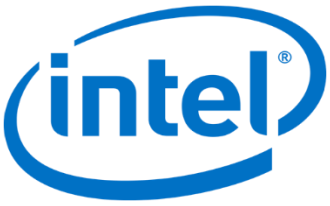

Intel® Software Optimization for Theano* Installation Guide

Version 1.3

1 Source: https://github.com/intel/theano/blob/master/Install_Guide.pdf

Contents

List of Tables

Revision history

Version	Date	Name	Comments
1.0	2016-04-06	SSG/PCS Team	Initial Version
1.1	2016-07-04	SSG/PCS Team	Fix typo and add Python3.4 description
1.2	2016-07-16	SSG	Break into installation guide and benchmarking guide
1.3	2016-09-20	SSG	Release version

1 Introduction

Welcome to the Intel® Software Optimization for Theano* Installation Guide. This document introduces a new version of Theano* (Optimized-Theano*). This document also introduces a new version of NumPy* (Optimized-NumPy*). Theano is a Python* library, and NumPy is the fundamental package for doing scientific computing with Python.

This guide explains how to install the new versions of Optimized-Theano and Optimized-NumPy. These new versions of Optimized-Theano and Optimized-NumPy support both Python* 2.7 and Python 3.4.

If you already have a working version of Theano on your machine, you can simply upgrade your existing NumPy and Theano. In this case, skip to the installation instructions for Optimized-Theano and Optimized-NumPy. If you're new to Theano, follow the instructions listed in the installation sections, in order to set up your environment appropriately for Optimized-Theano.

Common terms used in this guide are listed at the end of this guide, in the glossary.

2 Installation

This section explains configuration requirements, as well as some recommendations based on extensive workload tests.

2.1 Configuration recommendations

Intel performed extensive workload tests on different machine configurations with Optimized-Theano and out-of-the-box Theano. The results showed that the performance of Optimized-Theano improved significantly on several workloads. Overall, Optimized-Theano showed an average 3.63x (GEOMEAN) performance advantage over out-of-the-box Theano.[1] Results also showed an average 1.17x (GEOMEAN) performance gain for Optimized-Theano, when executed with the Intel C++ Compiler.[1]

Based on the results of these tests, Intel strongly recommends that, when you use Optimized-Theano, you also:

- Use Optimized-NumPy.
- Use Intel® C++ Compiler (icpc).
- Use Intel® Math Kernel Library (Intel® MKL). This is a free, high performance math library for x86 and x86-x64 architectures. You can download the library from the Intel Web site.
- Upgrade your compilers and math libraries to the versions listed in this guide.

2.2 Requirements

The following table shows the requirements for installing Optimized-Theano. The list after the table provides additional installation and/or upgrade recommendations.

Table 1 Requirements for installing Optimized-Theano

Type	Requirement
Theano	Optimized-Theano (based on Theano 0.9.0dev1)
Math-related	Optimized-NumPy (based on NumPy v1.11.0rc1) Intel® Math Kernel Library (Intel® MKL) Scipy*-0.13.1
Compiler	Intel® C++ Compiler GNU compiler 4.8 (g++/gfortran) GNU binutils (>=2.4)
Python Software Foundation Python* 2.7	Python2.7.6 python-dev python-setuptools
Python 3.4	Python 3.4 libpython3-dev python3-dev python3-setuptools
Others	git* pip* nose* Cython* (required for recent version of NumPy) p7zip* (required for decompressing dataset for DBN-Kyoto workload)

To optimize CPU performance, you should install/upgrade your compilers and math libraries to these versions:

- GNU compiler = 4.8.x
- GNU binutils >= 2.4
- Intel C++ Compiler = 16.0
- Intel MKL = 11.3
- Optimized-NumPy

2.3 Installation

To optimize performance, you should use both the Intel C++ Compiler and Intel MKL. This discussion provides the download links for the compiler and math library, and then explains how to install the compiler and library.

2.3.1 Download and install the Intel® C++ Compiler

This discussion provides links for downloading the Intel C++ Compiler, then provides a sample script for installing the compiler.

