



TensorFlow-ZenDNN Plug-in User Guide

Publication # CSG-1	Revision # 0.2
Issue Date October 2023	

The information contained herein is for informational purposes only, and is subject to change without notice. While every precaution has been taken in the preparation of this document, it may contain technical inaccuracies, omissions and typographical errors, and AMD is under no obligation to update or otherwise correct this information. Advanced Micro Devices, Inc. makes no representations or warranties with respect to the accuracy or completeness of the contents of this document, and assumes no liability of any kind, including the implied warranties of noninfringement, merchantability or fitness for particular purposes, with respect to the operation or use of AMD hardware, software or other products described herein. No license, including implied or arising by estoppel, to any intellectual property rights is granted by this document. Terms and limitations applicable to the purchase or use of AMD's products are as set forth in a signed agreement between the parties or in AMD's Standard Terms and Conditions of Sale.

Trademarks

AMD, the AMD Arrow logo, and combinations thereof are trademarks of Advanced Micro Devices, Inc.

Dolby is a trademark of Dolby Laboratories.

ENERGY STAR is a registered trademark of the U.S. Environmental Protection Agency.

HDMI is a trademark of HDMI Licensing, LLC.

HyperTransport is a licensed trademark of the HyperTransport Technology Consortium.

Microsoft, Windows, Windows Vista, and DirectX are registered trademarks of Microsoft Corporation.

MMX is a trademark of Intel Corporation.

OpenCL is a trademark of Apple Inc. used by permission by Khronos.

PCIe is a registered trademark of PCI-Special Interest Group (PCI-SIG).

Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.

Dolby Laboratories, Inc.

Manufactured under license from Dolby Laboratories.

Rovi Corporation

This device is protected by U.S. patents and other intellectual property rights. The use of Rovi Corporation's copy protection technology in the device must be authorized by Rovi Corporation and is intended for home and other limited pay-per-view uses only, unless otherwise authorized in writing by Rovi Corporation.

Reverse engineering or disassembly is prohibited.

USE OF THIS PRODUCT IN ANY MANNER THAT COMPLIES WITH THE MPEG-2 STANDARD IS EXPRESSLY PROHIBITED WITHOUT A LICENSE UNDER APPLICABLE PATENTS IN THE MPEG-2 PATENT PORTFOLIO, WHICH LICENSE IS AVAILABLE FROM MPEG LA, L.L.C., 6312 S. FIDDLERS GREEN CIRCLE, SUITE 400E, GREENWOOD VILLAGE, COLORADO 80111.

Contents

Revision History	6
Chapter 1 Installing TensorFlow-ZenDNN Plug-in	7
1.1 TensorFlow-ZenDNN Plug-in Setup	7
1.1.1 Install with Binary Release	7
1.1.2 Build and Install from Source	7
Chapter 2 High-level Overview	9
Chapter 3 TensorFlow-ZenDNN Plug-in v0.2	10
Chapter 4 Environment Variables	11
Chapter 5 Tuning Guidelines	13
5.1 System	13
5.2 Environment Variables	13
Chapter 6 Convolution Algorithm Logic	15
Chapter 7 Technical Support	16

List of Figures

Figure 1.	TensorFlow-ZenDNN Plug-in	9
-----------	---------------------------------	---

List of Tables

Table 1.	ZenDNN Environment Variables-Generic	11
Table 2.	ZenDNN Environment Variables-Optimization	12
Table 3.	System Specification.	13
Table 4.	Convolution Algorithm Logic.	15

Revision History

Date	Revision	Description
October 2023	0.2	<ul style="list-style-type: none">• Updated supported ZenDNN and TensorFlow versions.• Added section 1.1.2 (Build and Install from Source).• Removed Chapter 6 (Blocked Format Support).
March 2023	0.1	Initial version.

Chapter 1 Installing TensorFlow-ZenDNN Plug-in

Note: Refer to the [ZenDNN 4.1 User Guide](#) before starting the installation.

