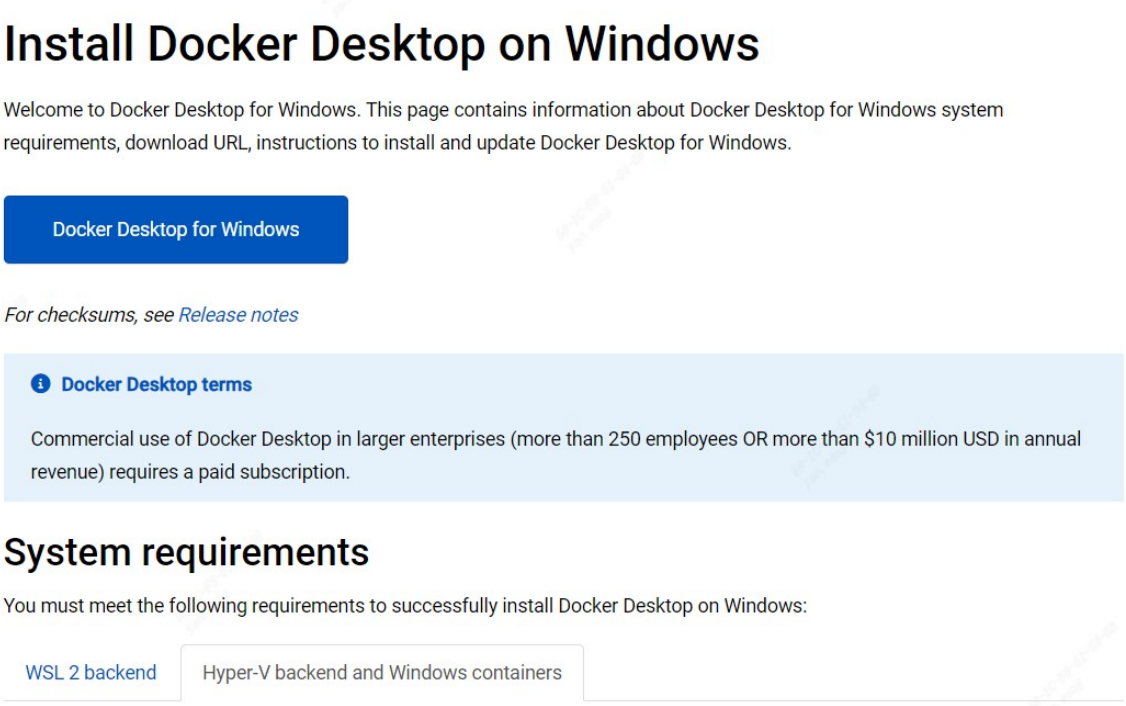
基于duo开发板的Resnet18图像分类

1.配置docker开发环境

docker安装

在windows环境下，可以安装Docker Desktop for Windows，[docker下载地址]

