RKNN User Guide

RKNN 변환 매뉴얼

RKNN이란 Rock칩에서 동작하기 위해 필요한 딥러닝 프레임워크 중 하나입니다. 본 매뉴얼은 RKNN Tool-kit 설치부터, RKNN모델 변환, 전&후처리에 관한 내용을 포함하고 있습니다.

목차

[개발 환경 1](#_Toc40796021)

[RKNN-Toolkit 설치 방법 4](#_Toc40796022)

[pb를 freeze하는 방법 10](#_Toc40796023)

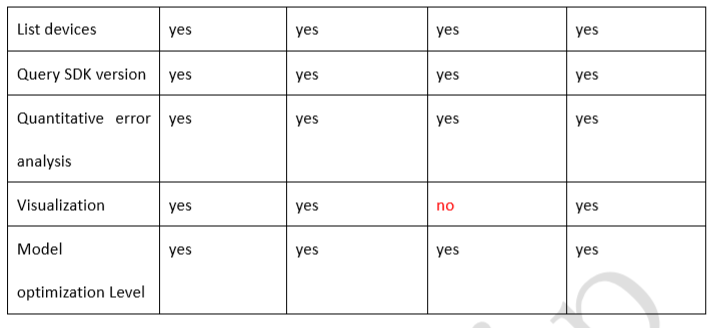
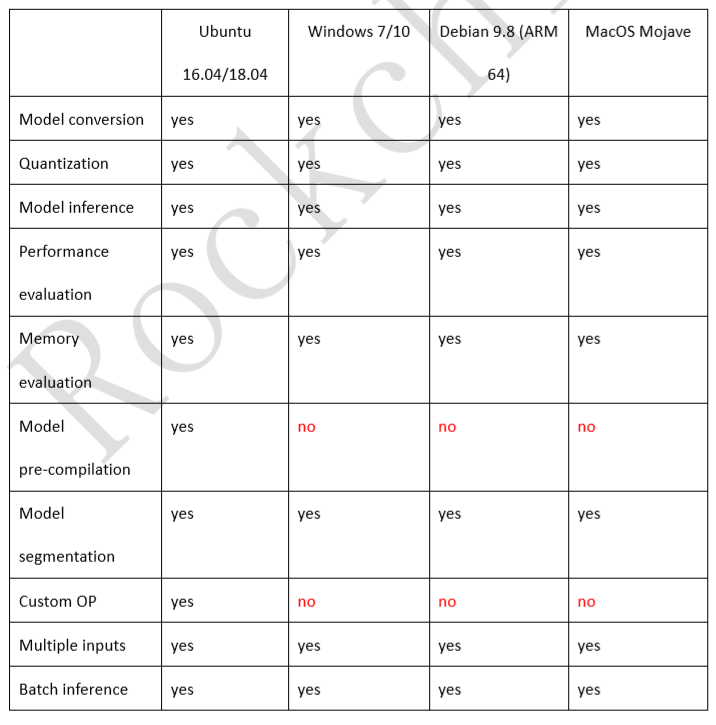
[프로젝트 시작하기 11](#_Toc40796024)

[전처리, 후처리 과정 12](#_Toc40796025)

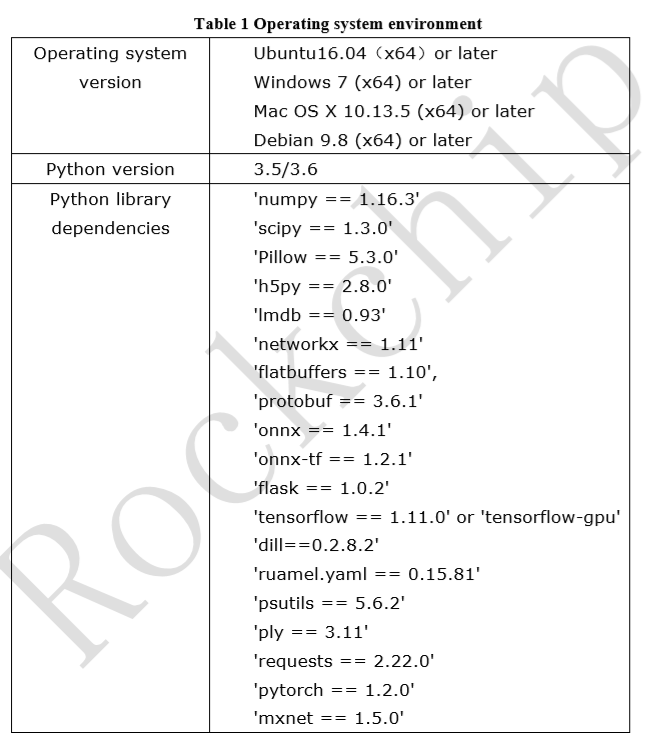
[Trouble Shooting 12](#_Toc40796026)

