绑定，它的实现原理是：warpCTC 首先编译成一个动态库 (libwarpctc.so)，然后在 TensorFlow 里实现一个自定义的 Operation 来使用这个动态库。由于 WarpCTC 最近没有太多的维护，所有和新版本的 TensorFlow 的基础有一些问题，所有首先介绍怎么安装 WarpCTC 以及 TensorFlow 的绑定。

安装 WarpCTC

作者是从源代码安装的 Tensorflow 1.6.0，使用的是 CUDA-9.1。如果读者是通过 pip 安装的，有些步骤可能需要注意和作者的环境差别

1. 得到 tensorflow 源代码

```
git clone https://github.com/tensorflow/tensorflow.git
git checkout r1.6.0
```

说明：如果读者使用的 Tensorflow 版本是别的版本，请 checkout 到相应的版本。

2. 设置环境变量 TENSORFLOW_SRC_PATH

```
#export TENSORFLOW_SRC_PATH=/path/to/tensorflow
```

读者请把这个环境变量设置成自己的路径

3. 修改配置对于 GCC5，我们需要加入 C++ 选项 -D_GLIBCXX_USE_CXX11_ABI=0。

另外如果是新版本的 Tensorflow，会出现 import warpctc_tensorflow 时会出现 undefined symbol: _ZTIN10tensorflow8OpKernelE。根据 https://github.com/tensorflow/tensorflow/issues/13607 https://www.jianshu.com/p/d714073594b6，我们需要链接时使用 tensorflow_framework.so

此外新版本的 Tensorflow nsync_cv.h 文件的位置也会发生变化，因此需要做如下修改：

```
lili@lili-Precision-7720:~/codes/warp-ctc$ git diff CMakeLists.txt
diff --git a/CMakeLists.txt b/CMakeLists.txt
index cdb4b3e..ec20845 100644
--- a/CMakeLists.txt
+++ b/CMakeLists.txt
@@ -6,6 +6,7 @@ ENDIF()

project(ctc_release)

+set(CMAKE_CXX_FLAGS "${CMAKE_CXX_FLAGS} -D_GLIBCXX_USE_CXX11_ABI=0 -L
    /home/lili/py3-env/lib/python3.5/site-packages/tensorflow
    -ltensorflow_framework")
```

上面是我做的修改，请读者根据自己的环境修改 tensorflow_framework.so **的路径，也就是** GCC **的**-L **参数，作者的位置是**/home/lili/py3-env/lib/python3.5/site-packages/tensorflow。

```
lili@lili-Precision-7720:~/codes/warp-ctc$ git diff
    tensorflow_binding/setup.py

...

--- a/tensorflow_binding/setup.py
+++ b/tensorflow_binding/setup.py
@@ -52,11 +52,13 @@ root_path = os.path.realpath(os.path.dirname(__file__))

tf_include = tf.sysconfig.get_include()
tf_src_dir = os.environ["TENSORFLOW_SRC_PATH"]
-tf_includes = [tf_include, tf_src_dir]
+tf_includes = [tf_include, tf_src_dir,
    ("/home/lili/py3-env/lib/python3.5/site-packages/tensorflow/include/external/nsync/publ
...
extra_compile_args = ['-std=c++11', '-fPIC']
+extra_compile_args += [ '-D_GLIBCXX_USE_CXX11_ABI=0']
...

-
+import tensorflow as tf
+TF_LIB=tf.sysconfig.get_lib()
+print(TF_LIB)
ext = setuptools.Extension('warpctc_tensorflow.kernels',
sources = lib_srcs,
language = 'c++',
include_dirs = include_dirs,
-                       library_dirs = [warp_ctc_path],
+                       library_dirs = [warp_ctc_path, TF_LIB],
runtime_library_dirs = [os.path.realpath(warp_ctc_path)],
-                       libraries = ['warpctc'],
+                       libraries = ['warpctc','tensorflow_framework'],
extra_compile_args = extra_compile_args)
```

setup.py **需要修改** 3 **个地方，第一个就是** tf_includes **里增加** nsync_cv.h **头文件的路径；第二个就是增加** C++ **选项**-D_GLIBCXX_USE_CXX11_ABI=0**；第三个**

就是在 setuptools.Extension 里增加 tensorflow_framework 库和它的路径。

4. build

```
cd warp-ctc
mkdir build; cd build
cmake ..
make
```

5. 安装

```
python setup.py install
```

6. 测试安装是否成功

```
python
import warpctc_tensorflow
```

如果没有错误信息，那就说明成功了，否则读者可能需要根据错误反馈自行解决。

运行代码

```
git clone https://github.com/fancyerii/lstm_ctc_ocr.git
cd lstm_ctc_ocr
./train.sh
```

代码阅读

我们这里介绍最新的 beta 版的代码，它的效果最好，经过几万次的迭代就能达到 97% 以上的准确率。

1. 数据处理

最早的版本训练的时候提前把数据用验证码生成器生成出来，现在最新的版本是每次训练的时候实时的生成图片，这样不需要读取磁盘，但是 CPU 消耗更高。当然最主要的好处是每次都是完全不同的图片，这相对于有无穷多的训练数据，避免过拟合到特定的数据上。

DeepSpeech

下载代码 git lfs 支持

lili@Ubuntu:/bigdata/lili/kaldi/egs/voxforge/s5/voxforge/selected/Aaron-20080318-kdl/wav/b0019.wav Aaron-20080318-kdl/mfc/b0019 HIS SLIM HANDS GRIPPED THE EDGES OF THE TABLE his slim hands groptages of the table