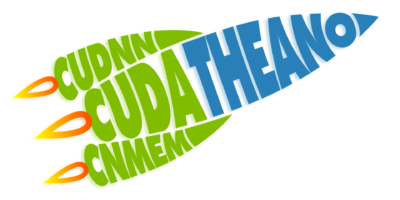
<http://ankivil.com/installing-keras-theano-and-dependencies-on-windows-10/>

# [Installing Keras, Theano and Dependencies on Windows 10](http://ankivil.com/installing-keras-theano-and-dependencies-on-windows-10/)

**Posted by**[**Fabien Tencé**](http://ankivil.com/author/fabientence/)**on**[**27 September 2016**](http://ankivil.com/installing-keras-theano-and-dependencies-on-windows-10/)[**96 Comments**](http://ankivil.com/installing-keras-theano-and-dependencies-on-windows-10/#comments)

[](http://ankivil.com/installing-keras-theano-and-dependencies-on-windows-10/)

**READ NEXT →**

[[](http://ankivil.com/making-theano-faster-with-cudnn-and-cnmem-on-windows-10/)](http://ankivil.com/making-theano-faster-with-cudnn-and-cnmem-on-windows-10/" \o "Making Theano Faster with CuDNN and CNMeM on Windows 10)

**[SOFTWARE](http://ankivil.com/making-theano-faster-with-cudnn-and-cnmem-on-windows-10/" \o "Making Theano Faster with CuDNN and CNMeM on Windows 10)**

### [Making Theano Faster with CuDNN and CNMeM on Windows 10](http://ankivil.com/making-theano-faster-with-cudnn-and-cnmem-on-windows-10/" \o "Making Theano Faster with CuDNN and CNMeM on Windows 10)

[[](http://ankivil.com/installing-keras-theano-and-dependencies-on-windows-10-python-34/)](http://ankivil.com/installing-keras-theano-and-dependencies-on-windows-10-python-34/" \o "Installing Keras, Theano and Dependencies on Windows 10 – Old way with Python 3.4)

**[SOFTWARE](http://ankivil.com/installing-keras-theano-and-dependencies-on-windows-10-python-34/" \o "Installing Keras, Theano and Dependencies on Windows 10 – Old way with Python 3.4)**

### [Installing Keras, Theano and Dependencies on Windows 10 – Old way with Python 3.4](http://ankivil.com/installing-keras-theano-and-dependencies-on-windows-10-python-34/" \o "Installing Keras, Theano and Dependencies on Windows 10 – Old way with Python 3.4)

[[](http://ankivil.com/choosing-a-deep-learning-software/)](http://ankivil.com/choosing-a-deep-learning-software/" \o "Choosing a Deep Learning Software)

**[SOFTWARE](http://ankivil.com/choosing-a-deep-learning-software/" \o "Choosing a Deep Learning Software)**

### [Choosing a Deep Learning Software](http://ankivil.com/choosing-a-deep-learning-software/" \o "Choosing a Deep Learning Software)

# Introduction

In [a previous article](http://ankivil.com/installing-keras-theano-and-dependencies-on-windows-10-python-34), I described a procedure to install Keras and Theano on Windows 10. The main downside of this procedure is that you had to downgrade to Python 3.4. This tutorial is an improved version which allows you to make Theano and Keras work with Python 3.5 on Windows 10.

Unless stated otherwise, all install options are left to their default/recommended values. In case you change installation paths, make sure you also change them in all subsequent commands.

# Installing CUDA

This step is optional but highly recommended for NVidia card owners as it massively speeds up Theano by allowing it to use the GPU:

1. Download and install [Visual Studio Community 2013 update 5](https://go.microsoft.com/fwlink/?LinkId=532495). This is a direct link which may not work in the future as your are now required to create a Microsoft account to download old versions of Visual Studio. **Do not** install Visual Studio 2015 as latest versions are not compatible with CUDA 8. Open VS2013 to see if it’s working.
2. Add  C:\Program Files (x86)\Microsoft Visual Studio 12.0\VC\bin to the PATH.
3. Download and install [CUDA 8.0](https://developer.nvidia.com/cuda-toolkit), you have to create a free NVidia account to download the file. Open the samples directory C:\ProgramData\NVIDIA Corporation\CUDA Samples\v8.0  and then the Samples\_vs2013.sln file. Run any project to see if CUDA is working, for instance, 5\_Simulations/oceanFFT. Right click on it, then click on Debug>Start new instance. Accept to build the project, then you should see a simulated ocean surface.
4. The CUDA installation may have downgraded your NVidia drivers, you can [upgrade](http://www.nvidia.fr/Download/index.aspx) them again.

