How to contribute

========

Thanks for taking the time to contribute to tiny-dnn! The following are a few guidelines for contributors.

These are just guidelines, not rules, and feel free to propose changes to this document in a pull request.

## Getting Started

- Make sure you have a C++11 compiler.

- Make sure you have a GitHub account.

- Register a report about your issue.

- Check [the issue list](https://github.com/tiny-dnn/tiny-dnn/issues) to see if the problem has already been reported.

- This can be skipped if the issue is trivial (fixing a typo, etc).

## Making Changes

- Create a topic branch where you want to base your work.

- This is usually the ```master``` branch.

- Make commits.

- Make sure you have added the necessary tests for your changes.

- Make sure you're sticking with our code style. You can run [`clang-format`](http://clang.llvm.org/docs/ClangFormat.html) manually or by using [pre-commit hook](https://github.com/arraiy/dacron/blob/master/etc/git/hooks/pre-commit). Currently `clang-format-4.0` is used.

- Submit a pull request.

- Make sure all CI builds are passed.

## Coding guides

- Keep header-only

- Keep dependency-free

- If your change requires 3rd party libraries, this should be \_\_optional\_\_ in tiny-dnn.

Please guard your 3rd party dependent code by ```#ifdef - #endif``` block, and write CMakelist option to enable the block -

but lesser these switches, the better.

- Keep platform-independent

- Use C++ standard library instead of Windows/POSIX dependent API

- CPU/GPU optimized code should be extracted as a separated file, and should be guarded as preprocessor macro.

### Preferred coding style

- Use [Google coding style guide](https://google.github.io/styleguide/cppguide.html) with some exceptions:

- Use ```CNN\_NAME\_OF\_THE\_MACRO``` style for preprocessor macros.

- Use ```snake\_case``` for rest of identifiers.

- ["We do not use C++ exceptions"](https://google.github.io/styleguide/cppguide.html#Exceptions) - We are using exceptions which throw ```tiny\_dnn::nn\_error``` or its subclass to keep error handling simple.

- ["Avoid using Run Time Type Information (RTTI)"](https://google.github.io/styleguide/cppguide.html#Run-Time\_Type\_Information\_\_RTTI\_) - We are using RTTI for serialization/deserialization.

- ["All parameters passed by reference must be labeled const"](https://google.github.io/styleguide/cppguide.html#Reference\_Arguments) - We sometimes use non-const reference to 1) avoid null-pointer dereference, or 2) keep code clean (especially when overloading ```operator << (std::ostream&,T)```.

- ["All header files should have #define guards to prevent multiple inclusion"](https://google.github.io/styleguide/cppguide.html#The\_\_define\_Guard) - We are using ```#pragma once``` because include guards are error-prone. It is implementation defined, but many compilers [support it](https://en.wikipedia.org/wiki/Pragma\_once#Portability).