2.3.1.1 Download the compiler

You can purchase the Intel C++ Compiler by following the instructions on this Web site:

https://registrationcenter.intel.com/en/forms/?productid=2558&licensetype=2

You can get the 30-day free evaluation version of the Intel C++ Compiler via this link:

https://software.intel.com/en-us/intel-parallel-studio-xe/try-buy#buynow

Students, educators, academic researchers, and open-source contributors may qualify for the Intel program: Free Software Tools. To see if you qualify for the program, use this link:

https://software.intel.com/en-us/qualify-for-free-software

2.3.1.2 Install both the Intel C++ Compiler and Intel® MKL

You can install the compiler and math library with root privilege for all users, or install with user-level access for just the current user. If you do not want to use the Intel C++ Compiler and want to use only the math library, skip to the next discussion.

To install both the Intel C++ Compiler and Intel MKL, use the following set of commands:

```
tar xvf parallel_studio_xe_2016.tgz
cd parallel_studio_xe_2016
./install.sh      # You will be prompted to enter your product license
cd ~
echo "source /opt/intel/compilers_and_libraries/linux/bin/compilervars.sh intel64" >>
~/.bashrc      # Replace the quoted part with your installation path
source ~/.bashrc
```

To verify the installation, run this command:

```
which icc
```

If the installation is verified, the output should be similar to this line:

/opt/intel/compilers_and_libraries_2016.0.109/linux/bin/intel64/icc

If the installation failed, your screen will display nothing. In this case, repeat the installation and verification process.

2.3.2 Install only the Intel MKL (optional)

Intel MKL is a high performance math library for x86 and x86-x64 architectures. It is available for Microsoft Windows,* Linux,* and Apple OS X* operating systems. The math library is free, and can be downloaded from the Intel Web site.

You can install the math library automatically as part of the installation of the Intel C++ Compiler (described in the previous discussion). Or, you can choose to install only the math library. To install only the math library follow these steps:

1. Click here to go to the Intel Web site.
2. Follow the instructions to register and download the free math library.
3. After the download is complete, run the following commands to install the math library:

```
tar xzvf l_mkl_p_11.3.x.yyy_ia64.tgz
./install.sh
```

2.3.3 Optimize BIOS/OS settings for hyperthreading and NUMA

Intel performed extensive tests on the compiler and math library, using different workload, with different BIOS settings for hyperthreading (HT) and NUMA optimization. The results showed that performance improved significantly when both hyperthreading and NUMA optimization were disabled.[1]

Intel recommends that you adjust BIOS/OS settings as appropriate for your configuration, in order to improve performance. There are two options for doing this: Disable hyperthreading and NUMA, or manually setting the number of threads to the number that is optimal for your application, based on the relevant workload.

2.3.3.1 Option 1: Disable hyperthreading and NUMA

To disable hyperthreading and NUMA optimization, you must reboot your machine, enter BIOS, and change the configuration. Follow these steps to disable those two BIOS options:

1. Reboot your machine.
2. Enter BIOS.
3. In the processor configuration section, disable hyperthreading.
4. In the memory configuration section, disable NUMA optimization .
5. Unlock the 100GB memory limitation by adding the following line to the end of **~/.bashrc** or **/etc/bash.bashrc:**

```
ulimit -m unlimited -v unlimited
```

2.3.3.2 Option 2: Set optimal number of threads for your application

The second option for improving performance is to set the number of threads for the math library (MKL) and for open multi-processing (OpenMP) to the optimum number of threads, as determined by the relevant workload for your application. As shown in Table 2, the optimal number of threads has been determined for the Intel C++ compiler (icpc) via various workload tests, for the options for the Linux environment parameters MKL_NUM_THREADS and OMP_NUM_THREADS.

Table 2 Number of threads that optimize performance for various workloads[1]

Number of threads that produce the best performance of Optimized-Theano, when running on an Intel® Xeon® processor E5-2699 v3	
Workload	**Compiler: icpc**
RBM	MKL=6, OMP=6
RNNRBM	MKL=6, OMP=6
LSTM	OMP=18
LSTM-Sentiment-Analysis	OMP=18
RNNSLU-Word_Embeddings (elman-forward)	MKL=6, OMP=6
RNNSLU-Word_Embeddings (jordan-forward)	MKL=6, OMP=6

2.3.4 Install Python-related packages

This discussion explains how to install Python-related packages, Optimized-NumPy, SciPy packages, and Optimized-Theano.