# 개발 환경

운영체제에 따른 지원 항목입니다. 운영체제에 따라 지원하지 않는 RKNN tool이있으므로, Ubuntu 사용을 권장드립니다.



다음은 운영체제 및 Python버전, Python 라이브러리 버전에 대한 표입니다.



[참고 사항]

1. Window는 현재 Python 3.6만 지원합니다.

2. MacOS는 Python 3.6, 3.7을 지원합니다.

3. Arm64는 Python 3.5, 3.7을 지원합니다.

4. Mac OS의 Scipy 버전은 1.3.0만 가능합니다.

# RKNN-Toolkit 설치 방법

Rknn-Toolkit은 RKNN 변환 및 실행을 가능하게 만들어 줍니다.

설치 방법에는 (1) pip install command (2) running docker image 두가지의 방법이 있으나, 본 메뉴얼에는 pip install에 대해서만 언급합니다.

**필자는 Ubuntu 18.04, python 3.5를 사용했습니다.**

**1) 가상 환경 설정**

: 다양한 프로젝트를 진행하게 되면, 각 프로젝트마다 새로운 개발 환경이 필요합니다. 가상 환경 설정을 하게 되면 프로젝트마다 개발 환경을 구축할 수 있습니다.

리눅스 터미널 혹은 서버에서 진행하면 됩니다.

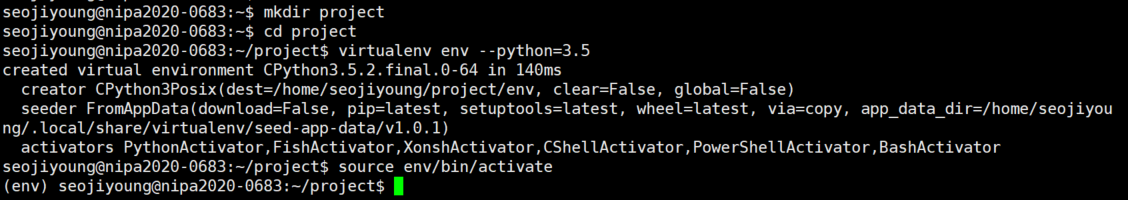
$sudo apt install virtualenv

$mkdir project

$cd project

$virtualenv env --python=3.5

$source env/bin/activate



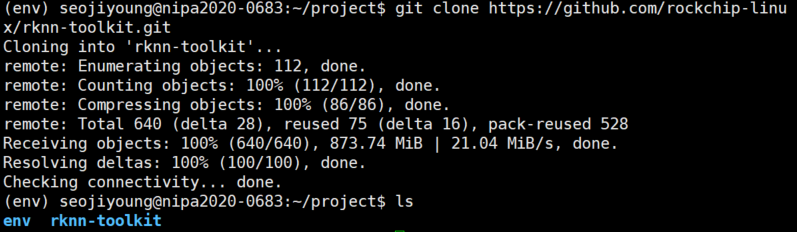
(env) 가상환경이 실행 중입니다.

