



PYTORCH  
AMPERE<sup>®</sup> AI OPTIMIZER (AIO)  
Documentation  
v.1.1



## Table of Contents

<b>RELEASE NOTES .....</b>	<b>2</b>
V1.1.....	2
<b>OVERVIEW.....</b>	<b>2</b>
<b>PYTORCH FRAMEWORK.....</b>	<b>2</b>
Versions Compatibility.....	2
Python.....	2
<b>AIO CONFIGURATIONS.....</b>	<b>2</b>
<b>QUICKSTART.....</b>	<b>4</b>

## RELEASE NOTES

### V1.1

- AIO updated to 0.2.1
- Batch Matmul supported (enhancing DLRM performance)
- Adaptive Avg Pool supported
- LeakyRelu supported
- AIO\_NUM\_THREADS no longer needed to set AIO threads. The AIO inherits Pytorch intra-op thread count.

## OVERVIEW

The Ampere® AIO inference acceleration engine is fully integrated with the PyTorch framework. PyTorch models and software written with the PyTorch API can run as-is, without modifications.

## PYTORCH FRAMEWORK

Python is installed with the AIO accelerated Pytorch package and all dependencies. No additional installation steps are needed.

### Versions Compatibility

This release is based on Pytorch 1.9.0 and comes with the compatible Torchvision 0.10.0 installed.

### Python

Pytorch 1.9.0 is built for Python 3.8. Regarding other Python versions, please contact your Ampere sales representative. If you are using the software through a third party, contact their customer support team for help. You can also contact the AI team at [ai-support@amperecomputing.com](mailto:ai-support@amperecomputing.com).

## AIO CONFIGURATIONS

The AIO inference engine can be configured by a set of environment variables for performance and debugging purposes. They can be set in the command line when running Pytorch models (e.g. AIO\_NUM\_THREADS=16 python run.py -p fp32) or set in the shell initialization script.

### AIO\_PROCESS\_MODE

This variable controls whether the AIO inference engine is used to run the Pytorch model.

- 0: AIO is disabled
- 1: AIO is enabled (Default)

### **AIO\_CPU\_BIND**

Enables core binding. If enabled, each AIO thread will bind itself to a single core.

- 0: Core binding disabled
- 1: Core binding enabled (Default)

### **AIO\_MEM\_BIND**

Bind memory to NUMA (Non-uniform memory access) node 0. For optimal performance, numactl (<https://linux.die.net/man/8/numactl>) is preferred. numactl bind will affect both the Pytorch framework and the AIO buffers, while the AIO is unable to affect buffers allocated by the Pytorch framework.

- 0: Membind disabled
- 1: Membind to node 0 (Default)

### **AIO\_NUMA\_CPUS**

Select the cores that the AIO should bind to (if CPU\_BIND is enabled).

- Not set: AIO will use the first N cores of the machine, excluding hyper-threaded (Default)
- Set: AIO will try to use N first cores from the list of cores for N threads. The list is in space-separated, 0-based number format. For example, selecting cores 0 to 1:  
AIO\_NUMA\_CPUS="0 1"

### **AIO\_NUM\_THREADS**

Specifies the number of cores that the AIO should use.

- Not set: AIO will use one core (Default)
- "all": AIO will use all cores, as specified by AIO\_NUMA\_CPUS
- N: AIO will use N cores

### **AIO\_DEBUG\_MODE**

Control the verbosity of debug messages.

- 0: No messages
- 1: Errors only
- 2: Basic information, warnings, and errors (Default)
- 3: Most messages
- 4: All messages

## QUICKSTART

The following instructions run on Altra/Altra Max Linux machines installed **with Docker**. To initialize the AIO environment run:

```
$ wget -O aio-pytorch.tar.gz "<your_unique_url>"
$ docker load < aio-pytorch.tar.gz
$ docker run --privileged=true --rm --name pytorch-aio --network host -it aio-pytorch-1.9.0:1.1.3
```

Skip the above steps if running **without a Docker** container.

You can try the AIO by either running the Jupyter Notebook examples or Python scripts on the CLI level.

To run the Jupyter Notebook QuickStart examples follow the instructions below:

Set AIO\_NUM\_THREADS to the requested value first.

```
$ export AIO_NUM_THREADS=16
$ cd ~/aio-examples/
$ bash start_notebook.sh
```

If you run the Jupyter Notebook Quickstart on a cloud instance, make sure your machine has port 8080 open and on your local device run:

```
$ ssh -N -L 8080:localhost:8080 -I <ssh_key> your_user@xxx.xxx.xxx.xxx
```

Use a browser to point to the URL printed out by the Jupyter Notebook launcher.

You will find Jupyter Notebook examples (examples.ipynb) under the /classification and /object detection folders.

The examples run through several inference models, visualize results they produce, and present the performance numbers.

To use CLI-level scripts:

Set AIO\_NUM\_THREADS to the requested value first.

```
$ export AIO_NUM_THREADS=16
$ cd ~/aio-examples/
```

Go to the directory of choice, e.g.

```
$ cd classification/resnet_50_v1
```

Evaluate the model.

```
$ numactl --physcpubind=0-15 python3 run.py -p fp32
```

---

**Ampere Computing® / 4655 Great America Parkway, Suite 601 / Santa Clara, CA 95054 /  
www.amperecomputing.com**

Ampere Computing, the Ampere Computing logo, Altra, and eMAG are registered trademarks of Ampere Computing.

Arm is a registered trademark of Arm Holdings in the US and/or elsewhere. All other trademarks are the property of their respective owners.

©2022 Ampere Computing. All rights reserved.

AMP 2019-0039