2.3.4.1 Install Python-related packages

Below are the installation commands to install the Python-related packages on Ubuntu-based or CentOS-based platforms. Select the installation appropriate for your environment:

- **For Ubuntu:** Enter the appropriate command for the Python version you will be using:

```
Python 2.7: sudo apt-get install python-dev python-setuptools
Python 3.4: sudo apt-get install libpython3-dev python3-dev python3-setuptools
```

- **For CentOS:** Enter the appropriate command for the Python version you will be using:

```
Python 2.7: sudo yum install python-setuptools python-dev
Python 3.4: sudo yum install libpython3-dev python3-dev python3-setuptools
```

2.3.4.2 Install Optimized-NumPy

Follow these steps to install Optimized-NumPy:

1. If you already have a NumPy installed on your machine, use the following command to get the location of your existing NumPy. (The path is a sample path.)

```
python -c "import numpy; print numpy"
<module 'numpy' from '/usr/local/lib/python2.7/dist-packages/numpy/__init__.pyc'>
```

2. Remove your existing NumPy, by entering the following line:

```
#rm -rf /usr/local/lib/python2.7/dist-packages/numpy/__init__.pyc'
```

3. Now build and install the updated Optimized-NumPy, using the following sample commands:

```
easy_install --user cython    (or: pip install cython)
git clone https://github.com/pcs-theano/numpy.git pcs-numpy
cd pcs-numpy
# make sure site.cfg is to set to the right path of Intel® MKL before executing below commands
python setup.py config --compiler=intelem build_clib --compiler=intelem build_ext --
compiler=intelem install --user
```

   If the build fails with the following error message:

```
ld: cannot find -lstdc++
```

   a. Make sure you have installed the g++ compiler.

   b. Now create a soft link to libstdc++.so.6, using the sudo commands shown here:

```
sudo apt-get install g++
sudo ln -s /usr/lib/x86_64-linux-gnu/libstdc++.so.6 /usr/lib/gcc/x86_64-linux-
gnu/4.8/libstdc++.so
```

4. Now use the following code to check the installation status of Optimized-NumPy, so you can make sure it is using the Intel MKL. If successful, you will see that library_dirs and include_dirs point to the Intel MKL installation location. If unsuccessful, you will see a blank or some other basic linear algebra subprograms (BLAS) location at the Intel MKL installation location.

*Note: If you are using some other BLAS — such as OpenBLAS — make sure set the correct path of OpenBLAS in **site.cfg** before trying to install the scipy.*

```
python -c "import numpy; print numpy.show_config()"

### the expected output should be as below:
lapack_opt_info:
    libraries = ['mkl_rt', 'pthread']
    library_dirs = ['/opt/intel/mkl/lib/intel64']
    define_macros = [('SCIPY_MKL_H', None), ('HAVE_CBLAS', None)]
    include_dirs = ['/opt/intel/mkl/include']
blas_opt_info:
    libraries = ['mkl_rt', 'pthread']
    library_dirs = ['/opt/intel/mkl/lib/intel64']
    define_macros = [('SCIPY_MKL_H', None), ('HAVE_CBLAS', None)]
    include_dirs = ['/opt/intel/mkl/include']
openblas_lapack_info:
  NOT AVAILABLE
lapack_mkl_info:
    libraries = ['mkl_rt', 'pthread']
    library_dirs = ['/opt/intel/mkl/lib/intel64']
    define_macros = [('SCIPY_MKL_H', None), ('HAVE_CBLAS', None)]
    include_dirs = ['/opt/intel/mkl/include']
blas_mkl_info:
    libraries = ['mkl_rt', 'pthread']
    library_dirs = ['/opt/intel/mkl/lib/intel64']
    define_macros = [('SCIPY_MKL_H', None), ('HAVE_CBLAS', None)]
    include_dirs = ['/opt/intel/mkl/include']
mkl_info:
    libraries = ['mkl_rt', 'pthread']
    library_dirs = ['/opt/intel/mkl/lib/intel64']
    define_macros = [('SCIPY_MKL_H', None), ('HAVE_CBLAS', None)]
    include_dirs = ['/opt/intel/mkl/include']
```

2.3.4.3 Install SciPy

You can install SciPy via a sudo command, or you can install SciPy from a source package. Select the installation command from the following list, as appropriate for your environment:

- Install via pip command (recommended)

```
pip install --user scipy
```

- **Install on a Ubuntu-based system:** Enter the appropriate command for the SciPy version you will be using:

```
Python 2.7: sudo apt-get install python-scipy
Python 3.4: sudo apt-get install python3-scipy
```