1.1 TensorFlow-ZenDNN Plug-in Setup

This section describes the procedure to setup the TensorFlow-ZenDNN plug-in for TensorFlow v2.14.

1.1.1 Install with Binary Release

Complete the following steps to install the TensorFlow-ZenDNN plug-in binary release:

1. Install TensorFlow v2.14:

```
pip install tensorflow-cpu==2.14.0
```

2. Download the TensorFlow-ZenDNN plug-in wheel file from the [Community supported TensorFlow builds](#).

3. Install TensorFlow-ZenDNN plug-in:

```
$ pip install tensorflow_zendnn_plugin-0.2.0-cp39-cp39-linux_x86_64.whl
```

The release binaries for TensorFlow-ZenDNN plug-in v0.2 are compiled with manylinux2014 and they provide compatibility with some older Linux distributions.

The release binaries are tested with the recent Linux distributions such as:

- Ubuntu 20.04 and later
- RHEL 9.0 and later

1.1.2 Build and Install from Source

To build and install the TensorFlow-ZenDNN plug-in from source:

Clone the repository:

```
$ git clone https://github.com/amd/ZenDNN-tensorflow-plugin.git  
$ cd ZenDNN-tensorflow-plugin/
```

The repository defaults to the master development branch. To build the TensorFlow-ZenDNN plug-in v0.2, you must check out the tag v0.2.

Follow the [build and install from source steps](#) to configure, build, and install the TensorFlow-ZenDNN plug-in.

Enable TensorFlow-ZenDNN Plug-in

Set the following environment variables to enable ZenDNN for inference:

- **TF_ENABLE_ZENDNN_OPTS=1**
- **TF_ENABLE_ONEDNN_OPTS=0**

By default, TensorFlow is shipped with oneDNN enabled.

To run a sample with the installed TensorFlow-ZenDNN plug-in, follow the instructions in [Unified Inference Frontend \(UIF\) 1.1 User Guide - Run a CPU Example](#).

Chapter 2 High-level Overview

The following is a high-level block diagram for the TensorFlow-ZenDNN plug-in package which utilizes ZenDNN as the core inference library:

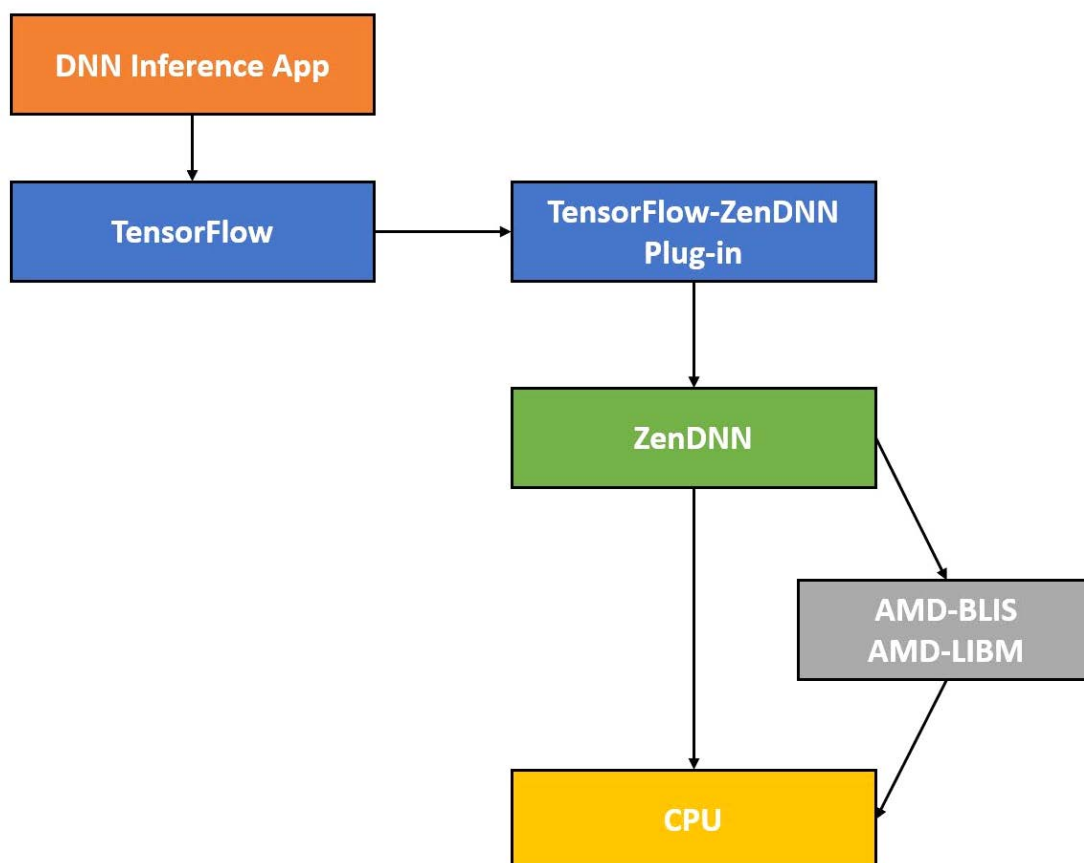


Figure 1. TensorFlow-ZenDNN Plug-in

Chapter 3 TensorFlow-ZenDNN Plug-in v0.2

TensorFlow-ZenDNN plug-in v0.2 is the AMD CPU plug-in release with Pluggable device approach of TensorFlow:

- This plug-in v0.2 is supported for TensorFlow v2.14 and later.
- The TensorFlow-ZenDNN plug-in open source is available at:
<https://github.com/amd/ZenDNN-tensorflow-plugin>
- It is integrated with ZenDNNv4.1 as the core inference library and compiled with GCC v9.3.1.
- As compared to the current TensorFlow-ZenDNN direct integration releases, this release provides:
 - On par performance for models, such as RefineDet, Inception, and VGG variants.
 - Sub-optimal performance for models, such as ResNet, MobileNet and EfficientNet.

Chapter 4 Environment Variables

TensorFlow-ZenDNN plug-in uses the following environment variables to tune performance and control logs:

Table 1. ZenDNN Environment Variables-Generic

Environment Variable	Default Value/User Defined Value
ZENDNN_LOG_OPTS	ALL:0
TF_ZEN_PRIMITIVE_REUSE_DISABLE	False
ZENDNN_ENABLE_MEMPOOL	The default value is set to 1, you can provide the value 0 to disable it. 1 is for Graph-based MEMPOOL and 2 is for Node-based MEMPOOL.
ZENDNN_PRIMITIVE_CACHE_CAPACITY	The default value is set to 1024, you can modify it as required
ZENDNN_TENSOR_BUF_MAXSIZE_ENABLE	0
TF_ENABLE_ZENDNN_OPTS	Default value is set to 0. Set it to 1 along with TF_ENABLE_ONEDNN_OPTS=0 for enabling ZenDNN for inference. You can set it to 0 when you want to enable vanilla training and inference.
TF_ENABLE_ONEDNN_OPTS	Default value is set to 1. By default, TensorFlow is shipped with oneDNN optimizations enabled. Hence, set it to 0 when you enable ZenDNN.

Table 2. ZenDNN Environment Variables-Optimization

Environment Variable	Default Value/User Defined Value
OMP_NUM_THREADS	Set it as per the number of cores in the user system ^a .
OMP_DYNAMIC	Set it to FALSE for optimal performance ^a .
OMP_PROC_BIND	Set it to FALSE for optimal performance ^a .
GOMP_CPU_AFFINITY	Set it as per the number of cores in the system being used ^a .
ZENDNN_TENSOR_POOL_LIMIT	The default value is set to 1024. You can modify it to 512 for CNNs for optimal performance.
ZENDNN_CONV_ALGO	<p>The default value is set to 1. Decides the convolution algorithm to be used in execution and the possible values are:</p> <ul style="list-style-type: none"> • 1 = im2row followed by GEMM • 2 = WinoGrad (fallback to im2row GEMM for unsupported input sizes) • 3 = Direct convolution with blocked inputs and filters • 4 = Direct convolution with blocked filters
ZENDNN_GEMM_ALGO	<p>The default value is 3. You can modify it to one of the following:</p> <ul style="list-style-type: none"> • 0 = Auto • 1 = AOCL-BLIS path • 2 = Partial AOCL-BLIS • 3 = ZenDNN JIT path • 4 = ZenDNN partial JIT path <p><i>Note: Auto is an experimental feature and should be used with application warm-up iteration >=15.</i></p>

a. You must set these environment variables explicitly.

