Installing and compiling the library

In order to install the library, follows these steps:

1. Download and unpack the library source code into a directory of your choice. Call the path to this directory `<MatConvNet>`.
2. Compile the library.
3. Start MATLAB and type:

   ```
   > run <MatConvNet>/matlab/vl_setupnn
   ```

   in order to add MatConvNet to MATLAB's search path.

At this point the library is ready to use. You can test it by using the command (using MATLAB R2014a or later):

```
> vl_testnn
```

To test GPU support (if you have compiled it) use instead:

```
> vl_testnn('gpu', true)
```

Note that the second tests runs slower than the CPU version; do not worry, this is an artefact of the test procedure.

Compiling

MatConvNet compiles under Linux, Mac, and Windows. This page discusses compiling MatConvNet using the MATLAB function `vl_compilenn` (../mfiles/vl_compilenn). While this is the easiest method, the command line or an IDE can be used as well (../install-alt/).

Compiling for CPU

If this is the first time you compile MatConvNet, consider trying first the CPU version. In order to do this, use the `vl_compilenn` (../mfiles/vl_compilenn) command supplied with the library:

1. Make sure that MATLAB is configured to use your compiler (http://www.mathworks.com/help/matlab/matlab_external/changing-default-compiler.html).
2. Open MATLAB and issue the commands:

   ```
   > cd <MatConvNet>
   > addpath matlab
   > vl_compilenn
   ```

At this point MatConvNet should start compiling. If all goes well, you are ready to use the library. If not, you can try debugging the problem by running the compilation script again in verbose mode:

```
> vl_compilenn('verbose', 1)
```

Increase the verbosity level to 2 to get even more information.

> **Remark:** The 'vl_imreadjpeg' tool uses an external image library to load images. In Mac OS X and Windows, the default is to use the system libraries (Quartz and GDI+ respectively), so this dependency is immaterial. In Linux, this tool requires the LibJPEG library and the corresponding development files to be installed in the system. If needed, the `ImageLibraryCompileFlags` and `ImageLibraryLinkFlags` options can be used to adjust the compiler and linker flags to match a specific library installation. It is also possible to use the `EnableImreadJpeg` option of `vl_compilenn` to turn off this feature.

Compiling the GPU support

To use the the GPU-accelerated version of the library, you will need a NVIDA GPU card with compute capability 2.0 or greater and a copy of the NVIDIA CUDA toolkit. Ideally, the version of the CUDA toolkit should match your MATLAB version:

MATLAB	CUDA toolkit
R2013b	5.5
R2014a	5.5
R2014b	6.0
R2015a	6.5
R2015b	7.0

You can also use the `gpuDevice` MATLAB command to find out MATLAB's version of the CUDA toolkit. It is also possible (and often necessary) to use a more recent version of CUDA than the one officially supported by MATLAB; this is explained later.

Assuming that there is only a single copy of the CUDA toolkit installed in your system and that it matches MATLAB's version, compile the library with:

```
> vl_compilenn('enableGpu', true)
```

If you have multiple versions of the CUDA toolkit, or if the script cannot find the toolkit for any reason, specify the path to the CUDA toolkit explicitly. For example, on a Mac this may look like:

```
> vl_compilenn('enableGpu', true, 'cudaRoot', '/Developer/NVIDI
A/CUDA-7.0')
```

Once more, you can use the `verbose` option to obtain more information if needed.

Using an unsupported CUDA toolkit version

MatConvNet can be compiled to use a more recent version of the CUDA toolkit than the one officially supported by MATLAB. While this may cause unforeseen issues (although none is known so far), it is necessary to use recent libraries such as cuDNN.

