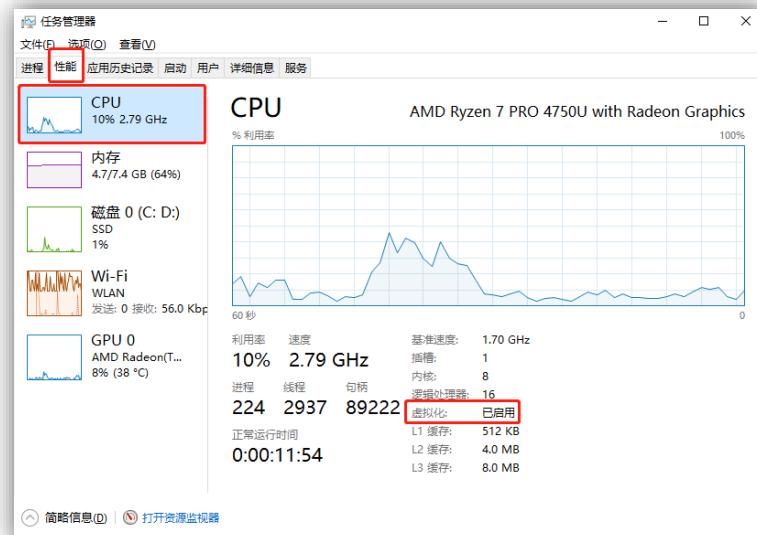

基于 Windows 下 Docker 模型转化教程

目录

- 一、查看 windows 相关配置2
- 二、Docker 下载和安装.....4
- 三、旧版 WSL 的手动安装步骤.....6
- 四、Docker 环境搭建.....7
 - 4.1. 导入镜像.....7
 - 4.2. 查看镜像.....7
 - 4.3. 创建容器.....7
 - 4.4. 退出容器.....7
 - 4.5. 进入容器.....8
 - 4.6. 模型转化.....8

一、查看 windows 相关配置

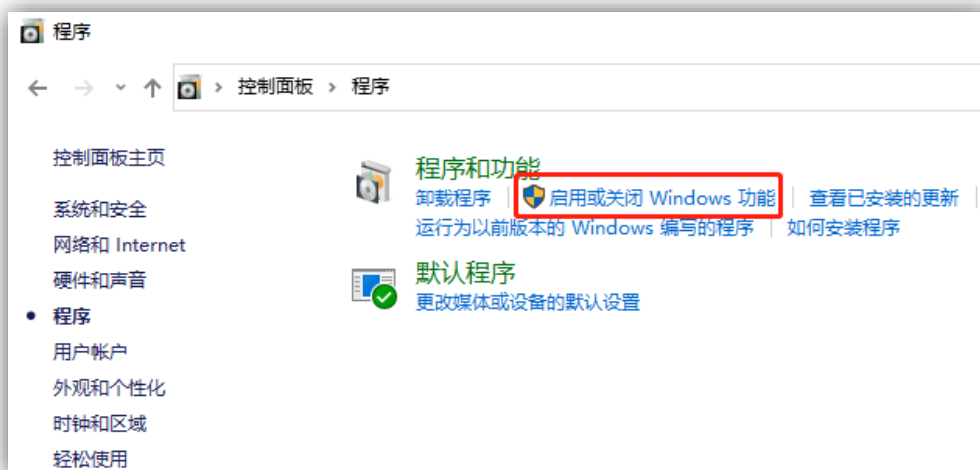
打开任务管理器（Ctrl+Shift+Esc）-> 选择性能 -> CPU，确认 PC 虚拟化功能是否已启用。



