**USER MANUAL FOR TENSORFLOW MODEL CONVERTER TOOL**

Table of Contents page

[**Introduction**](#Introduction)**2**

[**Prerequisites**](#Prerequisites)**2**

[**Usage**](#Usage)**2**

[Setting up the environment](#SettingUpTheEnv) **2**

[Setting up the model converter tool](#SettingUpModelConverter) **2**

[Running the model converter tool](#RunningTheModelConverterTool)**2**

[**Supported models**](#SupportedModels)**3**

[**Behavior of model converter for supported models**](#BehaviorForSupportedModels) **3**

[**Behavior of model converter for unsupported models**](#BehaviorForUnsupportedModels) **3**

1. **Introduction:**

Vitis AI is a development stack that accelerates AI inference on Vitis hardware platforms with Vitis Runtime library. The models in Vitis AI model zoo are optimized to run on Deep Learning Processor Unit (DPU). The aim of the model converter is to convert the given VitisAI input TF model to AMD CPU optimized TF model.

1. **Prerequisites:**

The requirements include:

* Conda
* Python 3.8
* TF\_v2.10 + ZenDNN\_v4.0

1. **Usage:**
   1. **Setting up the environment:**
      1. Install conda
      2. Then set up the TF + ZenDNN environment by following the steps in <https://developer.amd.com/wp-content/resources/ZenDNN/57301_ZenDNN_TF_UG_Rev_3.3.pdf>
   2. **Setting up the model converter tool:**
      1. From the .whl file provided for the model converter, model converter tool can be installed using the following command in the command line interface:

*python -m pip install <file\_name.whl>*

Example usage is as follows:

*python -m pip install ModelConverter-0.1-py3-none-linux\_x86\_64.whl*

* 1. **Running the model converter tool:**

Run the model converter using the following command:

*model\_converter --model\_file <path to the .pb model file> --out\_location <path to the output directory>*

Example usage is as follows:

*model\_converter --model\_file ~/inception\_v4\_tf1\_1102/inception\_v4\_pruned\_0.2/quantized/quantized\_pruned\_19.56B.pb --out\_location ../outputs/*

The expected command line arguments are explained as follows:

* model\_file - The model file that needs to be optimized
* out\_location – Directory where the output model is saved. Output model is generated with a name of the format <input file name> appended by ‘\_amd\_opt.pb’

1. **Supported models:**

The quantized VitisAI TF models that are currently supported by the model converter include:

* Inception V1
* Inception V3
* Inception V4
* Resnet 50
* Resnet 101
* Resnet 152
* Vgg 16
* Vgg 19
* Mobilenet V1
* Efficientnet L
* Efficientnet M
* Efficientnet S
* RefineDet

1. **Behavior of model converter for supported models:**

Both the baseline and pruned versions of the qint8 models mentioned earlier are supported for optimization. The result is an optimized graph that will be saved at the desired output location. Model is saved only after successful optimization, and we will get the following message:



1. **Behavior of model converter for unsupported models:**

The model converter might land into one of the following issues and not generate the output:

* 1. Gracefully exits with exceptions in case of pythonic errors between the conversion
  2. Exits from the optimization process if there are any potential attribute mismatch between successive nodes or unsupported attribute combination for a node found during the conversion. The possible output messages are as follows:



(or)



* 1. Exits if it generates incorrect graph. In order not to save invalid output graphs, it is handled as below:
     1. ‘graph\_validation.py’ is called from the ‘graph\_transform.py’ which will check for the correctness of the graph
     2. It passes a random input of the required dimensions (as per the model’s input requirement) through the optimized output model and checks for discrepancies like the presence of unsupported attribute, attribute mismatch, disconnectivity and presence of cycles. If any of these are present in the output model, then the error is caught and exits giving the following message:

