TensorFlowSharp are bindings to the native TensorFlow library.

You can either use the TensorFlow C-library release binaries, or build

your own from source. Here are some pre-built TensorFlow binaries you

can use for each platform:

- Linux

- CPU-only: https://storage.googleapis.com/tensorflow/libtensorflow/libtensorflow-cpu-linux-x86\_64-1.1.0.tar.gz

- GPU-enabled: https://storage.googleapis.com/tensorflow/libtensorflow/libtensorflow-gpu-linux-x86\_64-1.1.0.tar.gz

- Mac: https://storage.googleapis.com/tensorflow/libtensorflow/libtensorflow-cpu-darwin-x86\_64-1.1.0.tar.gz

- Windows: https://storage.googleapis.com/tensorflow/libtensorflow/libtensorflow-cpu-windows-x86\_64-1.2.0-rc0.zip

Unpack the above .tar.gz suitable for your system on a prefix that your

system's dynamic linker can use, for example, go to `/usr/local` and unpack there.

Mac note: the package contains a `.so` file, you will need to rename this to `.dylib` for

it to work.

Once you do that, you need to open the solution file on the top

level directory and build. This will produce both the TensorFlowSharp

library as well as compile the tests and samples.

## Building your own native TensorFlow library

You will wan to use Visual Studio 2017 or Visual Studio for Mac to build.

To build the TensorFlow C library from source,

[follow these instructions](https://github.com/tensorflow/tensorflow/blob/master/tensorflow/go/README.md#building-the-tensorflow-c-library-from-source).

This includes checking out the Tensorflow sources, installing Bazel,

and building the core.

Once you do that, you will need to build the shared library.

First, in the tensorflow directory, run:

```bash

./configure

```

and answer the various prompts about your build. Important:

building with CUDA support provides better runtime performance

but has additional dependencies as discussed in the Tensorflow

installation Web page.

Once configured, run:

```bash

bazel build -c opt //tensorflow:libtensorflow.so

```

If you want debug symbols for Tensorflow, while debugging the binding:

```bash

bazel build -c dbg --strip=never //tensorflow:libtensorflow.so

```

You will need the generated library (`libtensorflow.so`) to be installed in a

system accessible location like `/usr/local/lib`

On Linux:

```bash

sudo cp bazel-bin/tensorflow/libtensorflow.so /usr/local/lib/

```

On MacOS:

```bash

sudo cp bazel-bin/tensorflow/libtensorflow.so /usr/local/lib/libtensorflow.dylib

```

## Running the test

I am currently using Visual Studio for Mac to do the development, but this

should work on Windows with VS and Linux with MonoDevelop.

Before the solution will run you will need the shared library generated to

be on a location accessibly by the Mono runtime (for example /usr/local/lib).

While Tensorflow builds a library with the extension .so, you will need

to make sure that it has the proper name for your platform (tensorflow.dll on Windows,

tensorflow.dylib on Mac) and copy that there.

Tensorflow is a 64-bit library, so you will need to use a 64-bit Mono to run,

at home (where I am doing this work), I have a copy of 64-bit Mono on /mono,

so you will want to set that in your project configuration, to do this:

Ensure that your Build/Compiler settings set "Platform Target" to "x64".

Open the solution file in the top directory, and when you hit run, this will

run the API test.