End Term Report

About Me

Name : Arafat Dad Khan
Email : arafat.da.khan@gmail.com
GitHub : Arafatk
Blog : medium
University : Indian Institute of Technology, Kharagpur (link)

Contents

Overview

My primary goal was to make sure the every function in the Tensorflow C API can be called properly and that Ruby SWIG works properly with the build of C API. C/C++ based helper functions are added so that everything works well with Ruby too. Along with this Ruby protobuf usage was to be shifted to C protobuf for speed.

This was quite challenging considering that the C API was a lot more convoluted that originally anticipated. But, thanks to the incredibly smart work of tensorflow team, almost everything I need to make the protobuf fully functional was available.

The graph api of tensorflow relied completely on Ruby Protocol Buffers gem. Everything worked fine for small graphs, but for large and convoluted graphs the programs crashed and the errors were completely indecipherable. Improper graph specification lead to direct crashes so identifying the mistakes was not possible when making large graphs.

With time Tensorflow team saw a lot of requests for language bindings and that's when they refined the tensorflow C API further and made it completely self

reliant(This was one of the goals below the release of Tensorflow 1.0). I emailed Asim Shankar and Jonathan Hseu from Tensorflow team and they both updated me with the latest developments and the potential targets for Tensorflow C API. Google decided to officially support Go, Haskell, Rust and Java (Some of these are still in progress). And jonathan even gave a talk on Tensorflow dev summit recently. With the new C compatible API problems such as incorrect specification while graph construction or improper input specification have been removed and a massive speed up is observed. Also Tensorflow team has many interesting goals for Tensorflow 2.0.

Results

I had been worked on using C API only for making graphs and I finally ended up making a complete protobuf generator with the new C API. Then I completely removed any unnecessary dependencies. The tests for the previous API relating to Constants and Placeholders were replicated exactly as before along with addition of new tests for new functionality.

Removing the dependencies on Ruby-protocol buffers and narray lead to a massive speed up and the overall testing time changed from 5-6 seconds to 0.4-0.8 seconds seconds even with the addition of a few different tests. The previous inception model took 10+ seconds to run but this one works in 1-1.5 seconds.

The only drawback is that tensorflow variables class was dropped because the current API operation construction with TF_FinishOperation using current C API is different from making operations by using Ruby Protocol buffers and defining names and attributes manually and that is why they can't work together.

The graph import and export facility is now over simplified and this helps to easily identify the mistakes in graph construction. You can see a simple file that works well and also generates a protobuf file(This is converted using file) with the current gem. Similarly, I have completely shifted the graph class to rely only on C API and deleted the original helper files(for ruby protobuf) and so it's easy to make graph and then finally check them in human readable form after conversion.

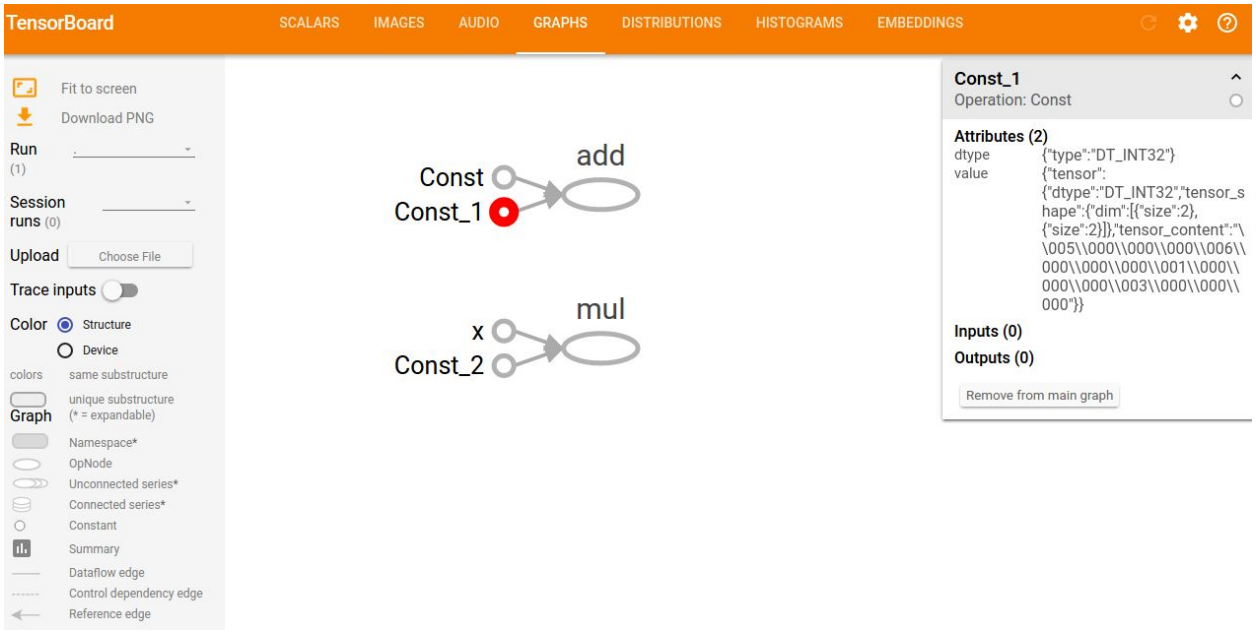
The session API has also been restructured to work fine well with TF_Operation instead of taking output names as parameters like previously and this is much safer as errors are very unlikely in this format. Scope class has been introduced for simplifying graph construction and status class has been introduced for easier error checking. Also eventually tensorflow will move to saved model for storing models which is even better than protobuf and I have added Saved Model functionality too to deal with that. I also faced some issues with the String data type encoding before but thanks to Asim Shankar, he pointed me in the right direction and I was was able to work with it.

All of my work can be seen in this pull request. I have done Automated testing with Circle CI and the tests are failing currently because it uses the docker image for tensorflow source and that needs to be updated. I will also email other developers about the progress to get some comments from them and merge finally when docker image is updated. I also hope to get comments from users to improve upon my design decisions.

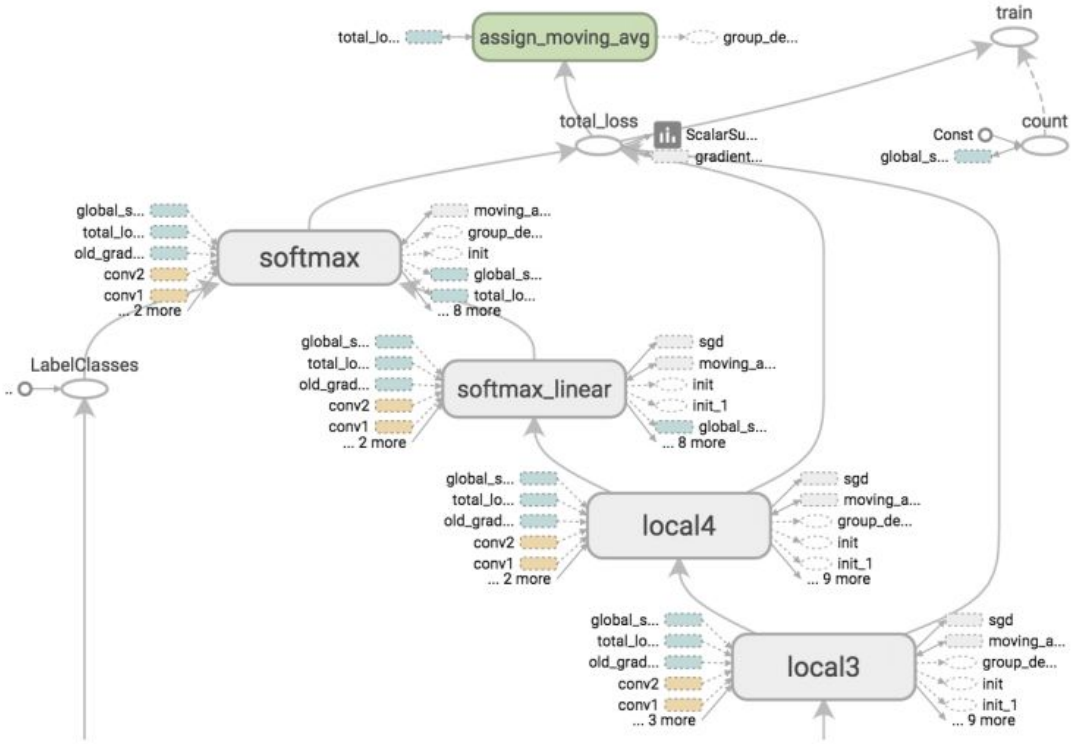
Tensorboard

I had originally anticipated adding tensorboard usage to be an extremely convoluted process but it turned out to be much simpler than I had anticipated. I have mentioned the entire process of using Tensorboard with Ruby in a markdown file and this can easily be used for more complicated examples too.

Now you can use the tensor board that helps us properly visualise Ruby graphs too



This photo above is a screenshot that I took after running the graph of the file in tensorboard. In simple cases like these the use of tensorboard seems very redundant, but it is very useful when dealing with very large graphs like the ones for inception model



Its very easy to use tensorboard and visualize what you have done.

Conclusion

I have completed the specified work for the tensorflow.rb and almost everything that I had targetted has been working fine. I have made the report very brief but any clarifications or any suggestions for improvement are very welcome.

Finally, I feel very grateful to Ruby community, my project mentors and many people who have given me extremely valuable recommendations to improve upon my work.