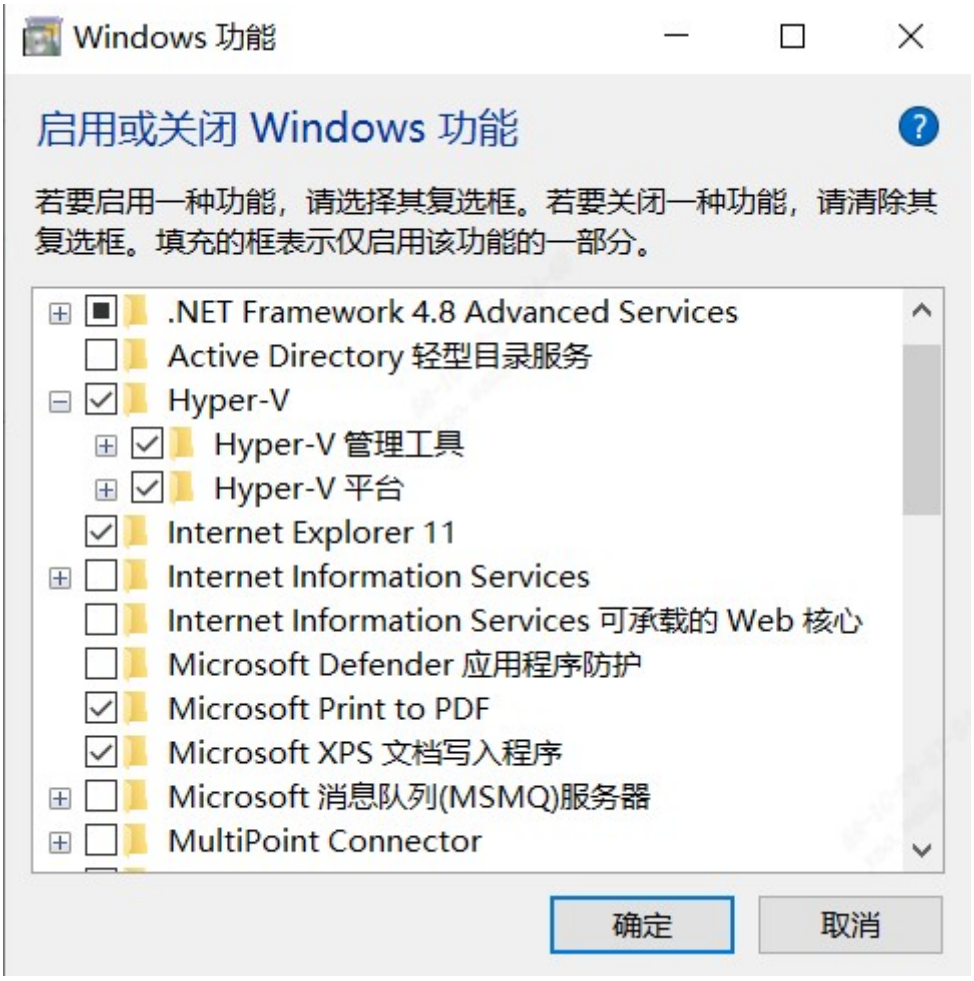


在Windows下运行docker需要相关依赖，即如图中所示，需要使用WSL2后端或者Hyper-V后端作为运行依赖

Hyper-V后端的启用方式如下：

1. 控制面板 —— 程序 —— 启用或关闭Windows功能

2. 找到Hyper-V，勾选Hyper-V管理工具和Hyper-V平台，等待系统文件配置完成后重启电脑



然后即可安装下载好Docker Desktop for Windows，在安装指引中根据选择的后端进行相应的勾选

安装完成后，需要重启电脑，然后即可使用docker

拉取开发所需docker镜像

从docker hub获取镜像文件：

```
docker pull sophgo/tpuc_dev:v2.2
```

启动容器

```
docker run --privileged --name <container_name> -v /workspace -it
sophgo/tpuc_dev:v2.2
```

其中，`<container_name>`为自己定义的容器名

获取开发工具包

可以从下载站台中获取开发工具包`tpu-mlir_xxxx.tar.gz`，`xxxx`为版本号，如`tpu-mlir_v1.2.89-g77a2268f-20230703`：

```
sftp://218.17.249.213
username: cvitek_mlir_2023
password: 7&2Wd%cu5k
```

另外，也可以从github上下载，[下载地址]

拷贝开发工具包

新建一个终端，并将开发工具包从windows拷贝到docker容器中

```
docker cp <path>/tpu-mlir_xxxx.tar.gz <container_name>:/tpu-mlir_xxxx.tar.gz
```

其中，`<path>`为windows系统中开发工具包所在的文件目录，`<container_name>`为容器名

将工具包解压并添加环境变量

在docker容器中，解压工具包并添加环境变量

```
$ tar -zxvf tpu-mlir_xxxx.tar.gz
$ source ./tpu-mlir_xxxx/envsetup.sh
```

2.在docker中准备工作目录

建立`resnet18`工作目录，注意是与`tpu-mlir_xxxx`同级的目录，并将模型文件和图片文件都放入该目录下

```
$ mkdir resnet18 && cd resnet18
$ wget
https://github.com/onnx/models/raw/main/vision/classification/resnet/model/resne
t18-v1-7.tar.gz
$ tar -zxvf resnet18-v1-7.tar.gz
$ cp -rf $TPUC_ROOT/regression/dataset/ILSVRC2012 .
$ cp -rf $TPUC_ROOT/regression/image .
```

这里的`$TPUC_ROOT`是环境变量，对应`tpu-mlir_xxxx`目录

将`resnet18-v1-7.tar.gz`解压后，会在当前目录生成`resnet18-v1-7`文件夹，该文件夹下包含有`resnet18-v1-7.onnx`模型文件