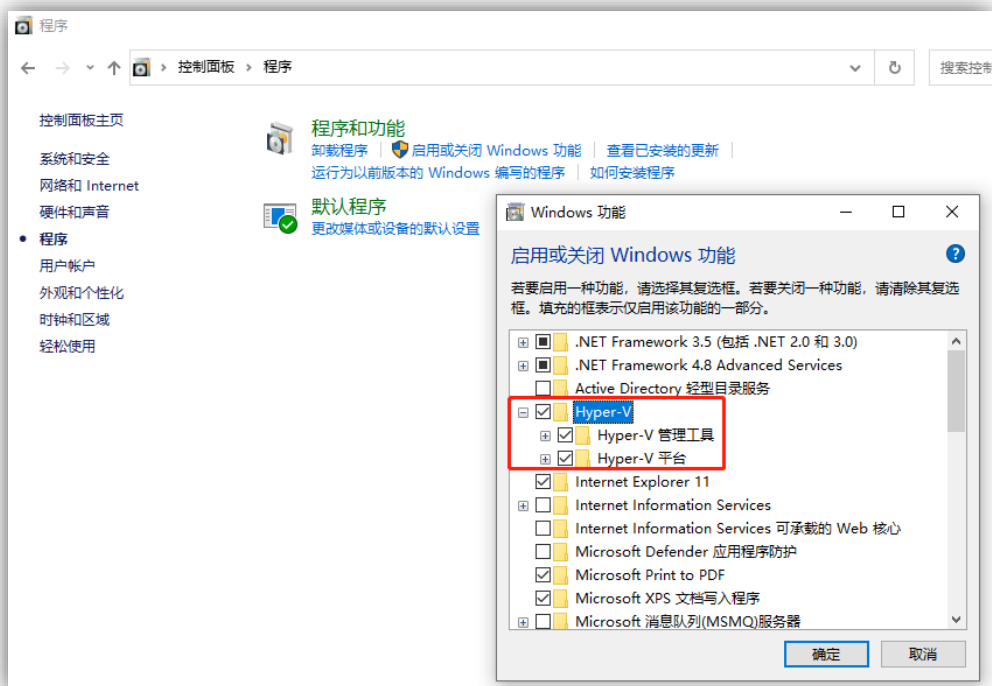
接下来，在控制面板中点击程序。



然后点击启动或关闭 windows 功能。



如图所示，确认 **Hyper-V** 功能正常启用，设置后重启 PC 生效。



如果没找到 **Hyper-v** 选项, 则在桌面新建“**Hyper-V.bat**”文件, 并填写如下内容:

```
pushd "%~dp0"
dir /b %SystemRoot%\servicing\Packages\*Hyper-V*.mum >hyper-v.txt
for /f %i in ('findstr /i . hyper-v.txt 2^>nul') do dism /online /norestart /add-
package:"%SystemRoot%\servicing\Packages\%i"
del hyper-v.txt
Dism /online /enable-feature /featurename:Microsoft-Hyper-V-All /LimitAccess /ALL
```

然后右键单击 **Hyper-V.bat** 文件并且用管理员身份运行。

二、 Docker 下载和安装

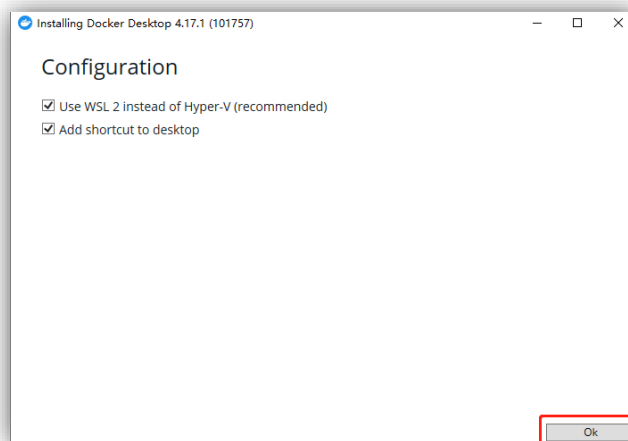
Windows 环境下图形化 docker 软件下载：[docker 下载地址](#)。

