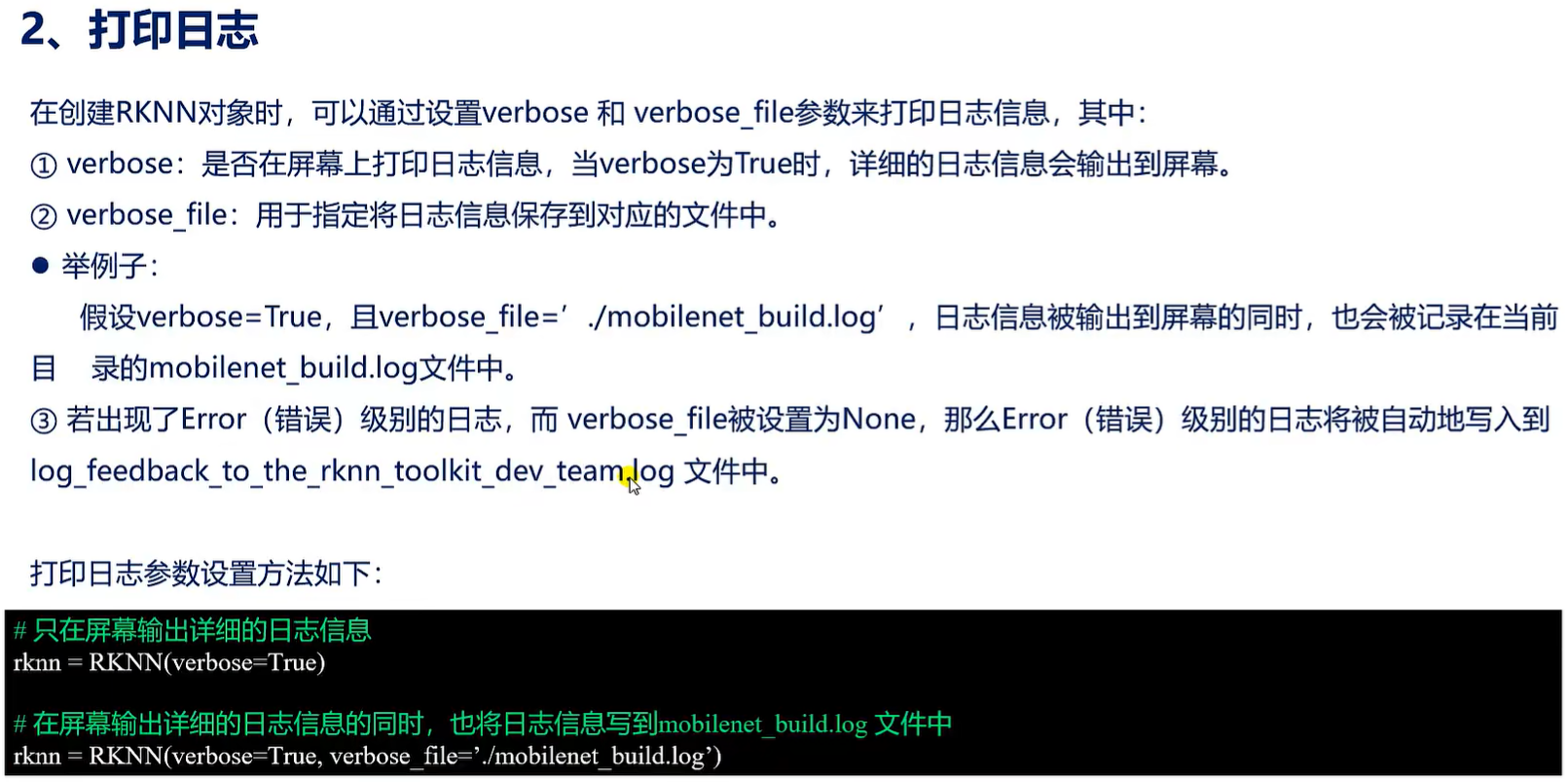
\*/



这就是转换RKNN模型的操作流程。

/\*

在创建RKNN对象的时候：



可以通过设置verbose和verbose\_file参数来打印日志信息。

其中verbose:控制是否在屏幕上打印日志信息。当它为True，就表示将详细的日志信息输出到屏幕上。

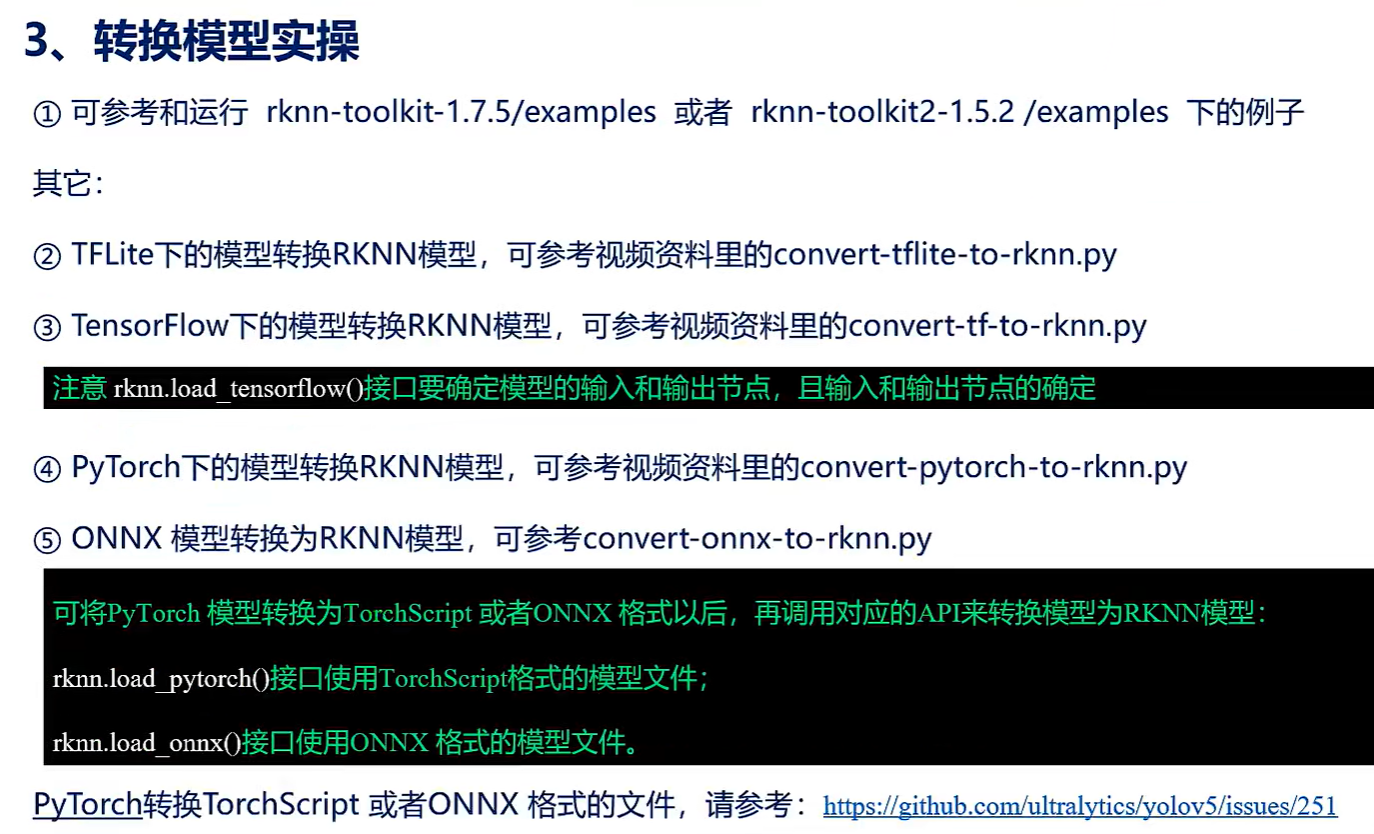
verbose\_file参数使用于将指定日志信息保存到对应的文件中。

如果出现了Error级别日志，且verbose\_file设置为None，则这个错误日志将自动写入一个文件在。这个文件是其自动创建的。

\*/

/\*

下面我们就以个别例子进行操作：



我们将其它深度学习框架下的模型转化为RKNN模型，模型转换可以直接看RKNN Toolkit里的示例代码：这里我们主要参考上面几个文件。



主要以TFLite、TensorFlow、PyTorch、ONNX模型为例子，讲述如何将模型转化为RKNN模型

\*/

\*/

那么下一讲课程，我们就来看怎么将TFLite转化为RKNN模型。