Tensorflex: Tensorflow bindings for the Elixir programming language

Anshuman Chhabra
Department of Computer Science
University of California
Davis, CA 95616
chhabra@ucdavis.edu

José Valim
Director and Founder of R&D
Plataformatec
Krakow, Poland
jose.valim@plataformatec.com.br

Abstract

Recently, with the advent of programmatic and practical machine learning tools, programmers have been able to successfully integrate applications for the web and the mobile with artificial intelligence capabilities. This trend has largely been possible because of major organizations and software companies releasing their machine learning frameworks to the public– such as Tensorflow (Google), MXnet (Amazon) and PyTorch (Facebook). Python has been the de facto choice as the programming language for these frameworks because of it's versatility and ease-of-use. In a similar vein, Elixir is the functional programming language equivalent of Python and Ruby, in that it combines the versatility and ease-of-use that Python and Ruby boast of, with functional programming paradigms and the Erlang VM's fault tolerance and robustness. However, despite these obvious advantages, Elixir, similar to other functional programming languages, does not provide developers with a machine learning toolset which is essential for equipping applications with deep learning and statistical inference features. To bridge this gap, we present Tensorflex, an open source framework that allows users to leverage pre-trained Tensorflow models (written in Python, C or C++) for Inference (generating predictions) in Elixir. Moreover, Tensorflex was written as part of a Google Summer of Code (2018) project by Anshuman Chhabra, and José Valim was the mentor for the same.

1 Introduction and Motivation

With the advent of Deep Learning (DL), it has become imperative for any programming language to support machine learning (ML) capabilities. Moreover, for functional programming languages such as Elixir [2] and Erlang [3], possessing ML/DL tools can lead to a dual benefit for developers as the Erlang VM's fault tolerant and robustness can be coupled with support for ML/DL. This can also lead to more interest from developers who have not utilized these languages before, and allow for more people to adopt functional programming paradigms in new systems and applications. Furthermore, existing Elixir users can also benefit from having ML/DL tools, as they can use these to improve and augment their extant applications with a wide variety of ML related features.

Tensorflow [1] is one of the most widely used deep learning frameworks for industry as well as research. Therefore, for the Google Summer of Code program [4], we sought to build Tensorflow bindings for Elixir, named Tensorflex [5]. The rest of the article is structured as follows: we detail the high level working of Tensorflex in the next section (Section 2). In Section 3, we cover some more aspects related to Tensorflex– such as the possible applications and the functions it currently possesses. In Section 4, we conclude and delineate future plans for Tensorflex.

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

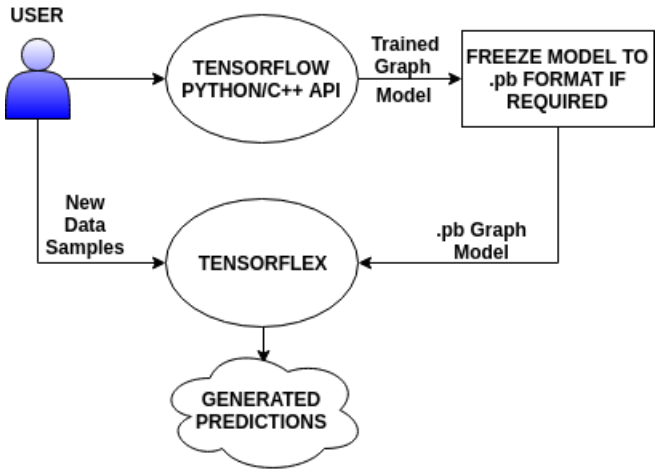

Figure 1: Working of Tensorflex

2 Overview of Tensorflex

Tensorflex is essentially built using the Tensorflow C API [6] and Erlang Native Implemented Functions (NIFs) [7] and at the moment only supports Inference. The open sourced code is present on GitHub [5] and allows users to use their pre-trained *protobuf* Tensorflow graph models for generating predictions from new data in Elixir. The Tensorflow `.pb` graph models can be created using the Python API just as in status quo. However, if developers utilize the checkpoint style of saving model data in Tensorflow (as is usually done in the Python API), there is a requirement of first *freezing* the graph, that is, converting it to the `.pb` format, before it can used in Tensorflex. We provide examples and scripts for *freezing* models in Python for use in Tensorflex in the GitHub repository [5]. Furthermore, the working of the Tensorflex framework is shown in Fig. 1.

Tensorflex also has a wide variety of functions which are not directly built atop the C API, but actually support ML and DL related functions. These include support for 2-D matrices, 3-D image matrices, and fast CSV-to-matrix loading, among others. These functions are generally natively written in C, and are many times faster than their counterparts in various Python packages. As an example, the `load_csv_as_matrix/2` function in Tensorflex loads some sample CSV data into Elixir as a matrix in just (on average) **1.711494** seconds whereas the `Pandas` Python package's function `read_csv` [8] takes (on average) **2.549233** seconds!

All of the aforementioned functions are listed as well as possess in-depth explanations for their functionalities in the Tensorflex documentation [9] (also written as part of Google Summer of Code, 2018). The `README` in the GitHub repository [5] also mentions these functions in a brief manner.

Tensorflex also supports all Neural Network architectures present in Tensorflow from version `r1.9` onwards. Examples are present on the GitHub repository [5] which use Google's Inception Model [13] based on Convolutional Neural Networks (CNNs) for image classification, Long-Short-Term-Memory (LSTM) based Recurrent Neural Networks (RNNs) for sentiment analysis of movie reviews, and also simple Feedforward Neural Networks for classification on simple datasets (such as the Iris Dataset [12]).