安装文件如图所示：

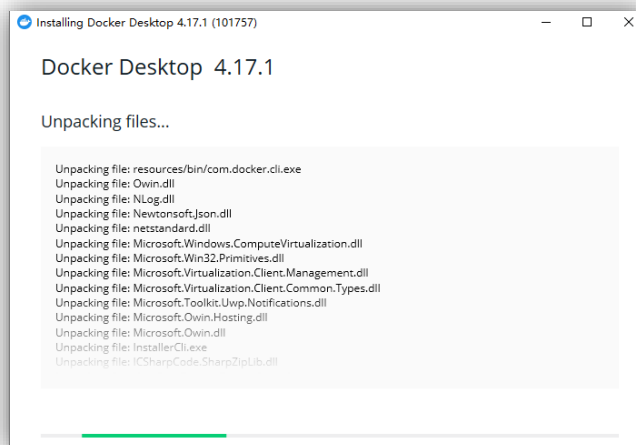


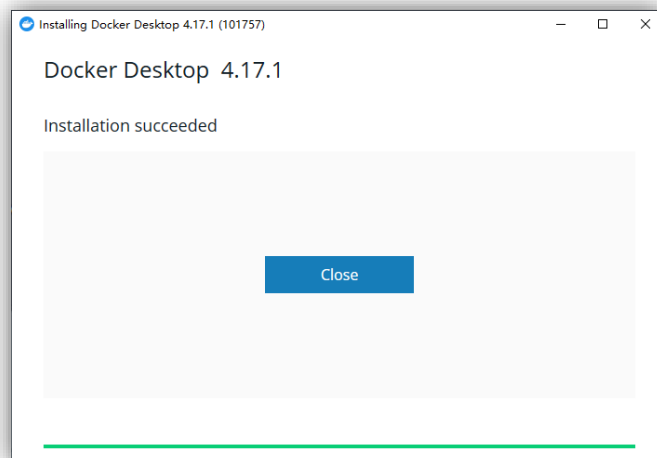
点击 **Docker Desktop installer.exe** 文件进行安装操作

单击 **ok** 即可进行安装：

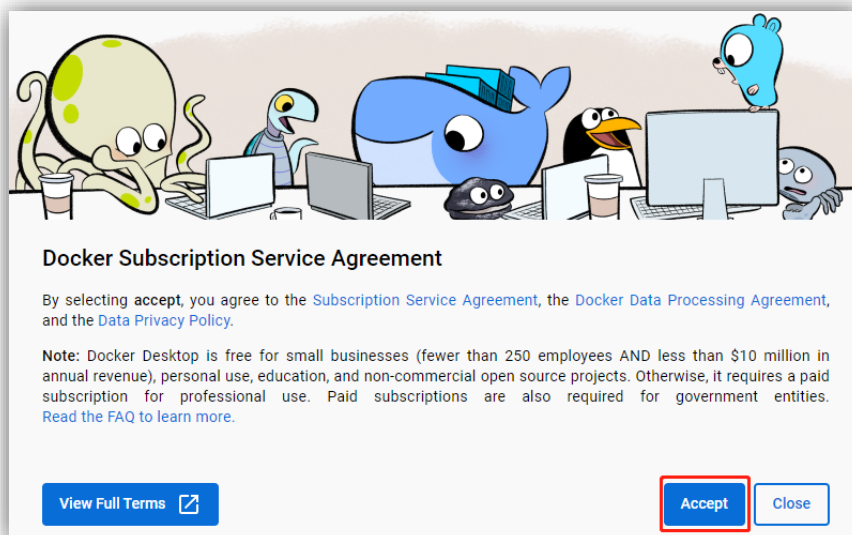


执行完毕后点击 close：





点击 **Accept** 完成安装:



打开 Windows 管理员终端 ([右键开始菜单](#)>Windows PowerShell), 查询 Docker 版本:

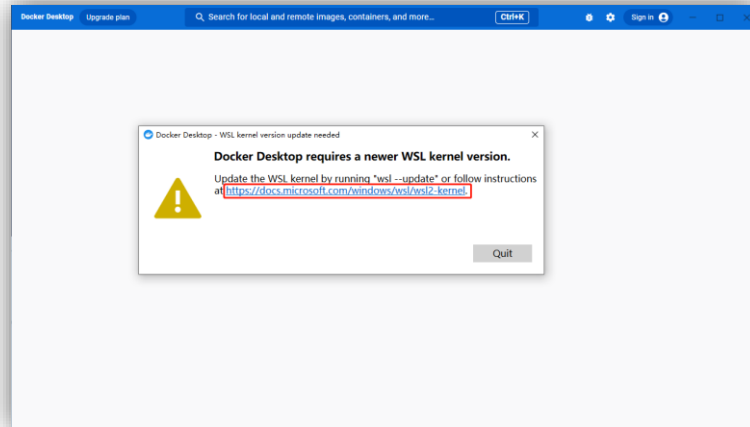
```
docker --version
```



此时 docker 已经安装好但是没有相对应的环境, 所以无法启动桌面上的 docker。

三、 旧版 WSL 的手动安装步骤

如果打开 docker 时遇到如图所示状况。



打开 Windows 管理员终端（右键开始菜单>Windows PowerShell），输入以下命令：

```
dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart
```

安装 WSL2 之前，必须启用“虚拟机平台”可选功能。计算机需要虚拟化功能才能使用此功能。以管理员身份打开 PowerShell 并运行：

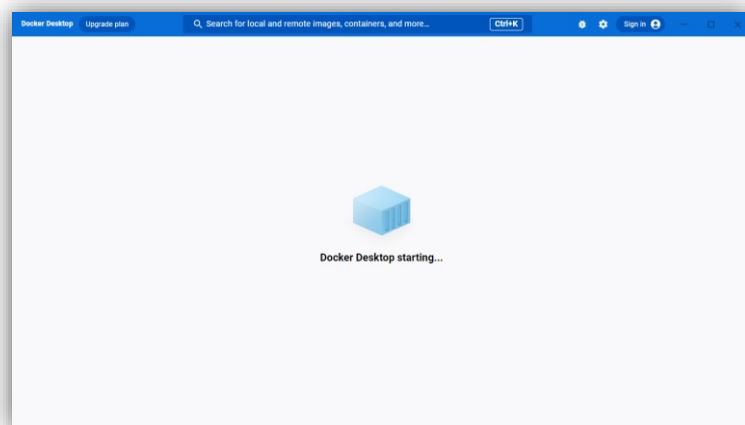
```
dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart
```

下载 [Linux 内核更新包](#) 并且下载的更新包，（双击以运行 - 系统将提示你提供提升的权限，选择“是”以批准此安装。）然后运行：

```
wsl --set-default-version 2
```

将 WSL2 设置为默认版本

操作完成之后即可正常启动 docker



四、 Docker 环境搭建

从相关路径下载“ppnc2.0.tar”镜像文件和“compiler.zip”源码文件。

4.1. 导入镜像

```
docker import ppnc2.0.tar ubuntu20:ppnc2.0
```

“ppnc2.0.tar”要修改为文件所在的真实路径（比如：C:\Users\liaot\Downloads\ppnc2.0.tar）。

```
PS C:\Users\liaot\Downloads> docker import ppnc2.0.tar ubuntu20:ppnc2.0
sha256:e0f684a08c99562068e5b9f370c7c67b721519bc3ae27e0caaf191a3bfe11a98
```

4.2. 查看镜像

输入以下指令查询镜像导入成功：

```
docker images
```

REPOSITORY	TAG	IMAGE ID	CREATED	SI
ppnc2.0				
ubuntu20				
ppnc2.0		e0f684a08c99	8 minutes ago	10

4.3. 创建容器

创建成功后会自动进入该容器

```
docker run -it --name ppnc2.0 -v /home/sasu/work/:/home/edgeboard/workspace
ubuntu20:ppnc2.0 /bin/bash
```

ppnc2.0: 为容器名称

/home/sasu/work/: PC 机上要映射给容器的目录（Windows 目录，后续传递文件）

/home/edgeboard/workspace: 容器中被映射到的目录（Docker 挂载目录）

```
root@85c48213e59b: /
PS C:\Users\liaot> docker run -it --name ppnc2.0 -v C:/Users/liaot/Downloads/work:/home/edgeboard/workspace ubuntu20:ppnc2.0 /bin/bash
root@85c48213e59b: /# |
```

