0.12.0 Release Notes

Performance

- Added full support for NVIDIA Volta GPU Architecture and CUDA 9. Training CNNs is up to 3.5x faster than Pascal when using float16 precision.
- Enabled JIT compilation. Autograd and Gluon hybridize now use less memory and has faster speed. Performance is almost the same with old symbolic style code.
- Improved ImageRecordIO image loading performance and added indexed RecordIO support.
- Added better openmp thread management to improve CPU performance.

New Features - Gluon

- Added enhancements to the `Gluon` package, a high-level interface designed to be easy to use while keeping most of the flexibility of low level API. Gluon supports both imperative and symbolic programming, making it easy to train complex models imperatively with minimal impact on performance. Neural networks (and other machine learning models) can be defined and trained with `gluon.nn` and `gluon.rnn` packages. For Gluon tutorials, see The Straight Dope.
- `Added new loss functions - SigmoidBinaryCrossEntropyLoss, CTCLoss, HuberLoss, HingeLoss, SquaredHingeLoss, LogisticLoss, TripletLoss.`
- `gluon.Trainer` now allows reading and setting learning rate with `trainer.learning_rate` property.
- Added API `HybridBlock.export` for exporting gluon models to MXNet format.
- Added `gluon.contrib` package.
    - Convolutional recurrent network cells for RNN, LSTM and GRU.
    - `VariationalDropoutCell`

New Features - Autograd

- Added enhancements to `autograd` package, which enables automatic differentiation of NDArray operations.
- `autograd.Function` allows defining both forward and backward computation for custom operators. See documentation for examples.
- Added `mx.autograd.grad` and experimental second order gradient support (most operators don't support second order gradient yet).
- Autograd now supports cross-device graphs. Use `x.copyto(mx.gpu(i))` and `x.copyto(mx.cpu())` to do computation on multiple devices.

New Features - Sparse Tensor Support

- Added support for sparse matrices. See documentation for more info.
- Added limited cpu support for two sparse formats in `Symbol` and `NDArray` - `CSRNDArray` and `RowSparseNDArray`.
- Added a sparse dot product operator and many element-wise sparse operators.
- Added a data iterator for sparse data input - `LibSVMIter`.
- Added three optimizers for sparse gradient updates: `Ftrl`, `SGD` and `Adam`.
- Added `push` and `row_sparse_pull` with `RowSparseNDArray` in distributed kvstore.

Other New Features

- Added limited support for fancy indexing, which allows you to very quickly access and modify complicated subsets of an array's values. `x[idx_arr0, idx_arr1, ..., idx_arrn]` is now supported. Features such as combining and slicing are planned for the next release.
- Random number generators in `mx.nd.random.*` and `mx.sym.random.*` now support both CPU and GPU.
- `NDArray` and `Symbol` now supports "fluent" methods. You can now use `x.exp()` etc instead of `mx.nd.exp(x)` or `mx.sym.exp(x)`.
- Added `mx.rtc.CudaModule` for writing and running CUDA kernels from python. See documentation for examples.
- Added `multi_precision` option to optimizer for easier float16 training.
- Better support for IDE auto-completion. IDEs like PyCharm can now correctly parse mxnet operators.

API Changes

- Operators like `mx.sym.linalg_*` and `mx.sym.random_*` are now moved to `mx.sym.linalg.*` and `mx.sym.random.*`. The old names are still available but deprecated.
- `sample_*` and `random_*` are now merged as `random.*`, which supports both scalar and `NDArray` distribution parameters.

Bug-fixes

- Fixed a bug that causes `argsort` operator to fail on large tensors.
- Fixed numerical stability issues when summing large tensors.
- Fixed a bug that causes arange operator to output wrong results for large ranges.
- Improved numerical precision for unary and binary operators on float64 inputs.

Known Issues

- There are some files that need their License Headers to be updated. This is being tracked here.
- Setting OMP_NUM_THREADS to any value will disable OMP (set OMP max number of threads to one)
- There's a race condition in RowSparsePull with distributed kvstore (Fixed on master here)
- mx.ndarray.sparse.csr_matrix() uses float32 as the default dtype, instead of using the dtype of the source array (fixed in this PR)

How to build MXNet

Please follow the instructions at https://mxnet.incubator.apache.org/install/index.html

List of submodules used by Apache MXNet (Incubating) and when they were updated last

Submodule:: Last updated by MXNet:: Last update in submodule

1. cub@:: 31-Jul :: 28-Aug
2. dlpack@: 08-Sep :: 06-Oct
3. dmlc-core@: 08-Sep:: 06-Oct
4. mshadow@: 03-Oct:: 09-Oct
5. nnvm@: 10-Sep:: 10-Oct
6. ps-lite@: 28-Mar:: 27-Jul