Building Deepspeech for Jetson Nano Platform

DeepSpeech is an open source Speech-To-Text engine, using a model trained by machine learning techniques based on Baidu's Deep Speech paper. It uses TensorFlow with a C++ implementation.

DeepSpeech provides binaries for several platforms. So ideally, the pre-built binaries for performing inference with a trained model can be installed with `pip3`.

A pre-trained English model for performing speech-to-text can be downloaded (along with other important inference material) from the DeepSpeech releases page. Alternatively, you can run the following command to download and unzip the model files in your current directory:

- `wget https://github.com/mozilla/DeepSpeech/releases/download/v0.5.1/deepspeech-0.5.1-models.tar.gz`
- `tar xvfz deepspeech-0.5.1-models.tar.gz`
- `wget https://github.com/mozilla/DeepSpeech/releases/download/v0.5.1/audio-0.5.1.tar.gz`
- `tar xvfz audio-0.5.1.tar.gz`

Once everything is installed, you can then use the `deepspeech` binary to do speech-to-text on short (approximately 3-second long) audio files as such:

- `pip3 install deepspeech`
- `deepspeech --model models/output_graph.pbmm --alphabet models/alphabet.txt --lm models/lm.binary --trie models/trie --audio my_audio_file.wav`

I successfully ran this on RPi3. But, unfortunately we cannot do this on Jetson Nano platform. We have to build the binaries on our own as the Project DeepSpeech does not support this platform. The following step-by-step instructions to build the deepspeech binaries, are borrowed from here with some changes.

Dependencies:

Following dependencies need to be installed before compiled deepspeech code.

1. General tensorflow requirements:
    - `sudo apt install python3-dev python3-pip`

- pip3 install -U --user pip six numpy wheel setuptools
- sudo apt install python-testresources

2. Libsox
    - sudo apt-get install libsox-dev
3. Bazel. It needs to be compiled and installed. There is no pre-built package for Jetson Nano. You can build bazel from scratch by using the instructions from here.
4. Mozilla's TensorFlow `r1.14` branch
5. SWIG >= 3.0.12
    - sudo apt-get install swig
6. node-pre-gyp (for Node.JS bindings only)

It is required to use their fork of TensorFlow since it includes fixes for common problems encountered when building the native client files.

One you install all the dependencies, check out their Tensorflow and Deepspeech sources and do the following.

-
- `git clone https://github.com/mozilla/tensorflow.git`
- `cd tensorflow`
- `git checkout origin/r1.14`
- `./configure`

Compile DeepSpeech

Within Mozilla's TensorFlow checkout, create a symbolic link to the DeepSpeech `native_client` directory. Assuming DeepSpeech and TensorFlow checkouts are in the same directory, do:

- `cd tensorflow`
- `ln -s ../DeepSpeech/native_client ./`

You can now use Bazel to build the main DeepSpeech library, `libdeepspeech.so`, as well as the `generate_trie` binary.

- ```
  bazel build --workspace_status_command="bash
  native_client/bazel_workspace_status_cmd.sh" --config=monolithic
  --config=cuda -c opt --copt=-O3 --copt="-std=gnu99"
  --copt="-D_GLIBCXX_USE_CXX11_ABI=0" --copt=-fvisibility=hidden
  //native_client:libdeepspeech.so //native_client:generate_trie
  ```

The generated binaries will be saved to `bazel-bin/native_client/`.
  ```

Compile Language Bindings

Now, `cd` into the `DeepSpeech/native_client` directory and use the `Makefile` to build all the language bindings (C++ client, Python package, Nodejs package, etc.). Set the environment variable `TFDIR` to point to your TensorFlow checkout.

- `TFDIR=~/tensorflow`
- `cd ../DeepSpeech/native_client`
- `make deepspeech`

Installing your own Binaries

After building, the library files and binary can optionally be installed to a system path for ease of development. This is also a required step for bindings generation.

```
PREFIX=/usr/local sudo make install
```

It is assumed that `$PREFIX/lib` is a valid library path, otherwise you may need to alter your environment.

Install Python bindings

Included are a set of generated Python bindings. After following the above build and installation instructions, these can be installed by executing the following commands (or equivalent on your system):

- `cd native_client/python`
- `make bindings`
- `pip install dist/deepspeech*`

The API mirrors the C++ API and is demonstrated in client.py. Refer to deepspeech.h for documentation.

Issues faced while building the binaries and python bindings:

I had to try a combination of many different ways to build this. At last, following packages were suitable for building the master branch of Deepspeech.

1. GCC and G++ version is very important. I tried with 4.8, 5, and 7. Only version 5 worked for me. You can install different versions using commands like below.

- `sudo apt-get install gcc-5`
- `sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-5 50`
- `sudo apt-get install g++-5`
- `sudo update-alternatives --install /usr/bin/g++ g++ /usr/bin/g++-5 50`

2. Bazel version. Mozilla's tensorflow expects a bazel version that is between 0.24.1 and 0.25.2. No greater version and no former version works.
3. Correct CUDA compute capability and tensorflow configuration is needed. I used the following configuration for tensorflow.

root@essence-desktop:/home/essence/tensorflow# ./configure

WARNING: Running Bazel server needs to be killed, because the startup options are different.

WARNING: --batch mode is deprecated. Please instead explicitly shut down your Bazel server using the command "bazel shutdown".

You have bazel 0.24.1- (@non-git) installed.

Please specify the location of python. [Default is /usr/bin/python]: /usr/bin/python3

Found possible Python library paths:

/usr/local/lib/python2.7/dist-packages

/usr/lib/python2.7/dist-packages

Please input the desired Python library path to use. Default is [/usr/local/lib/python2.7/dist-packages] Enter

Do you wish to build TensorFlow with XLA JIT support? [Y/n]: n

No XLA JIT support will be enabled for TensorFlow.

Do you wish to build TensorFlow with OpenCL SYCL support? [y/N]: n

No OpenCL SYCL support will be enabled for TensorFlow.

Do you wish to build TensorFlow with ROCm support? [y/N]: n

No ROCm support will be enabled for TensorFlow.

Do you wish to build TensorFlow with CUDA support? [y/N]: y

CUDA support will be enabled for TensorFlow.

Do you wish to build TensorFlow with TensorRT support? [y/N]: n

No TensorRT support will be enabled for TensorFlow.

Found CUDA 10.0 in:

/usr/local/cuda/lib64

/usr/local/cuda/include

Found cuDNN 7 in:

/usr/lib/aarch64-linux-gnu

/usr/include

Please specify a list of comma-separated CUDA compute capabilities you want to build with. You can find the compute capability of your device at: https://developer.nvidia.com/cuda-gpus.

Please note that each additional compute capability significantly increases your build time and binary size, and that TensorFlow only supports compute capabilities >= 3.5 [Default is: 3.5,7.0]: 5.3

Do you want to use clang as CUDA compiler? [y/N]: n

Do you wish to build TensorFlow with MPI support? [y/N]: n

No MPI support will be enabled for TensorFlow.

Please specify optimization flags to use during compilation when bazel option "--config=opt" is specified [Default is -march=native -Wno-sign-compare]: Enter

4. Adding Jetson Nano architecture support. Update the following files.

diff --git a/tensorflow/lite/kernels/internal/BUILD b/tensorflow/lite/kernels/internal/BUILD

```
index 4be3226938..7226f96fdf 100644
--- a/tensorflow/lite/kernels/internal/BUILD
+++ b/tensorflow/lite/kernels/internal/BUILD
@@ -22,15 +22,12 @@ HARD_FP_FLAGS_IF_APPLICABLE = select({
NEON_FLAGS_IF_APPLICABLE = select({
    ":arm": [
        "-O3",
-       "-mfpu=neon",
    ],
    ":armeabi-v7a": [
        "-O3",
-       "-mfpu=neon",
    ],
    ":armv7a": [
        "-O3",
-       "-mfpu=neon",
    ],
    "//conditions:default": [
        "-O3",
```

```
diff --git a/third_party/aws/BUILD.bazel b/third_party/aws/BUILD.bazel
index 5426f79e46..e08f8fc108 100644
--- a/third_party/aws/BUILD.bazel
+++ b/third_party/aws/BUILD.bazel
@@ -24,7 +24,7 @@ cc_library(
        "@org_tensorflow//tensorflow:raspberry_pi_armeabi": glob([
            "aws-cpp-sdk-core/source/platform/linux-shared/*.cpp",
        ]),
-       "//conditions:default": [],
+       "//conditions:default":
glob(["aws-cpp-sdk-core/source/platform/linux-shared/*.cpp",]),
    }) + glob([
        "aws-cpp-sdk-core/include/**/*.h",
```

```
        "aws-cpp-sdk-core/source/*.cpp",
```

```
diff --git a/third_party/gpus/crosstool/BUILD.tpl b/third_party/gpus/crosstool/BUILD.tpl
index db76306ffb..184cd35b87 100644
--- a/third_party/gpus/crosstool/BUILD.tpl
+++ b/third_party/gpus/crosstool/BUILD.tpl
@@ -24,6 +24,7 @@ cc_toolchain_suite(
         "x64_windows|msvc-cl": ":cc-compiler-windows",
         "x64_windows": ":cc-compiler-windows",
         "arm": ":cc-compiler-local",
+        "aarch64": ":cc-compiler-local",
         "k8": ":cc-compiler-local",
         "piii": ":cc-compiler-local",
         "ppc": ":cc-compiler-local",
```

5. Create Swap. I am not sure if this helped but I did it as suggested from here.
   - `$ fallocate -l 8G swapfile`
   - `$ ls -lh swapfile`
   - `$ sudo chmod 600 swapfile`
   - `$ ls -lh swapfile`
   - `$ sudo mkswap swapfile`
   - `$ sudo swapon swapfile`
   - `$ swapon -s`

6. Model compatibility
   - DeepSpeech models are versioned to keep you from trying to use an incompatible graph with a newer client after a breaking change was made to the code. If you get an error saying your model file version is too old for the client, you should either upgrade to a newer model release, re-export your model from the checkpoint using a newer version of the code, or downgrade your client if you need to use the old model and can't re-export it.

Errors:

1. c++: error: unrecognized command line option ‘-Wdate-time’;
    - Use -std=gnu99 option in the bazel configure command and with an installation of gcc-5 and g++-5.
2. ERROR: launchpadlib 1.10.6 requires testresources, which is not installed.
    - Install python-testresources using apt-get as mentioned in the begining.

References:

- https://github.com/mozilla/DeepSpeech
- https://github.com/mozilla/DeepSpeech/blob/master/native_client/README.md
- https://devtalk.nvidia.com/default/topic/1055131/jetson-agx-xavier/building-tensorflow-1-13-on-jetson-xavier/
- https://discourse.mozilla.org/t/deepspeech-installation-on-nvidia-jetson-tx2/30402/5
- https://github.com/mozilla/DeepSpeech/issues/2296