# Installing Theano

1. Download and install [Anaconda 64 bits with Python 3.5 for Windows](https://www.continuum.io/downloads). Open a command line and check that Python is working and that it is indeed the Anaconda python 3.5 version.
2. Download and extract MinGW [x86\_64-5.4.0-release-posix-seh-rt\_v5-rev0.7z](https://sourceforge.net/projects/mingw-w64/files/Toolchains%20targetting%20Win64/Personal%20Builds/mingw-builds/5.4.0/threads-posix/seh/x86_64-5.4.0-release-posix-seh-rt_v5-rev0.7z/download). Make sure you’re not using a newer version as it won’t work with Theano.
3. Add MinGW bin directory to the PATH. For instance, if you extracted MinGW onC: , then add  C:\x86\_64-5.4.0-release-posix-seh-rt\_v5-rev0\mingw64\bin  to the PATH.
4. Open a **new** command prompt (to take into account the modification to the PATH) and type:



|  |  |
| --- | --- |
| 1  2  3  4 | cd %USERPROFILE%\Anaconda3\libs  gendef ..\python35.dll  dlltool --as-flags=--64 -m i386:x86-64 -k --output-lib libpython35.a --input-def python35.def  del python35.def |

1. Download and install [Git for Windows](https://git-for-windows.github.io/).
2. In the command prompt type:



|  |  |
| --- | --- |
| 1  2  3  4  5  6 | cd %USERPROFILE%\Anaconda3\Lib\site-packages  git clone git://github.com/Theano/Theano.git  cd Theano  python setup.py develop  cd %USERPROFILE%  type NUL > .theanorc |

1. Open the explorer and type in the address bar %USERPROFILE%, open the .theanorc file and add the following lines:



|  |  |
| --- | --- |
| 1  2  3  4  5  6 | [global]  floatX = float32  device = gpu    [nvcc]  compiler\_bindir=C:\Program Files (x86)\Microsoft Visual Studio 12.0\VC\bin |

I assume you installed CUDA, if it is not the case, you have to replace  device =gpu   by  device = cpu  . If you have more than one GPU, you can also specify which one you want to use using gpu1 or gpu2  etc.

1. If your Windows username contains a space or any unusual character or if you want Theano to put its temporary files to a specific place (they can use several Gb of disk space) then add  base\_compiledir=C:\theano\_compiledir  under the [global]  section of your.theanorc file. Your .theanorc file should now contain:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | [global]  floatX = float32  device = gpu  base\_compiledir=C:\theano\_compiledir    [nvcc]  compiler\_bindir=C:\Program Files (x86)\Microsoft Visual Studio 12.0\VC\bin |

1. Now it’s time to test if Theano is working properly. Copy the following code into a file and run it in the command line.



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22 | from theano import function, config, shared, sandbox  import theano.tensor as T  import numpy  import time    vlen = 10 \* 30 \* 768  # 10 x #cores x # threads per core  iters = 1000    rng = numpy.random.RandomState(22)  x = shared(numpy.asarray(rng.rand(vlen), config.floatX))  f = function([], T.exp(x))  print(f.maker.fgraph.toposort())  t0 = time.time()  for i in range(iters):      r = f()  t1 = time.time()  print("Looping %d times took %f seconds" % (iters, t1 - t0))  print("Result is %s" % (r,))  if numpy.any([isinstance(x.op, T.Elemwise) for x in f.maker.fgraph.toposort()]):      print('Used the cpu')  else:      print('Used the gpu') |

If everything’s OK, this code will display something like that:



|  |  |
| --- | --- |
| 1  2  3  4  5  6 | Using gpu device 0: GeForce GTX 960 (CNMeM is disabled, CuDNN not available)  [GpuElemwise{exp,no\_inplace}(<CudaNdarrayType(float32, vector)>), HostFromGpu(GpuElemwise{exp,no\_inplace}.0)]  Looping 1000 times took 0.5765759944915771 seconds  Result is [ 1.23178029 1.61879349 1.52278066 ..., 2.20771813 2.29967761  1.62323296]  Used the gpu |

# Installing Keras

1. In a command prompt, type:



|  |  |
| --- | --- |
| 1  2  3  4 | cd "%USERPROFILE%\Anaconda3\Lib\site-packages"  git clone git://github.com/fchollet/keras.git  cd keras  python setup.py develop |

1. As Keras uses TensorFlow as the default backend, we have to change that for Theano. Open a command line and type:



|  |  |
| --- | --- |
| 1  2 | cd %USERPROFILE%  md .keras |

1. Create a file named  keras.json  in  %USERPROFILE%/.keras  and add the following lines:



|  |  |
| --- | --- |
| 1  2  3  4  5  6 | {      "floatx": "float32",      "epsilon": 1e-07,      "image\_dim\_ordering": "th",      "backend": "theano"  } |

1. Now it’s time to test if Keras is working properly. Copy the following code into a file and run it in the command line.



