About MatConvNet

MatConvNet was born in the Oxford Visual Geometry Group as both an educational and research platform for fast prototyping in Convolutional Neural Nets. Its main features are:

- *Flexibility.* Neural network layers are implemented in a straightforward manner, often directly in MATLAB code, so that they are easy to modify, extend, or integrate with new ones. Other toolboxes hide the neural network layers behind a wall of compiled code; here the granularity is much finer.
- *Power.* The implementation can run large models such as Krizhevsky et al., including the DeCAF and Caffe variants. Several pre-trained models are provided.
- *Efficiency.* The implementation is quite efficient, supporting both CPU and GPU computation.

This library may be merged in the future with VLFeat library (http://www.vlfeat.org/). It uses a very similar style, so if you are familiar with VLFeat, you should be right at home here.

Changes

- 1.0-beta17 (December 2015).

  New features

  - Mac OS X 10.11 support. Since setting `LD_LIBRARY_PATH` is not supported under this OS due to security reasons, now MatConvNet binaries hardcodes the location of the CUDA/cuDNN libraries as needed. This also simplifies starting up MATLAB.
  - This version changes slightly how cuDNN is configured; the cuDNN root directory is assumed to contain two subdirectories `lib` and `include` instead of the binary and include files directly. This matches how cuDNN is now distributed.
  - CuDNN v4 is now supported.
  - This version changes how batch normalization is handled. Now the average moments are learned together with the other parameters. The net result is that batch normalization is easy to bypass at test time (and implicitly done in validation, just like dropout).
  - The `disableDropout` parameter of `vl_simplenn` has been replaced by a more generic `mode` option that allows running in either normal mode or test mode. In the latter case, both dropout and batch normalization are bypassed. This is the same behavior of `DagNN.mode`.
  - Examples have been re-organized in subdirectories.
  - Compiles and works correctly with cuDNN v4. However, not all v4 features are used yet.

- Adds an option to specify the maximum workspace size in the convolution routines using cuDNN.
- The AlexNet, VGG-F, VGG-M, VGG-S examples provided in the `examples/imagenet` directory have been refined in order to produced deployable models. MatConvNet pretrained versions of these models are available for download.
- A new option in `vl_nnconv` and `vl_nnconvt` allows setting the maximum amount of memory used by CuDNN to perform convolution.

Changes affecting backward compatibility

- This version changes slightly how SimpleNN networks should be handled. Use the `vl_simplenn_tidy()` to upgrade existing networks to the latest version of MatConvNet. This function is also useful to fill in missing default values for the parameters of the network layers. It is therefore recommended to use `vl_simplenn_tidy()` also when new models are defined.
- The downloadable pre-trained models have been updated to match the new version of SimpleNN. The older models are still available for download. Note that old and new models are numerically equivalent, only the format is (slightly) different.
- Recent versions of CuDNN may use by default a very large amount of memory for computation.

- 1.0-beta16 (October 2015). Adds VGG-Face as a pretrained model. Bugfixes.
- 1.0-beta15 (September 2015). Supports for new `DagNN` blocks and import script for the FCN models. Improved `vl_nnbnorm`.
- 1.0-beta14 (August 2015). New `DagNN` wrapper for networks with complex topologies. GoogLeNet support. Rewritten `vl_nnloss` block with support for more loss functions. New blocks, better documentation, bugfixes, new demos.
- 1.0-beta13 (July 2015). Much faster batch normalization and several minor improvements and bugfixes.
- 1.0-beta12 (May 2015). Added `vl_nnconvt` (convolution transpose or deconvolution).
- 1.0-beta11 (April 2015) Added batch normalization, spatial normalization, sigmoid, p-distance. Extended the example training code to support multiple GPUs. Significantly improved the tuning of the ImageNet and CIFAR examples. Added the CIFAR Network in Network model.

  This version changes slightly the structure of `simplenn`. In particular, the `filters` and `biases` fields in certain layers have been replaced by a `weights` cell array containing both tensors, simplifying a significant amount of code. All examples and downloadable models have been updated to reflect this change. Models using the old structure format still work but are deprecated.

  The `cnn_train` training code example has been rewritten to support multiple GPUs. The interface is nearly the same, but the `useGpu` option has been replaced by a `gpus` list of GPUs to use.

- 1.0-beta10 (March 2015) `vl_imreadjpeg` works under Windows as well.
- 1.0-beta9 (February 2015) CuDNN support. Major rewrite of the C/CUDA core.
- 1.0-beta8 (December 2014) New website. Experimental Windows support.

- 1.0-beta7 (September 2014) Adds VGG very deep models.
- 1.0-beta6 (September 2014) Performance improvements.
- 1.0-beta5 (September 2014) Bugfixes, adds more documentation, improves ImageNet example.
- 1.0-beta4 (August 2014) Further cleanup.
- 1.0-beta3 (August 2014) Cleanup.
- 1.0-beta2 (July 2014) Adds a set of standard models.
- 1.0-beta1 (June 2014) First public release.

Contributors

MatConvNet is developed by several hands:

- Andrea Vedaldi, project coordinator
- Karel Lenc, DaG, several building blocks and examples
- Sébastien Ehrhardt, GPU implementation of batch normalization, FCN building blocks and examples
- Max Jaderberg, general improvements and bugfixes

MatConvNet quality also depends on the many people using the toolbox and providing us with feedback and bug reports.

Copyright

This package was originally created by Andrea Vedaldi (http://www.robots.ox.ac.uk/~vedaldi) and Karel Lenc and it is currently developed by a small community of contributors. It is distributed under the permissive BSD license (see also the file `COPYING`):

Acknowledgments

The implementation of the computational blocks in this library, and in particular of the convolution operators, is inspired by Caffe (http://caffe.berkeleyvision.org).

We gratefully acknowledge the support of NVIDIA Corporation with the donation of the GPUs used to develop this software.