Compiling with a newer version of CUDA requires using the `cudaMethod,nvcc` option. For example, on a Mac this may look like:

```
> vl_compilenn('enableGpu', true, ...
               'cudaRoot', '/Developer/NVIDIA/CUDA-7.0', ...
               'cudaMethod', 'nvcc')
```

Note that at this point MatConvNet MEX files are linked *against the specified CUDA libraries* instead of the one distributed with MATLAB. Hence, in order to use MatConvNet it is now necessary to allow MATLAB accessing these libraries. On Linux one way to do so is to start MATLAB from the command line (terminal) specifying the `LD_LIBRARY_PATH` option. For instance, on Linux this may look like:

```
$ LD_LIBRARY_PATH=/Developer/NVIDIA/CUDA-7.0/lib64 matlab
```

On Windows, chances are that the CUDA libraries are already visible to MATLAB so that nothing else needs to be done.

On Mac, this step should not be necessary as the library paths are hardcoded during compilation.

Compiling the cuDNN support

MatConvNet supports the NVIDIA cuDNN library (https://developer.nvidia.com/cuDNN) for deep learning (and in particular their fast convolution code). In order to use it, obtain the cuDNN (http://devblogs.nvidia.com/parallelforall/accelerate-machine-learning-cudnn-deep-neural-network-library) library from NVIDIA (cuDNN v2 to v4 should work; however, later version are *strongly recommended* as earlier version had a few bugs). Make sure that the CUDA toolkit matches the one in cuDNN (e.g. 6.5). This often means that the CUDA toolkit will *not* match the one used internally by MATLAB, such that the compilation method discussed above must be used.

Unpack the cuDNN library binaries and header files in a place `<Cudnn>` of you choice. In the rest of this example, it will be assumed that this cuDNN RC4 has been unpacked in `local/cudnn-rc4` in the `<MatConvNet>` root directory, (i.e. `<Cudnn> = <MatConvNet>/local/cudnn-rc4`). For example, the directory structure on a Mac should look like:

```
COPYING
Makefile
Makefile.mex
...
examples/
local/
  cudnn-rc4/
    include/
      cudnn.h
    lib/
      libcudnn.7.5.dylib
      libcudnn.dylib
    ...
```

Use `vl_compilenn` with the `cudnnEnable,true` option to compile the library; do not forget to use `cudaMethod,nvcc` as, at it is likely, the CUDA toolkit version is newer than MATLAB's CUDA toolkit. For example, on Mac this may look like:

```
> vl_compilenn('enableGpu', true, ...
               'cudaRoot', '/Developer/NVIDIA/CUDA-7.5', ...
               'cudaMethod', 'nvcc', ...
               'enableCudnn', 'true', ...
               'cudnnRoot', 'local/cudnn-rc4') ;
```

MatConvNet is now compiled with cuDNN support. When starting MATLAB, however, do not forget to point it to the paths of both the CUDA and cuDNN libraries. On a Linux terminal, this may look like:

```
$ cd <MatConvNet>
$ LD_LIBRARY_PATH=/Developer/NVIDIA/CUDA-7.5/lib64:local matlab
```

On Windows, copy the cuDNN DLL file `<Cudnn>/cudnn*dll` (or from wherever you unpacked cuDNN) into the `<MatConvNet>/matlab/mex` directory.

On Mac, this step should not be necessary as the library paths are hardcoded during compilation.

Further examples

To compile all the features in MatConvNet on a Mac and MATLAB 2014b, CUDA toolkit 6.5 and cuDNN Release Candidate 2, use:

```
> vl_compilenn('enableGpu', true, 'cudaMethod', 'nvcc', ...
               'cudaRoot', '/Developer/NVIDIA/CUDA-6.5', ...
               'enableCudnn', true, 'cudnnRoot', 'local/cudnn-rc
2') ;
```

The equivalent command on Ubuntu Linux would look like:

```
> vl_compilenn('enableGpu', true, 'cudaMethod', 'nvcc', ...
               'cudaRoot', '/opt/local/cuda-6.5', ...
               'enableCudnn', true, 'cudnnRoot', 'local/cudnn-rc
2') ;
```

Using MATLAB 2015b, CUDA 7.5, and cuDNN R4:

```
> vl_compilenn('enableGpu', true, ...
               'cudaMethod', 'nvcc', ...
               'cudaRoot', '/opt/local/cuda-7.5', ...
               'enableCudnn', true, ...
               'cudnnRoot', 'local/cudnn-rc4') ;
```