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39 | from keras.models import Sequential  from keras.layers import Dense, Dropout, Activation  from keras.optimizers import SGD  import numpy as np    data\_dim = 20  nb\_classes = 4    model = Sequential()    # Dense(64) is a fully-connected layer with 64 hidden units.  # in the first layer, you must specify the expected input data shape:  # here, 20-dimensional vectors.  model.add(Dense(64, input\_dim=data\_dim, init='uniform'))  model.add(Activation('tanh'))  model.add(Dropout(0.5))  model.add(Dense(64, init='uniform'))  model.add(Activation('tanh'))  model.add(Dropout(0.5))  model.add(Dense(nb\_classes, init='uniform'))  model.add(Activation('softmax'))    model.compile(loss='categorical\_crossentropy',                optimizer='sgd',                metrics=["accuracy"])    # generate dummy training data  x\_train = np.random.random((1000, data\_dim))  y\_train = np.random.random((1000, nb\_classes))    # generate dummy test data  x\_test = np.random.random((100, data\_dim))  y\_test = np.random.random((100, nb\_classes))    model.fit(x\_train, y\_train,            nb\_epoch=20,            batch\_size=16)    score = model.evaluate(x\_test, y\_test, batch\_size=16) |

If everything is working properly you should see (it can take several minutes on the first run):



|  |  |
| --- | --- |
| 1  2  3  4 | Using Theano backend.  Using gpu device 0: GeForce GTX 960 (CNMeM is disabled, CuDNN not available)    [Lots of Epoch N/20 with loss and accuracy measures] |

1. To be able to draw graph representation of models, there are two more dependencies to install. In the command prompt, type:



|  |  |
| --- | --- |
| 1  2 | conda install graphviz  pip install git+https://github.com/nlhepler/pydot.git |

1. Pydot may have trouble finding the path to GraphViz. Open the file  \_\_init\_\_.py  in  %USERPROFILE%\Anaconda3\Lib\site-packages\pydot and add



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | elif os.path.exists(os.path.join(path, prg + '.bat')):                  if was\_quoted:                      progs[prg] = '"' + os.path.join(path, prg + '.bat') + '"'                  else:                      progs[prg] = os.path.join(path, prg + '.bat')                    success = True |

just after



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | elif os.path.exists(os.path.join(path, prg + '.exe')):                  if was\_quoted:                      progs[prg] = '"' + os.path.join(path, prg + '.exe') + '"'                  else:                      progs[prg] = os.path.join(path, prg + '.exe')                    success = True |

# Troubleshooting

If you encounter any error message during the installation and tests, check that your PATH is clean. Open a **new** command prompt and type:

1. This command should return the path where you installed Anaconda:



|  |  |
| --- | --- |
| 1 | where python |

1. All these command should return the path where you extracted MinGW:



|  |  |
| --- | --- |
| 1  2  3  4 | where gcc  where g++  where gendef  where dlltool |

1. This command should return the path of the bin folder of your VS2013 installation:



|  |  |
| --- | --- |
| 1 | where cl |

1. This command should return the path where you installed CUDA (if you use it):



|  |  |
| --- | --- |
| 1 | where nvcc |

1. Older versions of Theano were not supposed to work with Python 3.5. If you encounter any problem, you have to bypass this limitation (without any know problem) by editing the file \_\_init\_\_.py  located in%USERPROFILE%\Anaconda3\lib\site-packages\theano\theano\  and delete the lines:



|  |  |
| --- | --- |
| 1  2  3 | if sys.platform == 'win32' and sys.version\_info[0:2] == (3, 5):      raise RuntimeError(          "Theano do not support Python 3.5 on Windows. Use Python 2.7 or 3.4.") |

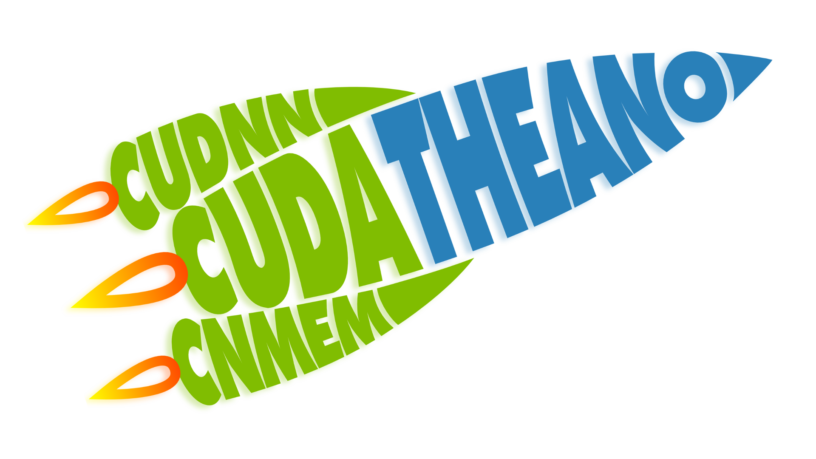
# Conclusion

Now you should be ready to do amazing stuff with Keras and Theano. If you want to make your networks run faster, you should check my article on [how to use CuDNN and CNMeM with Theano](http://ankivil.com/making-theano-faster-with-cudnn-and-cnmem-on-windows-10/).