4.4. 退出容器

#在容器中执行

```
Exit
```

```
管理员: Windows PowerShell
PS C:\Users\liaot> docker run -it --name ppnc2.0 -v C:/Users/liaot/Downloads/work:/home/edgeboard/workspace ubuntu20:ppnc2.0 /bin/bash
root@85c48213e59b:/# exit
exit
PS C:\Users\liaot> |
```

4.5. 进入容器

#启动容器

```
docker start ppnc2.0
```

#进入容器

```
docker exec -it ppnc2.0 /bin/bash
```

```
PS C:\Users\liaot\Downloads> docker start ppnc2.0
ppnc2.0
PS C:\Users\liaot\Downloads> docker exec -it ppnc2.0 /bin/bash
root@65ffd9221f61:/# |
```

4.6. 模型转化

#导入环境变量

```
export PPNC_HOME=/usr/local/ppnc
```

#导入相关路径

```
source /usr/local/ppnc/scripts/activate_env.sh
```

#准备工作

在 PC 中将 compiler.zip 压缩包存放至“work”文件夹下并解压，在 compiler 文件夹下创建一个“source”文件夹包含以下两个文件夹(model、 image):

model: 将在 ai studio 上导出的模型放到该目录下

Image: 将测试集的图片放到该目录下，请确保图片数量大于 50 张，并覆盖所有的种类。

#对 compiler 文件夹中的 config.json 文件进行配置

```
{
  "model_dir": "/home/yongzhen/work/compiler/source",
  "shape": "[320, 320]",
  "quantize_num": "50",
  "split": true
}
```

model_dir: 该“source”文件夹为上述<准备工作>定义的目录，在其内存放相关输入数据，该目录下包含 model 和 image 两个文件夹;