然后在当前目录下新建`work`目录

```
mkdir work && cd work
```

3.ONNX转MLIR

本例中，模型是RGB输入，`mean`和`scale`分别为`123.675,116.28,103.53`和`0.0171,0.0175,0.0174`

将onnx模型转换为mlir模型的命令如下：

```
$ model_transform.py \
  --model_name resnet18 \
  --model_def ../resnet18-v1-7/resnet18-v1-7.onnx \
  --test_input ../image/cat.jpg \
  --input_shapes [[1,3,224,224]] \
  --resize_dims 256,256 \
  --mean 123.675,116.28,103.53 \
  --scale 0.0171,0.0175,0.0174 \
  --pixel_format rgb \
  --test_result resnet18_top_outputs.npz \
  --mlir resnet18.mlir
```

运行成功示例：

```
[resnetv15_pool1_fwd_GlobalAveragePool]        CLOSE [PASSED]
    (1, 512, 1, 1) float32
    close order           = 4
[flatten_170_Flatten            ]        CLOSE [PASSED]
    (1, 512) float32
    close order           = 4
[resnetv15_dense0_fwd_Gemm      ]        CLOSE [PASSED]
    (1, 1000) float32
    close order           = 3
32 compared
32 passed
  0 equal, 5 close, 27 similar
0 failed
  0 not equal, 0 not similar
min_similiarity = (0.9999998807907104, 0.9999992611135766, 121.29961967468262)
Target    resnet18_top_outputs.npz
Reference resnet18_ref_outputs.npz
npz compare PASSED.
compare resnetv15_dense0_fwd_Gemm: 100%|                                        | 32/32 [00:00<00:00, 112.61it/s]
[Success]: npz_tool.py compare resnet18_top_outputs.npz resnet18_ref_outputs.npz --tolerance 0.99,0.99 --except - -vv
```

转成mlir模型后，会生成一个`resnet18.mlir`文件，该文件即为mlir模型文件，还会生成一个`resnet18_in_f32.npz`和一个`resnet18_top_outputs.npz`文件，该文件是后续转模型的输入文件

```
final_opt.onnx  resnet18_in_f32.npz         resnet18_origin.mlir            resnet18_top_outputs.npz
resnet18.mlir   resnet18_opt.onnx.prototxt  resnet18_top_f32_all_weight.npz
```

4.MLIR转INT8模型

生成校准表

在转int8模型之前需要先生成校准表，这里用现有的100张来自ILSVRC2012的图片举例，执行calibration：

```
$ run_calibration.py resnet18.mlir \
  --dataset ../ILSVRC2012 \
  --input_num 100 \
  -o resnet18_cali_table
```

运行成功示例：

```
input_num = 100, ref = 100
real input_num = 100
activation_collect_and_calc_th for op: resnetv15_dense0_fwd_Gemm: 100%|                 | 33/33 [00:05<00:00,  5.78it/s]
[2048] threshold: resnetv15_dense0_fwd_Gemm: 100%|                                     | 33/33 [00:00<00:00, 1043.01it/s]
prepare data from 100
tune op: resnetv15_dense0_fwd_Gemm: 100%|                                           | 33/33 [00:09<00:00,  3.58it/s]
auto tune end, run time:9.400993347167969
```

运行完成后，会生成`resnet18_cali_table`文件，该文件用于后续编译int8模型

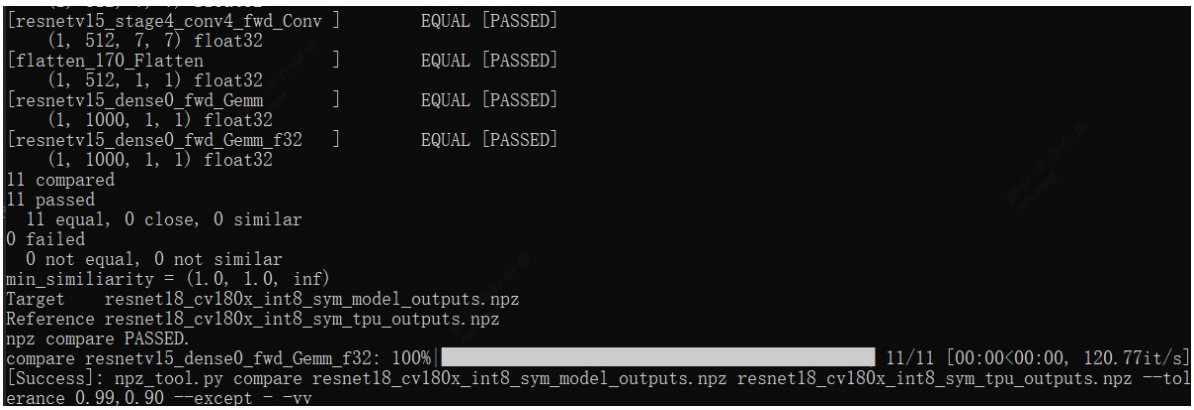
```
final_opt.onnx  resnet18_cali_table  resnet18_opt.onnx.prototxt  resnet18_top_f32_all_weight.npz
resnet18.mlir   resnet18_in_f32.npz  resnet18_origin.mlir        resnet18_top_outputs.npz
```