<http://ankivil.com/making-theano-faster-with-cudnn-and-cnmem-on-windows-10/>

# [Making Theano Faster with CuDNN and CNMeM on Windows 10](http://ankivil.com/making-theano-faster-with-cudnn-and-cnmem-on-windows-10/)

**Posted by**[**Fabien Tencé**](http://ankivil.com/author/fabientence/)**on**[**2 August 2016**](http://ankivil.com/making-theano-faster-with-cudnn-and-cnmem-on-windows-10/)[**12 Comments**](http://ankivil.com/making-theano-faster-with-cudnn-and-cnmem-on-windows-10/#comments)

[](http://ankivil.com/making-theano-faster-with-cudnn-and-cnmem-on-windows-10/)

**READ NEXT →**

[[](http://ankivil.com/installing-keras-theano-and-dependencies-on-windows-10/)](http://ankivil.com/installing-keras-theano-and-dependencies-on-windows-10/" \o "Installing Keras, Theano and Dependencies on Windows 10)

**[SOFTWARE](http://ankivil.com/installing-keras-theano-and-dependencies-on-windows-10/" \o "Installing Keras, Theano and Dependencies on Windows 10)**

### [Installing Keras, Theano and Dependencies on Windows 10](http://ankivil.com/installing-keras-theano-and-dependencies-on-windows-10/" \o "Installing Keras, Theano and Dependencies on Windows 10)

[[](http://ankivil.com/installing-keras-theano-and-dependencies-on-windows-10-python-34/)](http://ankivil.com/installing-keras-theano-and-dependencies-on-windows-10-python-34/" \o "Installing Keras, Theano and Dependencies on Windows 10 – Old way with Python 3.4)

**[SOFTWARE](http://ankivil.com/installing-keras-theano-and-dependencies-on-windows-10-python-34/" \o "Installing Keras, Theano and Dependencies on Windows 10 – Old way with Python 3.4)**

### [Installing Keras, Theano and Dependencies on Windows 10 – Old way with Python 3.4](http://ankivil.com/installing-keras-theano-and-dependencies-on-windows-10-python-34/" \o "Installing Keras, Theano and Dependencies on Windows 10 – Old way with Python 3.4)

[[](http://ankivil.com/choosing-a-deep-learning-software/)](http://ankivil.com/choosing-a-deep-learning-software/" \o "Choosing a Deep Learning Software)

**[SOFTWARE](http://ankivil.com/choosing-a-deep-learning-software/" \o "Choosing a Deep Learning Software)**

### [Choosing a Deep Learning Software](http://ankivil.com/choosing-a-deep-learning-software/" \o "Choosing a Deep Learning Software)

# Introduction

After installing Theano and CUDA (see [my previous article](http://ankivil.com/installing-keras-theano-and-dependencies-on-windows-10/)) you can tweak your configuration to substantially increase the speed of your networks. There are no or few drawbacks, the only condition is to use Theano with CUDA. This article specifically addresses the problem of having the following message each time you load Theano: (CNMeM is disabled, CuDNN not available).

# Installing CuDNN

CuDNN is a library for CUDA, developed by NVIDIA, which provides highly tuned implementations of primitives for deep neural networks. CuDNN is said to make deep nets run faster and sometimes using less memory.

Here are the step to make Theano use CuDNN:

1. Register at [NVIDIA cuDNN](https://developer.nvidia.com/cudnn).
2. Download [cuDNN v5 Library for Windows 10](https://developer.nvidia.com/rdp/assets/cudnn-7.5-windows10-x64-v5.0-rc-zip).
3. Extract the archive and copy the 3 directories ( bin , include and lib ) inC:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v7.5  or  C:\Program Files\NVIDIA GPU Computing Toolkit\CUDA\v8.0 . If you’re unsure which version of CUDA Theano uses, check your  CUDA\_PATH  environment variable.
4. Open a python console and enter import theano . You should see the message(CNMeM is disabled, CuDNN 5004) . If you have errors, try upgrading Theano using the bleeding-edge version (see [here](http://ankivil.com/installing-keras-theano-and-dependencies-on-windows-10/)), especially if you have the messageUserWarning: Your CuDNN version is more recent then Theano. If you see problems, try updating Theano or downgrading CuDNN to version 4. You may also check that your .theanorc  file does **not** contain:



|  |  |
| --- | --- |
| 1  2 | [dnn]  enabled = False |

## About The .theanorc File

The .theanorc  file should be located in your user profile. As you cannot create a file beginning with a dot in the file explorer, you have to open a terminal and type:



|  |  |
| --- | --- |
| 1  2 | cd %USERPROFILE%  type NUL > .theanorc |

Then you can open the .theanorc  file and add the following lines:



|  |  |
| --- | --- |
| 1  2  3 | [global]  floatX = float32  device = gpu |

Depending on your installation setup, you may also have to add:



|  |  |
| --- | --- |
| 1  2  3 | [nvcc]  flags=-L"C:\Program=Files\Anaconda\libs"  compiler\_bindir="C:\Program Files (x86)\Microsoft Visual Studio 12.0\VC\bin" |