Chapter 5 Tuning Guidelines

The hardware configuration, OS, Kernel, and BIOS settings play an important role in performance. The details for the environment variables used on a 4th Gen AMD EPYC™ server to achieve the optimal performance numbers are as follows:

5.1 System

A system with the following specifications has been used:

Table 3. System Specification

Model name	4 th Gen AMD EPYC™ 9654P 96-Core Processor
DPU MHz	Up to 3.7 GHz
No of Cores	96
1P/2P	1
SMT: Thread(s) per Core	2
Mem-Dims	12x64 GB

OS Used: Ubuntu 20.04.02 LTS

5.2 Environment Variables

The following environment variables have been used:

ZENDNN_LOG_OPTS=ALL:0

OMP_NUM_THREADS=96

OMP_WAIT_POLICY=ACTIVE

OMP_PROC_BIND=FALSE

OMP_DYNAMIC=FALSE

ZENDNN_ENABLE_MEMPOOL=1

ZENDNN_GEMM_ALGO=3

***Note:** For NLP and Recommender models, better performance is observed with ZENDNN_GEMM_ALGO=4. However, these details should be verified empirically.*

ZENDNN_TENSOR_POOL_LIMIT=1024

ZENDNN_TENSOR_BUF_MAXSIZE_ENABLE=0

ZENDNN_CONV_ALGO=4

ZENDNN_PRIMITIVE_CACHE_CAPACITY=1024

TF_ENABLE_ZENDNN_OPTS=1

TF_ENABLE_ONEDNN_OPTS=0

GOMP_CPU_AFFINITY=0-95

The environment variables **OMP_NUM_THREADS**, **OMP_WAIT_POLICY**, **OMP_PROC_BIND**, and **GOMP_CPU_AFFINITY** can be used to tune performance. For optimal performance, the **Batch Size** must be a multiple of the total number of cores (used by the threads). On a 4th Gen AMD EPYC™ server (configuration: AMD EPYC™ 9654P 96-Core, 2P, and **SMT=ON**) with the above environment variable values, **OMP_NUM_THREADS=96** and **GOMP_CPU_AFFINITY=0-95** yield the best throughput numbers for a single socket.

A few of the models (for example, publicly available ResNet50) gain performance with Transparent Huge Pages settings (THP). THP can be enabled as a sudo user using the following command:

```
echo always > /sys/kernel/mm/transparent_hugepage/enabled
```

For more information, refer to the Tuning Guidelines section of the [ZenDNN 4.1 User Guide](#).

Chapter 6 Convolution Algorithm Logic

Convolution kernels take Input and Filter/Weights as arguments and return Output. The table below describes the expected Layout for each of the convolution algorithms currently supported by TensorFlow-ZenDNN.

Table 4. Convolution Algorithm Logic

zenConvAlgoType	ZENDNN_CONV_ALGO	Input Layout	Filter Layout	Output Layout
GEMM	1	NHWC	HWIO	NHWC
WINOGRAD	2	NHWC	HWIO	NHWC
DIRECT1	3	nChw8c	Ohwi8o	nChw8c
DIRECT2	4	NHWC	Ohwi8o/ Ohwi16o	NHWC

Note: In the context of Filter Layouts, HWIO is equivalent to HWCN but with I instead of C representing input channels and O instead of N representing output channels.

Chapter 7 Technical Support

Please email zendnnsupport@amd.com for questions, issues, and feedback.