编译为int8模型

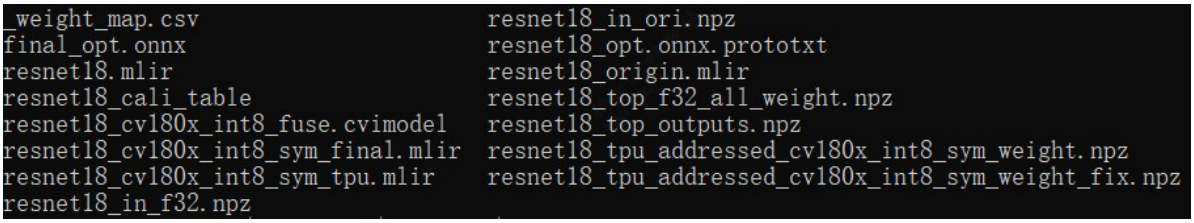
将mlir模型转换为int8模型的命令如下：

```
$ model_deploy.py \
  --mlir resnet18.mlir \
  --quantize INT8 \
  --calibration_table resnet18_cali_table \
  --chip cv180x \
  --test_input ../image/cat.jpg \
  --test_reference resnet18_top_outputs.npz \
  --compare_all \
  --fuse_preprocess \
  --model resnet18_cv180x_int8_fuse.cvimodel
```

编译成功示例：



编译完成后，会生成`resnet18_cv180x_int8_fuse.cvimodel`文件



5.在duo开发板上进行验证

连接duo开发板

根据前面的教程完成duo开发板与电脑的连接，并使用mobaxterm开启终端操作duo开发板

获取cvitek_tpu_sdk

可以从下载站台中获取开发工具包`cvitek_tpu_sdk_cv180x_musl_riscv64_rvv.tar.gz`，注意需要选择`cv180x`的工具包，下载站台如下：

```
sftp://218.17.249.213
username: cvitek_mlir_2023
password: 7&2Wd%cu5k
```

下载完成后，拷贝到docker中并在docker中进行解压

```
$ tar -zxvf cvitek_tpu_sdk_cv180x_musl_riscv64_rvv.tar.gz
```

解压完成后会生成`cvitek_tpu_sdk`文件夹

将开发工具包和模型文件拷贝到开发板上

在duo开发板的终端中，新建文件目录`/home/milkv/`

```
$ mkdir /home/milkv && cd /home/milkv
```

在docker的终端中，将开发工具包和模型文件拷贝到开发板上

```
$ scp -r cvitek_tpu_sdk root@192.168.42.1:/home/milkv
$ scp /workspace/resnet18/work/resnet18_cv180x_int8_fuse.cvimodel
root@192.168.42.1:/home/milkv/cvitek_tpu_sdk
```

设置环境变量

在duo开发板的终端中，进行环境变量的设置

```
$ cd ./cvitek_tpu_sdk
$ source ./envs_tpu_sdk.sh
```

进行图像分类

在duo开发板中，对以下图像进行分类：

在duo开发板的终端中，输入如下命令，使用`resnet18_cv180x_int8_fuse.cvimodel`模型进行图像分类：

```
$ ./samples/bin/cvi_sample_classifier_fused_preprocess \
  ./resnet18_cv180x_int8_fuse.cvimodel \
  ./samples/data/cat.jpg \
  ./samples/data/synset_words.txt
```

运行成功后会输出如下信息：