\*참고) 가상환경을 종료하고 싶다면

deactivate

**2) RKNN-Toolkit 파일 git clone**

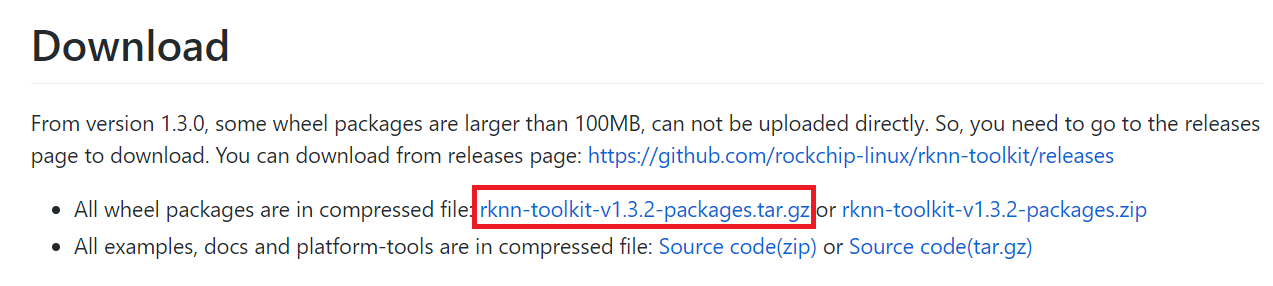
git clone https://github.com/rockchip-linux/rknn-toolkit.git



**3) RKNN-Toolkit 파일 설치**

<https://github.com/rockchip-linux/rknn-toolkit>

위의 링크에 접속하시어, Download의 rknn-toolkit-v1.3.2-packages.tar.gz 파일을 다운받아 주세요. 압축을 풀어, 각 파이썬과 운영체제에 맞는 버전을 설치하시면 됩니다.



ex) Python=3.5, Ubuntu 18.04의 경우에는 아래 파일을 설치합니다.

$pip3 install rknn\_toolkit-1.3.2-cp36-cp36m-linux\_x86\_64.whl

> 제대로 설치 되었는지 확인하는 코드

$python3

>>>**from** **rknn.api** **import** RKNN *#오류가 나지 않으면 성공!*

>>>exit() *#파이썬 나가는 함수*

**4) 필요한 라이브러리 설치**

/rknn-toolkit/packages 로 들어간 후 실행해야 합니다.

$pip3 install -r requirements-gpu.txt

gpu가 없다면, requirements-cpu.txt 파일을 설치해주세요.

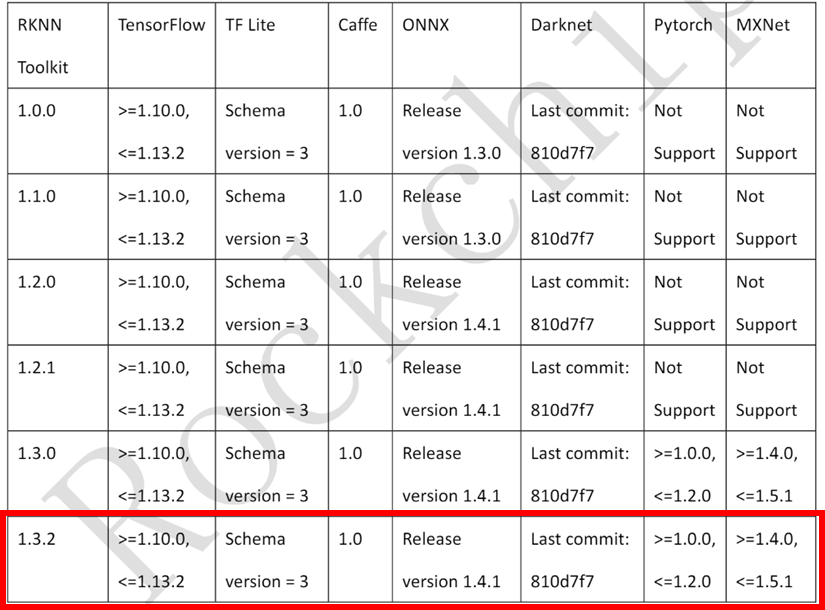
**프로젝트를 진행하면서 필요한 라이브러리가 있다면, 추가로 설치해주세요.**

RKNN 모델 변환 방법

REX 보드에 올리기 전에, PC에서 RKNN api를 이용하여 결과가 잘 나오는지 확인해야 합니다. 이후에는 C언어를 사용하여 입출력을 정의하는 등의 과정이 추가로 필요합니다.