3 Further Details

3.1 Tensorflex Functions

Tensorflex contains three main structs which handle different datatypes. These are `%Graph`, `%Matrix` and `%Tensor`. `%Graph` type structs handle pre-trained graph models, `%Matrix` handles Tensorflex 2-D matrices, and `%Tensor` handles Tensorflow Tensor types. The functions as of Tensorflex `v0.1.2` are listed below and are covered in more detail in the Tensorflex documentation [9]:

- `read_graph/1`
- `get_graph_ops/1`
- `create_matrix/3`
- `matrix_pos/3`
- `size_of_matrix/1`
- `append_to_matrix/2`
- `matrix_to_lists/1`
- `float64_tensor/2`
- `float32_tensor/2`
- `int32_tensor/2`
- `float64_tensor/1`
- `float32_tensor/1`
- `int32_tensor/1`
- `string_tensor/1`
- `float64_tensor_alloc/1`
- `float32_tensor_alloc/1`
- `int32_tensor_alloc/1`
- `tensor_datatype/1`
- `load_image_as_tensor/1`
- `load_csv_as_matrix/2`
- `run_session/5`
- `add_scalar_to_matrix/2`
- `subtract_scalar_from_matrix/2`
- `multiply_matrix_with_scalar/2`
- `divide_matrix_by_scalar/2`
- `add_matrices/2`
- `subtract_matrices/2`
- `tensor_to_matrix/1`

3.2 Applications

Tensorflex has been utilized for Inference on a wide-variety of existing Tensorflow graph models. Moreover, Tensorflex functions are very simple to understand and are written keeping ease-of-use in mind, so it should be trivial to utilize the Tensorflex API for incorporating Inference pipelines of more models as well. Tensorflex thus allows for DL to be leveraged in Elixir using various Neural Network architectures for a multitude of applications. These can include image classification using the CNN Inception Model [13], sentiment analysis of movie reviews using LSTM-RNNs, as well as using MLPs for predicting classes in different datasets. Some of these applications that have been implemented [5,9,10] are expounded below:

- **Google CNN Inception Model**: The Inception Model [13] is used for image recognition and has 1008 labels for classes. The Inception example in Tensorflex has been covered in the documentation [9] under the `load_image_as_tensor/1` function as well as in the Examples section of the `README` on the GitHub repository [10]. Further uses include (as an example), clicking a *selfie* using a phone in real-time and then classifying it in Elixir in real-time using Inception [13].
- **Sentiment Analysis using RNNs**: In this example, one requires a single sentence review of movie. Then this textual data is inputted to an RNN based classifier and is classified as either positive or negative sentiment in Tensorflex. This example has been covered in the Examples section of the `README` in the GitHub repository [10].

- **Simple examples on simple datasets**: For example, these could use Feedforward Neural Networks similar to the Iris dataset [12] Tensorflex introductory example [11]. Other datasets such as the MNIST dataset could also be used. As another example, the digits could be hand-drawn in real-time in an Elixir application utilizing Tensorflex in the backend which then generates predictions.

4 Conclusion

Through this article, we delineate the Tensorflex framework which consists of Tensorflex bindings for Elixir. Enabling ML and DL for functional programming languages via open source is a positive step for both the functional programming language and machine learning communities. Traditionally Elixir and Erlang have not been associated with large-scale data analysis capabilities, as are generally required in ML or DL. Tensorflex instead asserts the contrary notion and allows developers/users to utilize Inference capabilities in their applications for the web or mobile.

Future work for Tensorflex is going to look toward implementing gradient support which allows for training models directly in Elixir itself, and the implementation of 3-D matrices.

References

[1] Martín Abadi et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Plataformatec. Elixir. `https://elixir-lang.org/`, 2012-2018. [Online; accessed 1-September-2018].

[3] Joe Armstrong. *Making reliable distributed systems in the presence of software errors*. PhD thesis, Mikroelektronik och informationsteknik, 2003.

[4] Google. Google Summer of Code, 2018. `https://summerofcode.withgoogle.com/`, 2018. [Online; accessed 1-September-2018].

[5] Anshuman Chhabra. Tensorflex. `https://github.com/anshuman23/tensorflex` [Online; accessed 1-September-2018].

[6] Tensorflow C API. `https://www.tensorflow.org/install/install_c` [Online; accessed 1-September-2018].

[7] Erlang NIFs. `http://erlang.org/doc/man/erl_nif.html` [Online; accessed 1-September-2018].

[8] `read_csv`. `https://pandas.pydata.org/pandas-docs/stable/generated/pandas.read_csv.html` [Online; accessed 1-September-2018].

[9] Anshuman Chhabra. Tensorflex v0.1.2 documentation. `https://hexdocs.pm/tensorflex/Tensorflex.html` [Online; accessed 1-September-2018].

[10] Anshuman Chhabra. Tensorflex - Examples. `https://github.com/anshuman23/tensorflex/#examples` [Online; accessed 1-September-2018].

[11] Anshuman Chhabra. An introduction to Tensorflex. `http://www.anshumanc.ml/gsoc/2018/06/14/gsoc/` [Online; accessed 1-September-2018].

[12] Ronald A Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2):179–188, 1936.

[13] Google Inception Model V3. `http://download.tensorflow.org/models/image/imagenet/inception-2015-12-05.tgz` [Online; accessed 1-September-2018].