An Automated Build of C++ Tensorflow Projects on Windows

Running Tensorflow in C++ requires the use of a static library that can be tricky and time-consuming to link to on Windows. Instead of setting each Tensorflow link manually in Visual Studio, we can use CMake to manage build configurations. Currently, the Tensorflow repository provides CMake instructions (seen here) for building an example program within the Tensorflow source directory, however, it is often preferable for your project to live outside of the Tensorflow directory.

I wanted to make use of CMake, but keep the project separate from the Tensorflow source. One approach is to add the Tensorflow library as a subdirectory of the main project. I created a new CMakeLists.txt for the main project and used the command add\_subdirectory to invoke the CMakeList of the Tensorflow library and generate a project that automatically checks/builds Tensorflow through its lifetime. However, references and links from the Tensorflow library are not visible to the main project, and must be rewritten into the project’s CMakeList.

I also encountered some strange behavior resulting from the Tensorflow CMakeList’s use of CMAKE\_BINARY\_DIR, which, unlike CMAKE\_CURRENT\_BINARY\_DIR, is not updated to reflect the subdirectory. Attempting to call the Tensorflow CMake from outside of the main project resulted in the Tensorflow build looking for dependencies in the wrong places and required too many changes to the original Tensorflow CMakeList to fix.

modify the CMake of the