LibNC Library

Table of Contents

1 Introduction

LibNC is a C library for tensor manipulation. It supports automatic differentiation and can be used to implement machine learning models such as LSTM and Transformers. It has the following features:

- C API.
- Small library, no external dependency, available for Linux and Windows.
- Define-by-run automatic differentiation engine (same idea as PyTorch).
- High performance for both CPU (x86) and GPU (CUDA support). Optimized support of float32 and BF16 data types.
- CPU backend optimized for inference and small batch sizes.
- Optimized for online learning (i.e. simultaneous evaluation and training) using LSTM or Transformer models.
- Fully deterministic: return the same results at each run.
- Reproducible results (CPU backend only): return the same results regardless the CPU brand and OS.

2 Tutorial

2.1 Usage

The library is provided as a DLL for Linux or Windows. It has a C API so it is easily usable from any application.

LibNC requires an x86 CPU with AVX2 support.

The CUDA support is currently only available for Linux. CUDA version 11.x must be installed. Only Ampere GPUs are currently supported.

`nctest.c` provides simple examples and auto differentiation testing code. `ncspeed.c` and `matmul_test.c` are benchmarks. `dump_coefs.c` documents the parameter dumps.

Larger programs using it are NNCP (LSTM and Transformer models for lossless text compression) and GPT2TC (GPT-2 implementation).

2.2 Main concepts

The API is defined in the `libnc.h` file.

`NCContext` represents an instance of the library. There is usually one by project. It is created with `nc_context_init`.

`NCTensor` represents a tensor (multi-dimensional array). It may be created with `nc_new_tensor`. Each tensor references a tensor buffer (`NCTensorBuffer` which contains its raw data (array of bytes). `NCTensorBuffer` reside on a compute device (e.g. CPU or GPU) represented by `NCDevice`.

`NCTensor` and `NCTensorBuffer` objects are reference counted. By default, each function consumes (=decrements) its arguments and return a live object. `const` function parameters indicate that the object is not consumed. Use `nc_free_tensor()` (resp. `nc_dup_tensor()`) to decrement (resp. increment) the reference count of a tensor.

The operands of most operations must reside on the same device. Tensors can be moved between device with `nc_tensor_to_device()`. When the tensor is on the CPU device, it is possible to have a pointer on its raw memory with `nc_tensor_get_ptr()`.

Unlike Pytorch, tensor operations don't do automatic broadcast. However, for convenience, `nc_add()` and `nc_mul()` broadcast their second argument in some common cases.

2.3 Tensor Layout

In a newly created tensor the elements are contiguous in memory. The offset of an element [a1, a0] in a tensor of shape (d1, d0) is given by (a1 * d0 + a0).

LibNC functions enumerate shapes using the smallest dimension first, i.e. `d0` first, then `d1`:

```
nc_new_tensor_2d(device, NC_TYPE_F32, d0, d1);
```

Matrices are stored in the column-first representation, e.g. the matrix:

```
[ 1 3 5 ]
[ 2 4 6 ]
```

of 2 rows and 3 columns is represented as a tensor of shape (3, 2). Its memory representation is:

```
[ 1, 2, 3, 4, 5, 6 ]
```

2.4 Automatic differentiation engine

Similarly to Pytorch, LibNC dynamically builds a computation graph. This graph is used to compute a gradient with `nc_backward()`. More precisely, Each `NCTensor` may have a reference to a `NCNode` object representing a computation graph node. Operations applied on tensors having an associated node return a tensor associated to a new node.

Newly user created tensors do not have an associated node. `nc_set_param` adds a user defined node to a tensor. It is used to create function parameters. Then `nc_backward()` calls a callback for each parameter with the calculated gradient.

Higher level APIs are normally used to create parameters such as `nc_new_param()`. LibNC provides built-in optimizers such as ADAM but the user is free to provide his own. `nc_backward()` can be used to compute higher order derivative too (Hessian vector product, see example in `nctest.c`).

3 Implementation details

3.1 Determinism

As LibNC is used for lossless data compression in NNCP, a fully deterministic behavior is required. It means the same result is returned at each run for the same computation on the same system. It is provided for both the CPU and GPU backends.

3.2 Reproductibility

The results are not modified when using a different number of threads, CPU brand or OS. Hence the code does not rely on CPU floating point instructions having a implementation defined behavior and does not use the transcendental functions of the C library.

However, in the current implementation, the CPU and GPU backends do not provide the same exact results mainly due to the use of the NVidia Tensor Core which has an undefined rounding behavior.

3.3 Compute graph optimization

LibNC does various optimizations on the compute graph such as matrix product factorisation.

Functions are provided to manually optimize the graph in the case of online learning. For this case, evaluation is done sequentially but the model parameters are still updated by training. In the training phase it is beneficial to combine all the steps of the sequential evaluation to make a better usage of the compute device parallelism. The function `nc_concat_optimization()` is employed for this purpose.

3.4 CUDA backend

NVIDIA CUDA support is optional and fully contained in the `libnc_cuda` DLL. This DLL depends on the CUDA and the CUBLAS libraries. Only Ampere GPUs are currently supported in order to have hardware BF16 support. The LibNC custom CUDA memory allocator allocates memory by chunks of 500 MB.

3.5 BF16 support

BF16 (truncated IEEE 32 bit floats to 16 bits) are supported on both the CPU and GPU backends. The ADAM optimizer internally keeps the low 16 bit part of the parameters so that no precision is lost during the gradient update.

4 License

The LibNC library is free to use as a binary shared library. Contact the author if access to its source code is required.