**\* 현재 출시된 1.3.2버전에서 지원하는 딥러닝 프레임워크**

지원하는 모델: Caffe, TensorFlow, TensorFlw Lite, ONNX, Darknet, Pytorch, MXNet model



빨간 박스를 참고해주세요. 다음과 같은 버전으로 만든 모델을 rknn으로 변환해야 합니다.

예를 들어, tensorflow는 1.10.0~1.13.2버전의 함수 기준으로 rknn으로 convert 됩니다.

약간의 유동성은 존재하니, 참고만 해주세요.

**1) Python 파일 편집**

vi convert.py

\* 이제 cmd창이 아닌, python파일에서 코드를 작성해주세요.

**2) rknn object 생성**

**from** **rknn.api** **import** RKNN

*#create RKNN object*

rknn = RKNN(verbose=**True**)

rknn 객체를 만들어 주었습니다.

(선택 사항) verbose=True를 써주면 변환 과정에서의 오류를 볼 수 있습니다.

**3) Non-RKNN model 로드**

각 모델의 load 방식이 조금씩 다르므로, Ctrl+F를 이용하여 찾으시길 바랍니다.

Rock Chip에서 tensorflow 함수를 가장 많이 지원하기 때문에, 먼저 tensorflow 모델을 변환해보는 걸 추천해드립니다.

**> Tensorflow 모델**

*#Load model*

print('--> Loading model') *#로딩하는 중임을 알려주는 출력*

*#freeze된 pb파일, input 노드, output노드, input 사이즈 리스트 입력*

*#노드 확인은 tensorboard, netron을 활용*

*#아래 예시는 image fusion을 참고했습니다.*

ret = rknn.load\_tensorflow(tf\_pb='./deepfusion.pb',

inputs=['content','style'],

outputs=['BiasAdd\_6'],

input\_size\_list=[[ INPUT\_SIZE\_HEIGHT,INPUT\_SIZE\_WIDTH, 1],

[INPUT\_SIZE\_HEIGHT, INPUT\_SIZE\_WIDTH, 1]])

*#오류 발생시 오류 메시지 출력*

**if** ret !=0:

print('Load failed!')

exit(ret)

*#Load 완료시 'done'출력*

print('done')

**\* Tensorflow 모델을 변환할 때, 일반 pb파일이 아닌 freeze된 pb파일을 넣어주어야 합니다. 페이지 10쪽을 참고해주세요.**

**> Tensorflow Lite 모델**

*#Load model*

print('--> Loading model') *#로딩하는 중임을 알려주는 출력*

ret = rknn.load\_tflite(model='./mobilenet\_v1.tflite')

*#오류 발생시 오류 메시지 출력*

**if** ret !=0:

print('Load failed!')

exit(ret)

*#Load 완료시 'done'출력*

print('done')

**> Caffe 모델**

*#Load model*

print('--> Loading model') *#로딩하는 중임을 알려주는 출력*

ret = rknn.load\_caffe(model='./mobilenet\_v2.prototxt', proto='caffe',

blobs='./mobilenet\_v2.caffemodel')

*#오류 발생시 오류 메시지 출력*

**if** ret !=0:

print('Load failed!')

exit(ret)

*#Load 완료시 'done'출력*

print('done')

**> ONNX 모델**

*#Load model*

print('--> Loading model') *#로딩하는 중임을 알려주는 출력*

ret = rknn.load\_onnx(model='./arcface.onnx')

*#오류 발생시 오류 메시지 출력*

**if** ret !=0:

print('Load failed!')

exit(ret)

*#Load 완료시 'done'출력*

print('done')

**> Darknet 모델**

*#Load model*

print('--> Loading model') *#로딩하는 중임을 알려주는 출력*

ret = rknn.load\_darknet(model='./yolov3-tiny.cfg', weight='./yolov3.weights')

*#오류 발생시 오류 메시지 출력*

**if** ret !=0:

print('Load failed!')

exit(ret)