You have to adjust the paths to where you have installed Anaconda and CUDA.  Be careful with the nvcc flags as they do not handle spaces, you have to replace them with = .

# Using CNMeM

CNMeM is a library, developed by NVIDIA, which helps deep learning frameworks managing CUDA memory. CNMeM is already integrated in Theano so you don’t have to install anything. To enable CNMeM in Theano, you have to add to the.theanorc  file the lines:



|  |  |
| --- | --- |
| 1  2 | [lib]  cnmem = 0.8 |

The cnmem value specifies the amount of GPU memory allocated for Theano. To quote [the documentation](http://deeplearning.net/software/theano/library/config.html#config.config.lib.cnmem):

* 0: not enabled.
* 0 < N <= 1: use this fraction of the total GPU memory (clipped to .95 for driver memory). [Note: This should be a float value, for instance 0.25 or 1.0]
* > 1: use this number in megabytes (MB) of memory.

In theory, for cards dedicated to deep learning you should put the value to 1.0. For cards used also to render to the screen you should put a value around 0.8, it depends what applications you use while Theano runs (internet browser, etc.) and the size of your GPU memory.

In practice, using a graphics card with 4Go of memory, I couldn’t go over 3395Mo exactly which is approximately 83% of the memory. Whether the graphics card was dedicated to deep learning or plugged to a monitor did not change this value.  Of course, all the software I used to monitor the GPU memory usage (GPU-Z and nvidia-smi) reported that several hundreds of MB of memory were left free so it must be a bug with CNMeM, Theano, my card or the combination of all three.

**Be careful!** If you enable CNMeM, you should put the highest possible value as low values could result in memory fragmentation, see the “Performance Gains” section.

To test is Theano uses CNMeM, open a Python interpreter and type  import theano . You should see  (CNMeM is enabled with initial size: 80.0% of memory, CuDNN 5004)  if everything’s OK.

# About Installing OpenBLAS

OpenBLAS is an optimized Basic Linear Algebra Subprograms library. Usually, Theano uses the default BLAS library through Numpy. It is also possible to link directly Theano to one of the fastest BLAS library: OpenBLAS.

Doing so makes Theano run significantly faster **on CPU**, it does not change the performances if you run Theano on GPU from what I measured. If you still want to give it a try, here are the steps:

* Download [mingw64\_dll.zip](http://sourceforge.net/projects/openblas/files/v0.2.14/mingw64_dll.zip/download)
* Download [OpenBLAS-v0.2.14-Win64-int32.zip](http://sourceforge.net/projects/openblas/files/v0.2.14/OpenBLAS-v0.2.14-Win64-int32.zip/download)
* Create a directory to put OpenBlas,  C:\openblas for instance
* Copy the DLLs from mingw64\_dll  and OpenBlas/bin  into C:\openblas
* Add to following lines to your  .theanorc file:



|  |  |
| --- | --- |
| 1  2 | [blas]  ldflags=-LC:\openblas -lopenblas |

* Add C:\OpenBlas to the  PATH environment variable

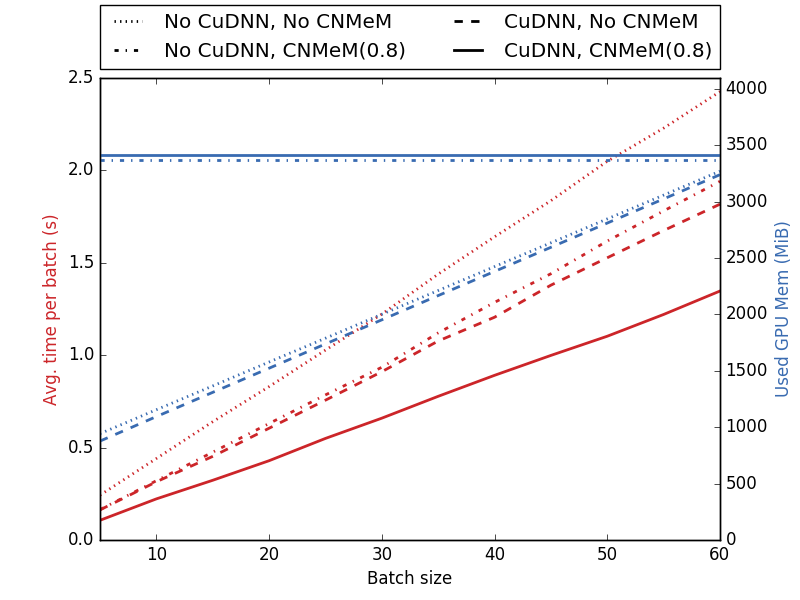
You can test if there is any gain in speed using  the check\_blas.py  in your Theano installation, under the Theano/misc directory. It indeed improved performance on the CPU, in my case, but it didn’t change a thing while running on the GPU (which is a **lot** faster).

