3. MNIST in a C header

Author(s)

Name	Institution	Mail Address	Social Contacts
Lucio Anderlini	INFN Sezione di Firenze	Lucio.Anderlini@fi.infn.it	Hangouts: l.anderlini@gmail.com
Matteo Turisini	Università La Sapienza	Matteo.Turisini@roma1.infn.it	N/A

How to Obtain Support

Mail	Lucio.Anderlini@fi.infn.it
Social	Hangouts: l.anderlini@gmail.com
Jira	N/A

General Information

ML/DL Technologies	Forward Neural Networks; Deployment in C
Science Fields	Generic
Difficulty	Medium
Language	English
Type	fully annotated

Software and Tools

Programming Language	Python, C
ML Toolset	Keras + Tensorflow 1.x
Additional libraries	scikit-learn
Suggested Environments	INFN-Cloud VM, bare Linux Node, Google CoLab

Needed datasets

Data Creator	scikit-learn community
Data Type	image
Data Size	< 1 GB
Data Source	scikit-learn package

Short Description of the Use Case

In the life of a physicist, the time arrives when you get disgusted by the software dependencies that evaluating a simple neural network requires and you want the plain, pure and clean function to be evaluated in a whatever context.

For me it was to run a trained algorithm on a Arduino board that we were using in the lab to stabilize the temperature of a device, for a colleague it was to use HLS to push a simple neural network in his FPGA project for another friend it was to feed into the TTree::Draw function of ROOT the evaluator of a neural network trained in keras to avoid rewriting the rest of the software stack originally designed for TMVA.

In this tutorial we propose an example of how to translate a simple neural network into single header file (.h) that can be exported and #included a whatever C application without a single dependency to get the trained algorithm hard-coded and compiled together with your program.

Colleagues from particle physics have used a similar approach to provide a tensorflow-free python evaluator of (more complicated) neural networks. Check out the tf.deploy package for additional information: https://github.com/riga/tfdeploy

Back to C/C++, you can checkout the LWTNN project born within the ATLAS community (https://github.com/lwtnn/lwtnn), which needs few external dependencies and cannot be used on Arduino for instance, but allows to load dynamically the neural networks from disk avoiding all that horrible hard-coding.

But since this is a tutorial and we are not writing production code, let's have a look.

We will take as problem the MNIST problem which is a kind of Hello World for multi-category classification tasks in machine learning, in a way that we can also review this exercise that cannot be missing in a ML-related web page.

The other reason is that the original data that we used to train this are not public.

How to execute it

Download and run the jupyter notebook: https://github.com/landerlini/MLINFN-TutorialNotebooks/blob/master/LHCbMasterclassExplained.ipynb

Requirements

To run this exercise you will need a plain installation of python with tensorflow 1.x (should work with tensorflow 2.x, but it was not tested).

Contents

Presentation made on 27 Jul 2020 : https://agenda.infn.it/event/23648/contributions/118758/attachments/74371/94574/Teaching_Material_on_the_KB.pdf