*#Load 완료시 'done'출력*

print('done')

**> Pytorch 모델**

*#Load model*

print('--> Loading model') *#로딩하는 중임을 알려주는 출력*

ret = rknn.load\_pytorch(model='./resnet18.pt', input\_size\_list=[[3, 224, 224]])

*#오류 발생시 오류 메시지 출력*

**if** ret !=0:

print('Load failed!')

exit(ret)

*#Load 완료시 'done'출력*

print('done')

**> MXNet 모델**

*#Load model*

print('--> Loading model') *#로딩하는 중임을 알려주는 출력*

ret = rknn.load\_mxnet(symbol='resnext50\_32x4d-symbol.json',

params='resnext50\_32x4d-4ecf62e2.params',

input\_size\_list=[[3,224,224]]

*#오류 발생시 오류 메시지 출력*

**if** ret !=0:

print('Load failed!')

exit(ret)

*#Load 완료시 'done'출력*

print('done')

**4) Non-RKNN model 빌드**

*#Build model*

print('--> Building model') *#빌드하는 중임을 알려주는 출력*

ret = rknn.build(do\_quantization=**False**)

*#오류 발생시 오류 메시지 출력*

**if** ret !=0:

print('Build failed!')

exit(ret)

*#Build 완료시 'done'출력*

print('done')

quantization으로 dataset 업로드가 가능하나, 굳이 필요하지 않아서 False로 해도 무방합니다.

**5) Non-RKNN model 내보내기**

*#Export rknn model*

print('--> Export RKNN model')

ret = rknn.export\_rknn('./mobilenet\_v1.rknn') *#다음과 같은 이름으로 저장됩니다.*

*#오류 발생시 오류 메시지 출력*

**if** ret != 0:

print('Export failed!')

exit(ret)

*#export 완료시 'done'출력*

print('done')

# pb를 freeze하는 방법

**1) 새로운 directory를 만들어 작업을 해줍니다.**

$mkdir freeze

**2) freeze\_graph를 사용하기 위해 tensorflow tool을 git clone 합니다.**

$git clone https://github.com/tensorflow/tensorflow.git

**3) python파일을 만들어 다음과 같이 작성해줍니다.**

**import** **tensorflow** **as** **tf**

**from** **tensorflow.python.tools** **import** freeze\_graph

freeze\_graph.freeze\_graph(input\_graph='./final1.pbtxt',

input\_saver="",

input\_binary=**False**,

input\_checkpoint='./checkpoint/ckp-528',

output\_node\_names='segment/Sigmoid',

restore\_op\_name="",

filename\_tensor\_name="",

output\_graph='./final1.pb',

clear\_devices=**False**,

initializer\_nodes="")

input\_graph에 pbtxt파일, input\_checkpoint에 checkpoint파일의 path를 작성해줍니다. output\_node\_names는 netron또는 tensorboard를 이용하여 출력 노드를 작성해줍니다.

output\_graph는 생성될 pb파일의 이름을 작성합니다.

**4) python 파일을 실행하면, pb파일이 생성됩니다. 이를 rknn변환에 이용하면 됩니다.**

# 프로젝트 시작하기

만들어진 rknn을 불러와 실행하는 방법입니다.

새로운 Python 파일을 만들어 줍니다.

vi run.py

**1) RKNN 객체 생성 & model 로드하기**

**from** **rknn.api** **import** RKNN

rknn = RKNN(verbose=**True**)

*#Direct Load RKNN Model*

rknn.load\_rknn('./deepfusion.rknn') *#만들어진 rknn을 로드*

print('--> load success') *#성공 메세지 출력*

**2) RKNN 런타임 실행 환경**

런타임을 실패할 경우, 에러 메시지를 출력하고 실행을 중지합니다.