# Performance Gains

To evaluate how much the performance is improved by CuDNN and CNMeM, I ran several tests. The protocol is the following: time and memory usage is monitored for a single forward pass using random samples and increasing batch size.

## VGG16

Here are the results for the [VGG16](https://gist.github.com/ksimonyan/211839e770f7b538e2d8#file-readme-md) network:

[](http://ankivil.com/wp-content/uploads/2016/08/cuddn_cnmem_combined_graph_vgg16.png)

### Speed

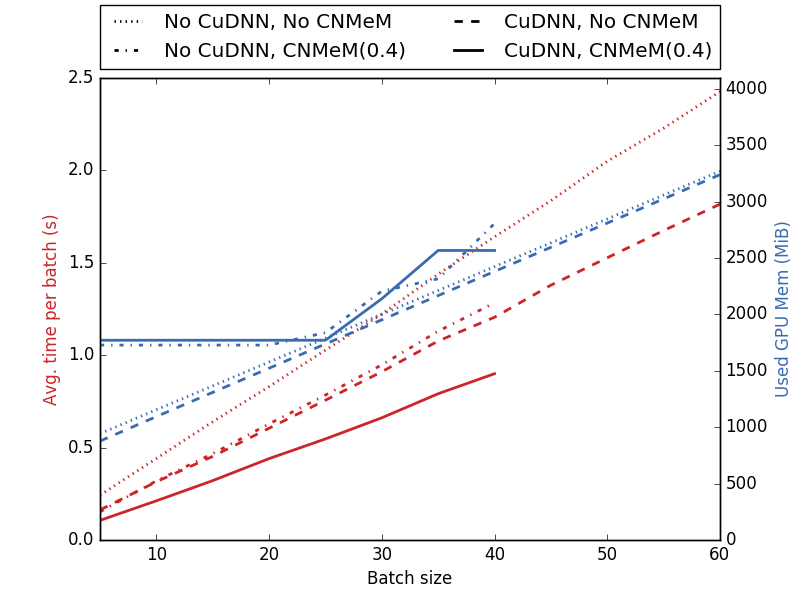
Both CuDNN and CnMeM improves significantly the speed of the network with CuDNN giving the highest speed boost. To achieve maximum performance, both CuDNN and CNMeM should be enabled.

### Memory Usage

Using CuDNN does not seems to impact the memory usage. However, using CNMeM makes Theano use a constant amount of memory, no matter how small the batch size is. If you really want to optimize the amount of memory dedicated to CNMeM, you should adjust it depending on the size of the batch you are feeding to your network.

### Memory Fragmentation

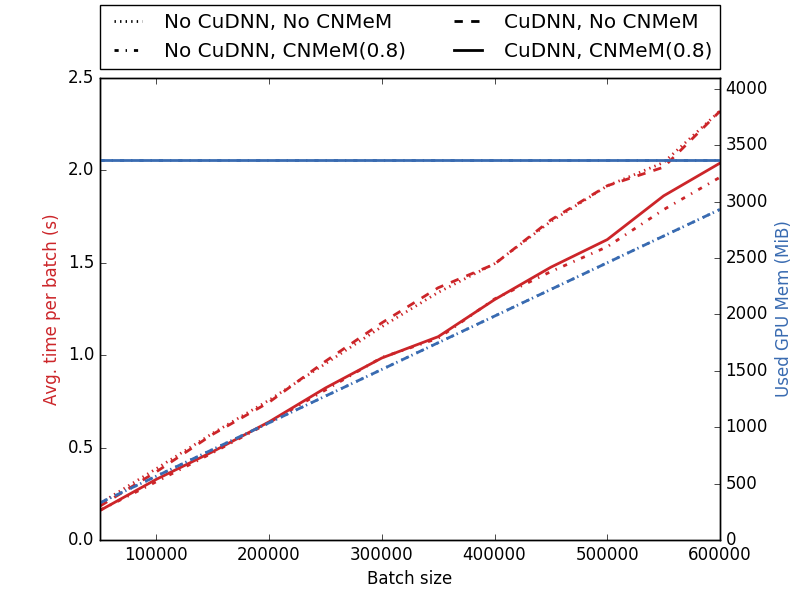
The Theano documentation warns of memory fragmentation if using low values for CNMeM. I tested this behavior using a CNMeM value of 0.4:

