Status of fast ECAL reconstruction

Dimitrije Maletic

Institute of Physics Belgrade, Serbia

13.12.2021, 2nd SPD collaboration meeting

Table of Contents

Introduction

- For online filter we need fast ECAL reconstruction, and this can be done by using deep learning of Artificial Neural Networks (ANN). We plan to use Convolutional Neural Network (CNN) which will be trained (and tested) using simulated events in form of whole surface image of ECAL (Barrel and Endcaps).

- Since online filter will be using CNNs which are built by training of simulated events, we need to monitor it’s outputs since CNN can not be trained for all possible detector conditions and states. The monitoring will be done by comparison of CNN output decisions and standard ECAL reconstruction.

- Another point is that the pi0/gamma separation is using also ANN, but simpler, Multi Layer Perceptron (MLP) with only one hidden layer. For this network, input variables are not whole surface images but variables which we get with clustering and somewhat significant calculations. This method of pi0/gamma separation can be used for comparison and to indicate preferable separation efficiency of CNN outputs.

- During development stage of fast ECAL reco software, the choice of software tools will depend on developers' preference, but I feel that application of CNN which will actually run on GPUs will need to be as simple as possible.

- The simplicity and speed of implementation of CNNs while running GPUs will depend on the way how the data transfers and memory allocations are organised, as well as exluding all abstract libraries we can.

Software packages and programming languages selection

- Examples of End-to-end event reconstruction is growing. Many use CMS open data to train and test their network architectures. Also, many use training for specific detectors, most frequently track reconstruction. Some examples of network implementations are ResNet, TrackNet and GraphNet.

- The response of CNNs on GPUs needs to be fast, so it is good to go towards using C/C++ code as running code, or code for writing applications of CNN evaluations, on (NVidia) Graphical cards.

- Besides code running on GPUs, if we do not need speed for training the networks, training can be done by any software package, C/C++ or python based (pyTorch). Result of training and development of CNNs will be, most importantly, **network architecture**, and **network parameters**. Those can be converted from any package to C/C++ implementation which will be running on GPUs.

- For simplicity, and my preference, and possibility of comparison with other software packages used for similar purpose, I chose to test CNNs using TMVA package in ROOT. After training, I will use those results for developing/transferring resulting architecture and use network parameters from training for C/C++ implementation on GPUs.

Some properties of CNN for Fast ECAL reco

- CNN architecture could have two or three convolutional layers and one or two dense (MLP) layers.

- Also, the Residual Networks' features are considered and will be tested if they will improve efficiency and speed.

- There is also possibility of Region of Interest feature to be tested and used, which will give us also the possibility that some variables to be transferred directly to dense networks like very important S_3x3 / S_5x5 variable.

- Could train CNN for robustness by supplying additional information on status of detector to dense network, or, better to have multiple outputs "lines" which can be used for event selection, like

    default: OK - NOT_OK, MagFld_OFF: OK - NOT_OK,

    or <some_detector>_off, <some_detector>_not_installed, EcalBarrel_off, EcalEndcap_off

and so on, all detector status we can expect. This could lower the frequency of future retraining of networks.

- Need for significant training sample. Also, sample in files that contain several time slices.

- Input images can have various forms, like stacking different detectors, time slices – like, three at a time or all at once. Also, time slices could be weighted – will look like signal processing. Also, images can be full or small - Region of Interest ones.

- In connection of training, transfer of training – which means that we can have the continuation of training, is favorable option which would greatly cut down the retraining time.

Some properties of application of CNN evaluation on GPU

- We should aim at as simple as possible implementation of application of CNN evaluation (getting the CNN output from GPUs). The code could therefore be written in C/C++ with only using cuda and it's **nvcc** compiler. There is also OpenCL for vendors other than Nvidia.

- Try to avoid using more abstract libraries, like cudnn, but which can be still used for CNN training applications. This way of implementation is useful for different CNN architectures and modified (custom) ones to include different network features, like residual and Region of Interest networks features, among others.

- We need to take a good look at memory allocations on Graphical card. Shared memory. Multiple layers of memory.
    - Aim to minimize to amount of data transferred between the host and device when possible.
    - higher bandwidth using page-locked or pinned memory
    - one initial transfer of most files, execution programs and parameters

    In case of implementation problems, we can get back to managed memory where copying is handled automatically by cuda runtime system.

- Optimize application using **nvprof**, a command line tool to check the collection of a timeline of CUDA-related activities on both CPU and GPU.

- We need to think about having C/C++ code be resident on graphical card after initial request, and how we get response by listening to ports, or is accessing the code directly preferable.

Some properties of application of CNN evaluation on GPU

- I made first, very simple and fast implementation of MLP (dense) network in 2007 (single threaded). No external libraries, just reading parameter files and calculating NN output in two loops.

- Simplest considered CNN has two convolutional layers more than MLP implementation.

- Need to modify simplest C/C++ applications to able to run like multi thread application on GPUs by using cuda and it’s nvcc compiler.

- MLP (dense) networks have most of CNN network parameters, and convolutional layers need more processing to get output (many matrix multiplications).

- Writing custom C/C++ code allows different implementations and modifications, like adding Residual connections or Region of Interest search, where images taken into consideration could be built around some tower energy greater than threshold (similar to clustering), which speeds up training significantly, and also can run in parallel with full image processing CNNs.

- Should report more frequently to software meetings about progress and tests to get more information on developments in subjects in connection to fast ECAL reco.

Conclusion

- Currently the focus is on coding and (response time) testing of simple application for CNN evaluation using GPUs.

- There is still work to be done to determine the best performing architecture of CNN.

- Residual and Region of interest network features are considered as an extension to CNNs.

- Produce training samples with time slices.

- Report more frequently to software meetings.

Thank you for your attention!