*#init runtime environment*

print('--> Init runtime environment')

ret = rknn.init\_runtime()

*#오류 메세지 출력*

**if** ret != 0:

print('Init runtime environment failed')

**3) RKNN 모델 실행**

*# Inference*

print('--> Running model')

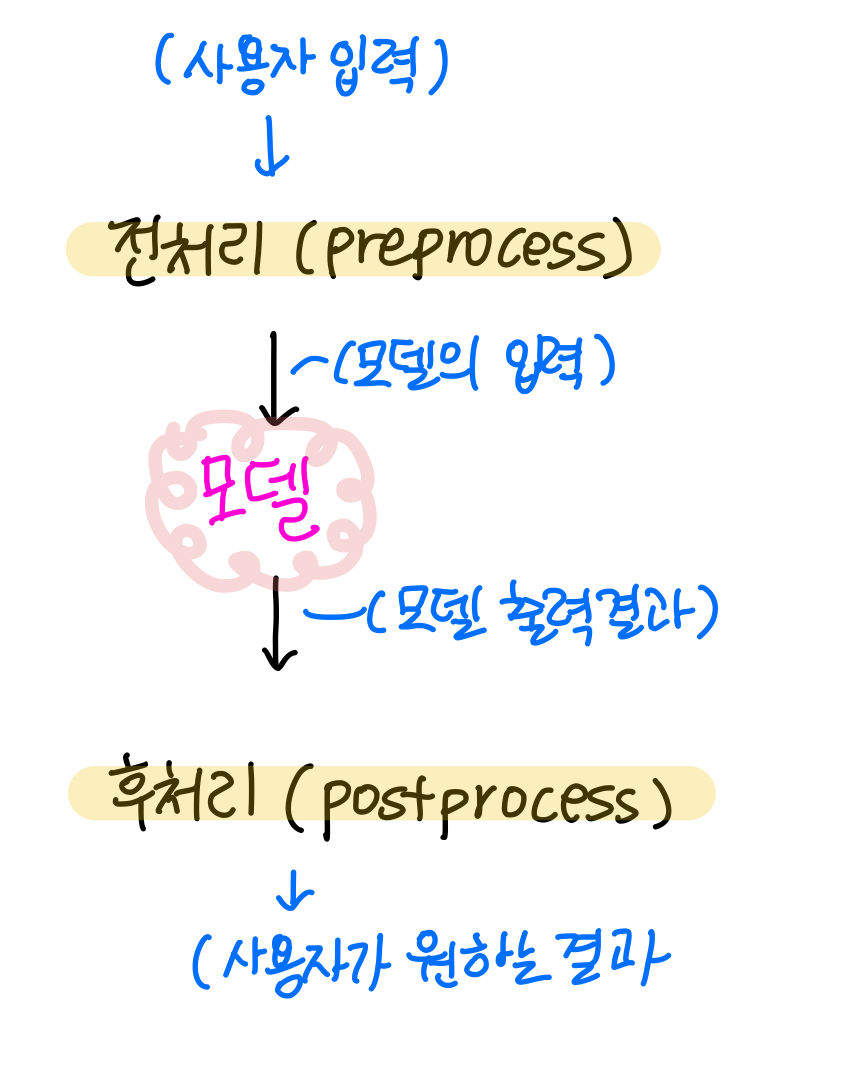
*#input 넣어 모델 돌린 후 결과 outputs에 저장*

outputs = rknn.inference(inputs=[visible\_input, infrared\_input])

print('done')

# 전처리, 후처리 과정

전처리로 사용자의 입력을 모델 입력에 맞추어 주고, 후처리를 통해 모델의 출력 결과를 사용자가 원하는 결과로 처리해줍니다.



**1) 이미지 전처리**

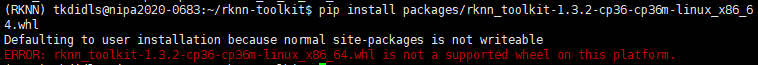
이미지를 reshape, tensor로 변환 등 모델에 원하는 입력으로 바꾸어 줍니다.

**2) 이미지 후처리**

모델의 출력 결과를 사용자가 원하는 결과로 변환하여 출력해줍니다.

