**ISRLU Custom Layer Plugin**

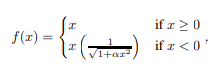
(Inverse Square Root Linear Unit)

**Table of Contents:**

* Description of activation function
* What all files does the sample contain
* Prerequisites
* Running this sample
* Results
* Known issues
* Citations

1. **Description of activation function**:

Inverse Square Root Linear unit is a activation function proposed by Brad Carlile and team in 2018[1]. It has a smoothly saturating negative values for negative inputs and the identity for positive inputs. The mathematical function is given as :



Where α is a hyperparamter which controls the output to which it saturates . The primary advantage is that it is able to faster computation ability in comparision to other functions like ELU(Exponential Linear Unit).

ISRLU function is not yet supported in TensorRT and we developed a custom plugin for the same.

1. **What all files does the sample contain:**

* ISRLU Model/ISRLU:
  + Contains the “MainFile(1).ipynb” which has the code in which the actual implantation is shown .
  + “my\_model” contains the “saved\_model.pb” generated by the model which is given into TensorRT.
  + Model\_trtfp32 contains the final “saved\_model.pb” generated by the TensorRT.
  + It also contains all the other files generated during execution.
* Plugin:
  + “IsrluPlugin.cu” is the main CUDA kernel which is called by TensorRT whenever the activation is called
  + “IsrluPlugin.h” is the header file for the custom plugin(ISRLU)
* “ISRLU\_model\_pytorch.py”(extra file) is the pytorch model implementing the ISRLU activation function.
* “TRT\_inference\_engine\_creation.py” : jupyter notebook for generating the build engine.
* “trt\_graph.pb”(extra) : sample Testing graph.

1. **Pre-requisites :**

A jupyter notebook is given which can be run google Colab notebook It is used the integrated way. If user wants to use local system they are expected to have Tensort 4.0+ and CUDA installed in their system.

1. **Running this sample:**

* **Native way:**
* The user can take the final “saved\_model.pb” from model\_trtfp32 folder and parse it into uff format using convert-to-uff <”name of file.pb”> command.
* For creating the inference engine, the user can run “TRT\_inference\_engine\_creation.ipynb” file.
* The isrlu\_config.py can be used in your model . This uses graph surgeon to create a node and in turn call the custom plugin implemented in Isrlu.cu and Isrlu.h files.
* **Integrated way:**
* For running this sample one can open “MainFile(1).ipynb in collab notebooks and execute the cells.

1. **Results:**

The inference was done on the MNIST dataset.The results before and after were as shown:

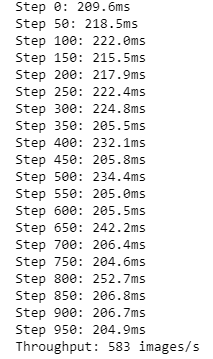
 

Fig 1: without TRT the accuracy was 98.8% Fig 2: with custom plugin the acc was 98%

1. **Known Issues:**

The user might experience some compatability issues depending upon the TensorRT version they use . In that case user can refer the TensorRT documentation.

1. **Citations:**

[1]: [arXiv:1710.09967](https://arxiv.org/abs/1710.09967) [cs.LG]