shape: 模型输入尺寸，需要与模型实际尺寸保存一致；

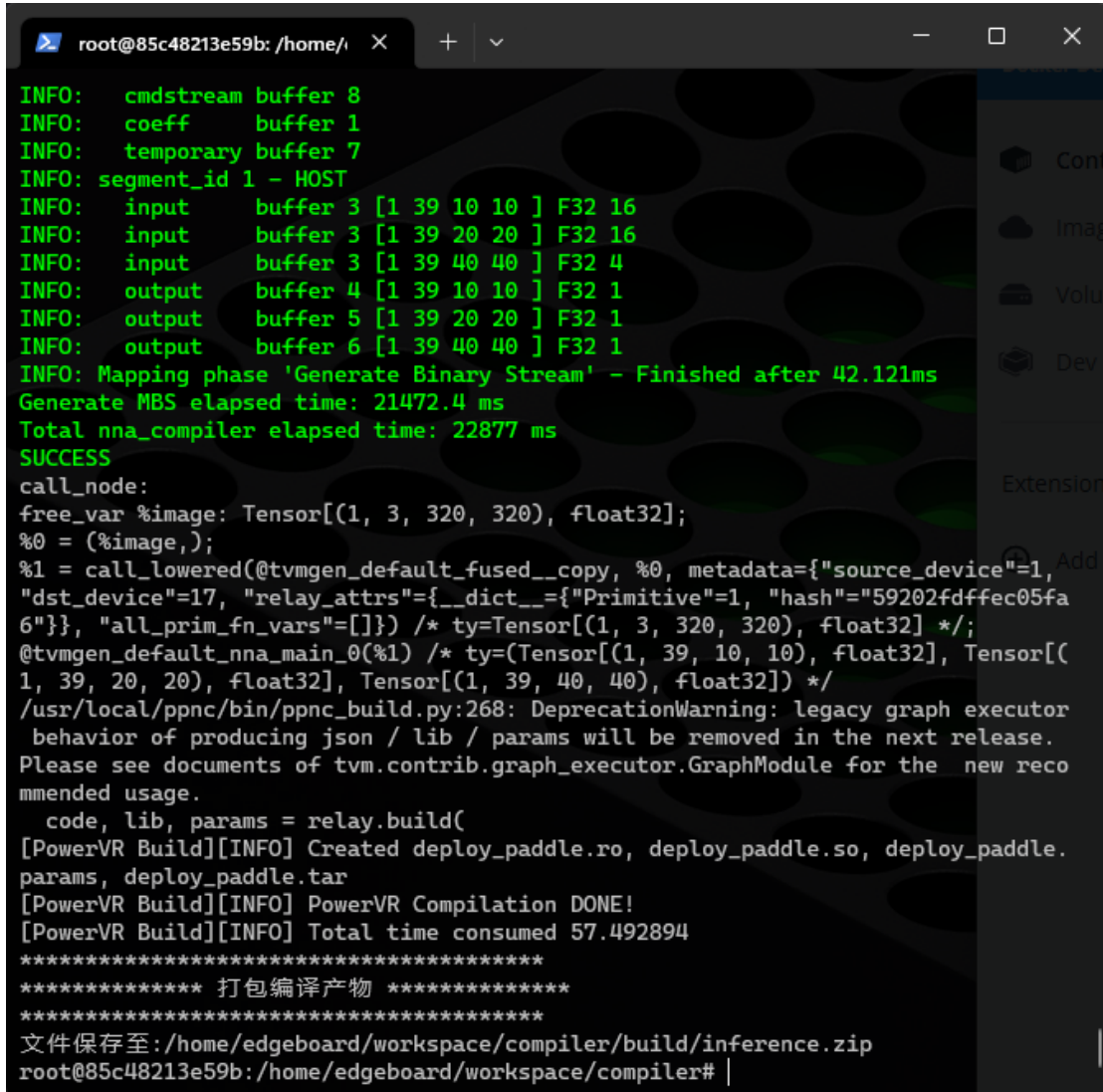
quantize_num: 设置为 50 张

#开始编译

进入到 compiler 文件夹下执行以下指令开始转化 AI 模型：

```
python3 compile.py config.json
```

如图所示模型转化成功结果（预计 3~5 分钟）：



```
INFO: cmdstream buffer 8
INFO: coeff buffer 1
INFO: temporary buffer 7
INFO: segment_id 1 - HOST
INFO: input buffer 3 [1 39 10 10 ] F32 16
INFO: input buffer 3 [1 39 20 20 ] F32 16
INFO: input buffer 3 [1 39 40 40 ] F32 4
INFO: output buffer 4 [1 39 10 10 ] F32 1
INFO: output buffer 5 [1 39 20 20 ] F32 1
INFO: output buffer 6 [1 39 40 40 ] F32 1
INFO: Mapping phase 'Generate Binary Stream' - Finished after 42.121ms
Generate MBS elapsed time: 21472.4 ms
Total nna_compiler elapsed time: 22877 ms
SUCCESS
call_node:
free_var %image: Tensor[(1, 3, 320, 320), float32];
%0 = (%image,);
%1 = call_lowered(@tvmgen_default_fused__copy, %0, metadata={"source_device"=1,
"dst_device"=17, "relay_attrs"={__dict__={"Primitive"=1, "hash"="59202fdffec05fa
6"}}, "all_prim_fn_vars"=[]}) /* ty=Tensor[(1, 3, 320, 320), float32] */;
@tvmgen_default_nna_main_0(%1) /* ty=(Tensor[(1, 39, 10, 10), float32], Tensor[(
1, 39, 20, 20), float32], Tensor[(1, 39, 40, 40), float32]) */
/usr/local/ppnc/bin/ppnc_build.py:268: DeprecationWarning: legacy graph executor
behavior of producing json / lib / params will be removed in the next release.
Please see documents of tvm.contrib.graph_executor.GraphModule for the new reco
mmended usage.
code, lib, params = relay.build(
[PowerVR Build][INFO] Created deploy_paddle.ro, deploy_paddle.so, deploy_paddle.
params, deploy_paddle.tar
[PowerVR Build][INFO] PowerVR Compilation DONE!
[PowerVR Build][INFO] Total time consumed 57.492894
*****
***** 打包编译产物 *****
*****
文件保存至:/home/edgeboard/workspace/compiler/build/inference.zip
root@85c48213e59b:/home/edgeboard/workspace/compiler#
```

#编译产物

会在 build 目录中生成 inference.zip 压缩包，将编译后的模型包拷贝并解压到板卡上的推理工程模型目录即可。