CVITEK TPU常见问题解答

> 文档版本: 1.5.9
>
> 发布日期: 2022-09-29
>
> 适用于 CV183x/CV182x/CV181x系列芯片

法律声明

本数据手册包含北京晶视智能科技有限公司（下称"晶视智能"）的保密信息。未经授权，禁止使用或披露本数据手册中包含的信息。如您未经授权披露全部或部分保密信息，导致晶视智能遭受任何损失或损害，您应对因之产生的损失/损害承担责任。

本文件内信息如有更改，恕不另行通知。晶视智能不对使用或依赖本文件所含信息承担任何责任。

本数据手册和本文件所含的所有信息均按"原样"提供，无任何明示、暗示、法定或其他形式的保证。晶视智能特别声明未做任何适销性、非侵权性和特定用途适用性的默示保证，亦对本数据手册所使用、包含或提供的任何第三方的软件不提供任何保证；用户同意仅向该第三方寻求与此相关的任何保证索赔。此外，晶视智能亦不对任何其根据用户规格或符合特定标准或公开讨论而制作的可交付成果承担责任。

目录

1 模型转换常见问题

1.1模型转换问题

1.1.1 pytorch，tensorflow等是否能直接转换为cvitek框架?

不支持，cvitek仅支持onnx或者caffe框架，可以通过onnx间接支持其他框架模型,因此需要将其他框架模型转化为onnx。

1.1.2 执行model_transform.py报错

model_transform.py脚本作用是将onnx,caffe框架模型转化为fp32 mlir形式，报错很大概率就是存在不支持的算子或者算子属性不兼容，可以反馈给cvitek tpu团队解决。

1.1.3 执行model_deploy.py报错

model_deploy.py作用是先将fp32 mlir通过量化转为int8/bf16mlir形式，然后再将int8/bf16mlir转化为cvimodel。在转化的过程中，会涉及到两次相似度的对比：一次是fp32 mlir与int8/bf16mlir之间的量化对比，一次是int8/bf16mlir与最终转化出来的cvimodel的相似度对比，若相似度对比失败则会出现下列问题。

（1）报错信息：RuntimeError: accuracy validation of quantized model failed

```
Apply dequantization with threhold 17.121788
[output_Transpose                ]      SIMILAR [PASSED]
    (1, 2125, 112) float32
    cosine_similarity      = 0.999203
    correlation_similarity = 0.999203
    euclidean_similarity   = 0.959979
    sqnr_similarity        = 20.596974
300 compared
154 passed
  1 equal, 0 close, 153 similar
146 failed
  0 not equal, 146 not similar
min_similiarity = (0.9182924032211304, 0.9182924032211304, 0.5555645316441598, 6.18746817111969)
Target    nanodetPlus_quantized_tensors_interp.npz
Reference nanodetPlus_full_precision_interp.npz
tolerance_name: tolerance
**********************************************Please Attention**********************************************
tolerance of cosine_similarity(0.99) maybe set too high!
tolerance of euclidean_similarity(0.7) maybe set too high!
##############################################################################
tolerance value you origin set is: --tolerance 0.99 0.99 0.7
We advice to set the tolerance value like this: --tolerance 0.91,0.91,0.55
##############################################################################
npz compare FAILED.
2022/08/26 16:05:30 - ERROR : [!Error]cmd: cvi_npz_tool.py compare nanodetPlus_quantized_tensors_interp.npz nanodetPlus_full_precision_interp.npz --op_info na
nodetPlus_quantized_op_info.csv --tolerance 0.99,0.99,0.7 --tolerance_name tolerance --dequant --stats_int8_tensor --except - -vv
2022/08/26 16:05:30 - ERROR : error occured: 255, func: compare
msg: CompletedProcess(args=['cvi_npz_tool.py', 'compare', 'nanodetPlus_quantized_tensors_interp.npz', 'nanodetPlus_full_precision_interp.npz', '--op_info', 'n
anodetPlus_quantized_op_info.csv', '--tolerance', '0.99,0.99,0.7', '--tolerance_name', 'tolerance', '--dequant', '--stats_int8_tensor', '--except', '-', '-vv'
], returncode=255)
accuracy validation of quantized model failed
```

解决方法：tolerance参数不对。模型转换过程会对int8/bf16 mlir与fp32 mlir的输出计算相似度，而tolerance作用就是限制相似度的最低值，若计算出的相似度的最小值低于对应的预设的tolerance值则程序会停止执行，并给出设置tolerance的值的建议（如上图中的红色框中的信息），可以考虑按照建议进行tolerance的设置。（如果相似度的最小值过低请反馈到cvitek tpu团队解决)

1.1.4 预处理参数如何使用?