[](http://ankivil.com/wp-content/uploads/2016/08/cuddn_cnmem_combined_graph_vgg16_frag.png)

The lines for CNMeM(0.4) are truncated because for batch size over 40, Theano crashes because it needs too much memory. We can see that when CNMeM runs out of memory and there is enough available on the graphics card, the memory usage goes up. However, it increases faster than without CNMeM which may indicate that memory fragmentation happens. In this case, CNMeM is using much more memory when reallocating memory than when CNMeM has a big enough memory pool from the beginning.

## MLP Gain

The last test is to see if CuDNN and CNMeM can improve the performance on very simple networks only composed of dense layers with ReLU and softmax activation functions:

[](http://ankivil.com/wp-content/uploads/2016/08/cuddn_cnmem_combined_graph_mlp.png)

In this case, the CuDNN library does not improve the speed but the CNMeM library does. So, while CuDNN provides the higher speed boost in convnets, CNMeM seems to increase the performance for most of the networks. Regarding the memory usage, the behavior is similar to the one for the VGG16 network.

# Other Optimizations

There are other tweaks to make Theano run faster (see [here](http://deeplearning.net/software/theano/faq.html#theano-memory-speed-trade-off)). I did not notice any gain similar in magnitude to CuDNN and CNMeM, so I advice you to stick to these two.

# Conclusion

Using Theano on the GPU with CUDA makes it run a lot faster than on the CPU. In this article, I showed that it is also possible to increase significantly the performance making Theano and CUDA  use CuDNN and CNMeM. Convnets seems to be the most impacted by this increase in performance but classic neural networks can be sped up too.

Enabling CuDNN does not seems to have any drawback. Regarding CNMeM you should try to enable it with maximum memory. Exceptions to this should be when you want to run several Theano sessions in parallel or if your graphics card is plugged to a monitor and you use graphics-heavy applications.

### **<http://computerscienceunveiled.blogspot.ru/2015/08/installing-openblas-for-theano-on.html>**

### **Installing OpenBLAS for Theano on Windows**

Once more, this proved to be quite difficult to do on Windows, so I'll explain how I finally got it to work. I'll assume you've read the previous post about the Theano installation and that you have mingw (64 bits) installed. The order in which you do these steps shouldn't really matter, but just in case, you can follow my order.  
  
First, you need to [download Open BLAS](http://sourceforge.net/projects/openblas/files/v0.2.14/). Get the one called:  
  
**OpenBLAS-v0.2.14-Win64-int32.zip**(I'm not sure if this also works with the -int64 version)  
  
Next, make a folder **C:\openblas**. I don't think it must have this exact name, but **it must not contain any spaces**, so keep it simple.  
  
Now, add this to your **.theanorc** file

[?](http://computerscienceunveiled.blogspot.ru/2015/08/installing-openblas-for-theano-on.html)

|  |  |
| --- | --- |
| 1  2 | [blas]  ldflags = -LC:\openblas -lopenblas |

Make sure you type it exactly like that, with the **-L** and everything.  
  
Next, from the **OpenBLAS**zip you downloaded, extract the **libopenblas.dll** file from the **bin** folder and the **libopenblas.a** file from the **lib** folder to the **openblas** folder you created. I'm not sure if you need both or not, the dll might be enough, but I just copied both.  
  
Finally, you also need to make the dependencies of this library available. I'm sure there's a better way to do this, but what I did was copy everything from **C:\Program Files\mingw-w64\x86\_64-5.1.0-posix-seh-rt\_v4-rev0\mingw64\x86\_64-w64-mingw32\lib**(where I installed my mingw 64 bits) to the **openblas** folder I created. I guess you could also add this to your system Path, but that's arguably even uglier.  
  
This did it for me. Please comment if you have any questions or suggestions on how to simplify this process.

<http://www.asozykin.ru/deep_learning/2016/12/25/Keras-Installation.html>

Установка Keras в Anaconda

Для выполнения практических заданий по курсу [“Программирование глубоких нейронных сетей на Python”](http://www.asozykin.ru/courses/nnpython) необходимо установить библиотеку [Keras](https://keras.io/), а также один из вычислительных бекендов для этой библиотеки - [Theano](http://deeplearning.net/software/theano/) или [TensorFlow](https://www.tensorflow.org/). Самый простой способ это сделать - установить диструбутив Python Anaconda и после этого установить все необходимые пакеты с помощью conda.