# Trouble Shooting

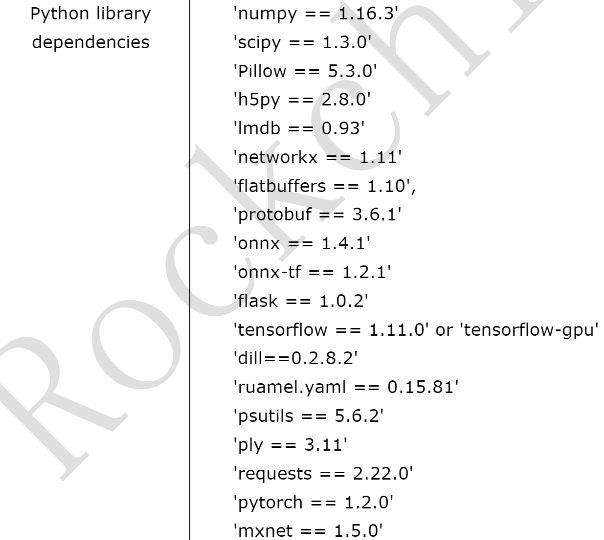
1.1 RKNN toolkit 설치 오류



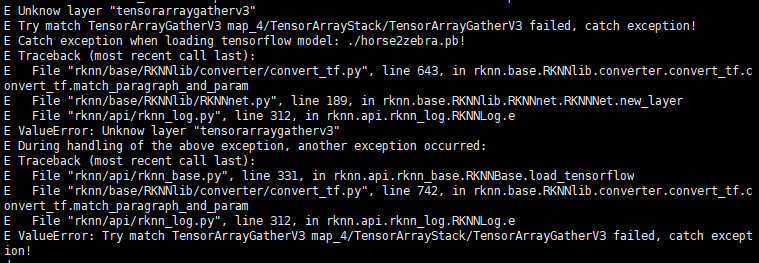
* rknn toolkit이 설치가 안되는 오류
* 현재 Python 버전을 확인하고 그에 맞는 whl 파일을 설치

ex) python3.6 > cp36, python3.5 > cp35

* 가상환경 종속성이 꼬였을 수 있으므로 가상환경 종료 후 재진입해서 다시 시도
* rknn은 필요한 서브 모듈들을 자동으로 설치해주지 않으므로 매뉴얼에 적혀 있는 서브 모듈 모두 제대로 설치했는지 확인 (아래 그림 참고)

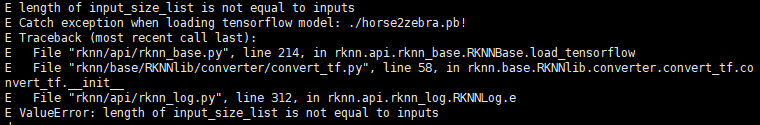


1.2 Unknow layer error

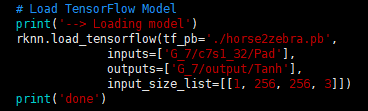


* rknn이 tensorflow의 메소드를 지원하지 않을 때 발생하는 오류
* RKNN이 지원하는 메소드 목록을 보며 지원하지 않는 메소드를 쓰진 않았나 점검
* 트레이닝을 tensorflow 1.13 이상의 환경에서 했다면 RKNN이 지원하는 메소드를 사용해도 변환이 안되는 현상이 발생하는 경우가 있으므로 트레이닝을 tensorflow 1.12 이하의 버전에서 진행해보는 것을 추천

1.3 input size error

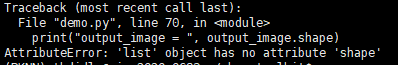


* Input 사이즈가 달라서 생기는 문제



* Input\_size\_list 수정 필요

1.4 list object error

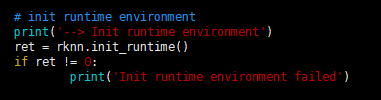


* Tensorflow에서는 모델 출력이 numpy 배열로 나오지만 RKNN의 출력은 List여서 생기는 문제
* Inference 메소드가 끝나고 난 뒤 numpy 배열로 변환해주는 작업을 넣어줄 것

1.5 run time init error



* 모델을 실행하기 전에 rknn init을 안해주면 발생하는 에러



* 모델 실행전에는 꼭 위의 코드를 넣어서 init\_runtime을 해주어야 한다