预处理过程用公式表达如下（x代表输入）：

$$y = \frac{x \times \frac{raw\_scale}{255.0} - mean}{std} \times input\_scale$$

1.1.5 model_transform.py的channel_order参数和model_deploy.py的pixel_format参数的差异?

channel_order是原始模型的输入图片类型（只支持rgb/bgr planar），pixel_format是转换成cvimodel后的输入图片类型，由客户自行决定（如果输入图片是通过VPSS或者VI获取的YUV图片，可以设置pixel_format为YUV格式）。如果channel_order与pixel_format不一致，cvimodel推理时会自动将输入转成channel_order指定的类型。

1.1.6 是否支持多输入模型，怎么进行预处理?

仅支持多输入图片使用同一种预处理方式的模型，不支持多输入图片使用不同预处理方式的模型。

1.2 量化问题

1.2.1 跑run_calibration.py提示KeyError: 'images'

传入的images的路径不对，请检查数据集的路径是否正确。

1.2.2 跑量化如何处理多输入问题?

多输入模型跑run_calibration.py时，需要使用--image_list参数不能使用--dataset，例如--image_list list1,list2

1.2.3 跑量化输入会进行预处理吗?

会的，根据model_transform.py的预处理参数保存到mlir文件中，量化过程会进行加载预处理参数进行预处理。

1.2.4 跑量化输入程序被系统kill或者显示分配内存失败

需要先检查主机的内存是否足够，常见的模型需要8G内存左右即可。如果内存不够，可尝试在运行run_calibration.py时，添加以下参数来减少内存需求。

```
--tune_thread_num 2              #默认为4，最大为8
--forward_thread_num 2           #默认为4
--buffer_size 2G                 #默认为4G
```

如果内存充足建议使用默认值，或者增大，这样会减少run_calibration.py运行时间。

1.2.5 是否支持手动修改calibration table?

支持，但是不建议修改。

1.3 模型转换常见问题

1.3.1 转换后的模型是否支持加密?

暂时不支持。

1.3.2 bf16的模型与int8模型的速度差异是多少?

大约是3-4倍时间差异，具体的数据需要通过实验验证。

1.3.3 是否支持动态shape?

不支持。如果是固定的几种shape可以依据输入的batch_size以及不同的h和w分别生成独立的cvimodel文件,通过共享权重的形式合并为一个cvimodel。
详见：cvitek_tpu_quick_start_guide.md（9 合并cvimodel模型文件）

2 模型评估常见问题

2.1 模型的评估流程?

先转化为bf16模型，通过cvimodel_tool -a dump -i xxxx.cvimodel命令来评估模型所需要的ION内存以及所占的存储空间，接着在板子上执行model_runner来评估模型运行的时间，之后根据提供的cvitek_sample来评估业务场景下模型精度效果。模型输出的效果准确性符合预期之后，再转化为int8模型再完成与bf16模型相同的流程。

2.2 量化后精度与原来模型对不上，如何调试?

1.转模型脚本（model_deploy.py）如果加了--image选项，会打印模型的相似度，确保相似度较高。
2.比较bf16模型与原始模型的运行结果，确保误差不大。如果误差较大，先确认预处理和后处理是否正确。
3.如果int8模型精度差，可以增加run_calibration.py使用的业务场景数据集（一般为100-1000张图片）。
4.需要注意如果转模型没有指定--fuse_preprocess，那么一般情况bf16模型input为fp32类型，int8模型input为int8，前处理上会有一些差异。input类型可以通过cvimodel_tool -a dump -i xxx.cvimodel查看

2.3 bf16模型的速度比较慢，int8模型精度不符合预期怎么办?

使用混精度量化方法，可参考cvitek快速入门手册混精度量化章节。

2.4 板子的DDR频宽是多少?

主要看ddr位宽设计，详细看下表。

板子型号	DDR带宽
182x	1866*16bit/8 = 3732MB/s
183x	1866*16bit/8 = 3732MB/s or 1866*32bit/8 = 7464MB/s

3 模型部署常见问题

3.1 CVI_NN_Forward接口调用多次后出错或者卡住时间过长?

可能驱动或者硬件问题，需要反馈给cvitek tpu团队解决。

3.2 模型预处理速度比较慢?