- **Install on a CentOS-based system:** Enter the appropriate command for the SciPy version you will be using:

```
Python 2.7: sudo yum -y install python-scipy
Python 3.4: sudo yum -y install python3-scipy
```

- **Install from source package:** To install SciPy from a source package, use these commands:

```
tar xzvf scipy-0.13.1.tar.gz
cd scipy-0.13.1
python setup.py build
python setup.py install
```

2.3.4.4 Install Optimized-Theano

To install Optimized-Theano on your machine, follow these steps:

1. First, enter these commands:

```
git clone https://github.com/intel/theano.git pcs-theano
cd pcs-theano
python setup.py build
python setup.py install --user
```

2. Now clean up the Theano cache. When you update or pull new codes from GitHub, you must clean up the Theano cache after the Theano installation. Use the following command to clean up the cache. (You can also simply remove the files within directory of ~/.theano/compiledir*/.)

```
theano-cache clear
```

3. Now overwrite the file **~/.theanorc** with the file **theanorc_icc** from pcs-theano, in order to enable Intel C++ Compiler and Intel MKL. To do this, enter these commands:

```
cp pcs-theano/theanorc_icc ~/.theanorc
cat ~/.theanorc
### the expected output should be as below:
[cuda]
root = /usr/local/cuda
[global]
device = cpu
floatX = float32
cxx = icpc
mode = FAST_RUN
openmp = True
openmp_elemwise_minsize = 10
[gcc]
cxxflags = -qopenmp -march=native -O3 -qopt-report=3 -fno-alias -opt-prefetch=2 -fp-trap=none
[blas]
ldflags = -lmkl_rt
```

*Note: To use a g++ compiler and OpenBLAS, replace **~/.theanorc** with file **theanorc_gcc** from pcs-theano.*

3 Verify the installation

Follow these steps to verify the installation:

- Verify Optimized-NumPy. Run the following command to check the current version of NumPy on your system:

  ```
  python -c "import numpy; print numpy.__version__"
  ```

  If the output is "1.11.0rc1", you have installed Optimized-NumPy correctly.

  If the output is "***No module named numpy***", or a version number other than 1.11.0rc1, follow the instructions in the previous section to reinstall Optimized-NumPy.

- Verify Optimized-Theano. Run the following command to verify the installation of Optimized-Theano:

  ```
  python -c "import theano; print theano.__version__"
  ```

  If as the output is "0.9.0dev1.dev-3bfbeee183096cd282a083063f7f1ce4f88ed4f6", you have installed Optimized-Theano correctly.

  If the output is "***No module named theano***", or a version number other than that which is listed above, follow the instructions in the previous section to reinstall Optimized-Theano.

4 Summary

This guide introduces a new version of Optimized-Theano based on Theano version 0.9.0dev1. This guide also shows how to install the new version of Optimized-Theano. The optimized version has demonstrated an average performance gain of 3.63x (GEOMEAN) over out-of-the-box versions of Theano, as shown by various workloads.[1] In addition, using the Intel C++ Compiler can further improve the performance of Optimized-Theano by up to 1.17x (GEOMEAN) over a g++ compiler.[1] Based on the results of extensive workloads, it is strongly recommended that you:

- Use Optimized-NumPy.
- Use Intel C++ Compiler (icpc).
- Use Intel Math Kernel Library (Intel MKL).
- Upgrade your compilers and math libraries to the versions listed earlier in this guide.

5 For more information

For more information about Theano and NumPy, visit these Web sites:

https://github.com/Theano

https://github.com/numpy

https://software.intel.com/en-us/articles/numpyscipy-with-intel-mkl

http://deeplearning.net/tutorial/

6 Glossary

This guide uses the following terms:

CA	Contractive auto-encoders
CNN	Convolutional neural network
DA	Denoising auto-encoders
DBN	Deep belief network
HMC	Hybrid Monte-Carlo sampling
LR	Logistic regression
LSTM	Long short term memory
MKL	Intel Math Kernel Library
MLP	Multi-layer perceptron
NUMA	Non-uniform memory access
OpenMP	Open multi-processing
RBM	Restricted Boltzmann machine
RNN	Recurrent neural network
SDA	Stacked denoising auto-encoders
SLU	Spoken language understanding