1. **Установка Anaconda**. Сначала необходимо установить диструбутив Python Anaconda. Скачайте с [сайта Continuum Analytics](https://www.continuum.io/downloads) версию Anaconda для своей операционной системы. Выбирайте **третью** версию Python, потому что именно она используется в курсе.
2. **Установка Theano**. Все примеры в курсе протестированы с библиотекой Theano. Чтобы установить Theano в Anaconda, выполните команду:
3. conda install theano
4. **Установка Keras**. Пока Keras не входит в основной набор пакетов Anaconda, но его можно установить из conda-forge:
5. conda install -c conda-forge keras
6. **Настраиваем Keras на работу с Theano**. В файле .keras/keras.json, который находится в домашнем каталоге пользователя, прописываем Theano в качестве бекенда:
7. {
8. "epsilon": 1e-07,
9. "backend": "theano",
10. "image\_dim\_ordering": "th",
11. "floatx": "float32"
12. }

Также указываем, что будем использовать порядок хранения измерений в изображениях, который применяется в Theano ("image\_dim\_ordering": "th"). В TensorFlow использиуется другой порядок.

1. **Проверка установки**. Напечатаем версию Keras, которая была установлена:
2. python -c "import keras; print(keras.\_\_version\_\_)"

Результат должен быть примерно таким:

Using Theano backend.

1.0.7

Установлена версия Keras 1.0.7, в качестве бекенда используется Theano.

На этом установка закончена, можно запускать [примеры кода из курса](https://github.com/sozykin/dlpython_course).

И еще, если при импорте keras выходит   
WARNING (theano.configdefaults): g++ not detected ! Theano will be  
unable to execute optimized C-implementations (for both CPU and GPU) and  
will default to Python implementations. Performance will be severely  
degraded. To remove this warning, set Theano flags cxx to an empty string.  
то нужно до установить

conda install mingw libpython

<https://uoa-eresearch.github.io/eresearch-cookbook/recipe/2014/11/20/conda/>

# **Create virtual environments for python with conda**

Nov 20, 2014

### **How to set up a virtual environments using conda for the Anaconda Python distribution**

A virtual environment is a named, isolated, working copy of Python that that maintains its own files, directories, and paths so that you can work with specific versions of libraries or Python itself without affecting other Python projects. Virtual environmets make it easy to cleanly separate different projects and avoid problems with different dependencies and version requiremetns across components. The*conda*command is the preferred interface for managing intstallations and virtual environments with the[*Anaconda*](https://store.continuum.io/cshop/anaconda/)Python distribution. If you have a vanilla Python installation or other Python distribution see[*virtualenv*](http://virtualenv.readthedocs.org/en/latest/)

## **Outline**

* Check conda is installed and available
* Update conda if necessary
* Create a virtual environment
* Activate a virtual environment
* Install additional python packages
* Deactivate a virtual environment
* Delete a virtual environment

## **Jargon**

link to PATH,

## **Requirements**

* Anaconda Python distribution installed and accessible

## **1. Check conda is installed and in your PATH**

1. Open a terminal client.
2. Enter conda -V into the terminal command line and press enter.
3. If conda is installed you should see somehting like the following.

$ conda -V

conda 3.7.0

## **2. Check conda is up to date**

1. In the terminal client enter

conda update conda

1. Upadate any packages if necessary by typing y to proceed.

## **3. Create a virtual environment for your project**

1. In the terminal client enter the following where yourenvname is the name you want to call your environment, and replace x.x with the Python version you wish to use. (To see a list of available python versions first, type conda search "^python$" and press enter.)

conda create -n yourenvname python**=**x.x anaconda

1. Press y to proceed. This will install the Python version and all the associated anaconda packaged libraries at “path\_to\_your\_anaconda\_location/anaconda/envs/yourenvname”

## **4. Activate your virtual environment.**

1. To activate or switch into your virtual environment, simply type the following where yourenvname is the name you gave to your environement at creation.

source activate yourenvname

1. Activating a conda environment modifies the PATH and shell variables to point to the specific isolated Python set-up you created. The command prompt will change to indicate which conda environemnt you are currently in by prepending (yourenvname). To see a list of all your environments, use the command conda info -e.

## **5. Install additional Python packages to a virtual environment.**

1. To install additional packages only to your virtual environment, enter the following command where yourenvname is the name of your environemnt, and [package] is the name of the package you wish to install. Failure to specify “-n yourenvname” will install the package to the root Python installation.

conda install -n yourenvname **[**package**]**

## **6. Deactivate your virtual environment.**

1. To end a session in the current environment, enter the following. There is no need to specify the envname - which ever is currently active will be deactivated, and the PATH and shell variables will be returned to normal.

source deactivate

## **6. Delete a no longer needed virtual environment**

1. To delete a conda environment, enter the following, where yourenvname is the name of the environment you wish to delete.

conda remove -n yourenvname -all

## **Related info**

The conda offical documentation can be found [here](http://conda.pydata.org/docs/intro.html).