1.转模型的时候可以在运行model_deploy.py时加上fuse_preprocess参数，将预处理放到TPU内部来处理。
2.如果图片是从vpss或者vi获取，那么可以在转模型时使用fuse_preprocess、aligned_input，然后使用CVI_NN_SetTensoPhysicalAddr等接口直接将input tensor地址设置为图片的物理地址，减少数据拷贝耗时。

3.3 docker的推理和evb推理的浮点和定点结果是否一样?

定点无差异，浮点有差异，但是相似度比较高，误差可以忽略。

3.4 如果要跑多个模型支持多线程并行吗?

支持多线程，但是多个模型在TPU上推理时是串行进行的。

3.5 填充input tensor相关接口区别

API	是否有内存拷贝	
CVI_NN_SetTensorPtr	是	设置input tensor的虚拟地址，原本的tensor内存不会释放。推理时从用户设置的虚拟地址拷贝数据到原本的tensor内存上。
CVI_NN_SetTensorPhysicalAddr	否	设置input tensor的物理地址，原本的tensor内存会释放。推理时直接从新设置的物理地址读取数据。从VPSS获取的Frame可以调用这个接口，传入Frame的首地址。注意需要转模型的时候model_deploy.py设置--fused_preprocess=1 --aligned_input=1才能调用此接口。
CVI_NN_SetTensorWithVideoFrame	-	通过VideoFrame结构体来填充Input Tensor。注意VideoFrame的地址为物理地址。如果转模型设置--fuse_preprocess=1 --aligned_input=1，则等同于CVI_NN_SetTensorPhysicalAddr，否则会将VideoFrame的数据拷贝到Input Tensor。
CVI_NN_SetTensorWithAlignedFrames	-	支持多batch，与CVI_NN_SetTensorWithVideoFrame类似。
CVI_NN_FeedTensorWithFrames	-	与CVI_NN_SetTensorWithVideoFrame类似

3.6 模型载入后ion内存分配问题

1.调用CVI_NN_RegisterModel后会为weight和cmdbuf分配ion内存（从cvimodel_tool可以看到weight和cmdbuf大小）
2.调用CVI_NN_GetInputOutputTensors后会为tensor（包括private_gmem，shared_gmem，io_mem）分配ion内存
3.CVI_NN_CloneModel可以共享weight和cmdbuf内存
4.其他接口均不会再申请ion内存，即除了初始化，其他阶段模型都不会再申请内存。
5.不同模型的shared_gmem是可以共享（包括多线程情况），因此优先初始化shared_gmem最大的模型可以节省ion内存。

3.7 加载业务程序后模型推理时间变长

设置环境变量export TPU_ENABLE_PMU=1后，模型推理时会打印tpu日志，记录tdma_exe_ms、tiu_exe_ms、inference_ms这3个耗时。
一般加载业务后tdma_exe_ms会变长，tiu_exe_ms不变，这是因为tdma_exe_ms是内存搬运数据耗时，如果内存带宽不够用了，tdma耗时就会增加。
优化的方向：
1.vpss/venc等优化chn，降低分辨率
2.业务层减少内存拷贝，如图片尽量保存引用，减少拷贝等
3.模型填充Input tensor时，使用无拷贝的方式

4 其他常见问题

4.1 在cv182x/cv181x板端环境中出现: taz:invalid option --z解压失败的情况

先在其他linux环境下解压，再放到板子中使用，因为window不支持软链接，所以在windows环境下解压可能导致软链接失效导致报错

4.2 若tensorflow模型为saved_model的pb形式，如何进行转化为frozen_model的pb形式

```
import tensorflow as tf
from tensorflow.keras.applications.mobilenet_v2 import MobileNetV2
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.mobilenet_v2 import preprocess_input,
decode_predictions
import numpy as np
import tf2onnx
import onnxruntime as rt

img_path = "./cat.jpg"
# pb model and variables should in model dir
pb_file_path = "your model dir"
img = image.load_img(img_path, target_size=(224, 224))
x = image.img_to_array(img)
x = np.expand_dims(x, axis=0)
# Or set your preprocess here
x = preprocess_input(x)

model = tf.keras.models.load_model(pb_file_path)
preds = model.predict(x)

# different model input shape and name will differently
spec = (tf.TensorSpec((1, 224, 224, 3), tf.float32, name="input"), )
output_path = model.name + ".onnx"

model_proto, _ = tf2onnx.convert.from_keras(model, input_signature=spec,
opset=13, output_path=output_path)
```