Fast ECAL reconstruction using deep learning

Dimitrije Maletic

Institute of Physics Belgrade, Serbia

09.06.2021, SPD collaboration meeting

The goal is to present tools, and to present possible next steps in looking into fast ECAL reconstruction using Convolutional Neural Networks (CNN). There are no results presented, just some tests.

Overview:

- Some notes on evolution of gamma / pi0,... separation

- Basics of Convolutional Neural Networks

- Setup for MC simulation and Neural Network training

- Outputs (logs) for Neural Network training and testing, design of NNs

- About MC simulation and preparation of training and testing events for NNs

- Ideas about next steps in using CNNs

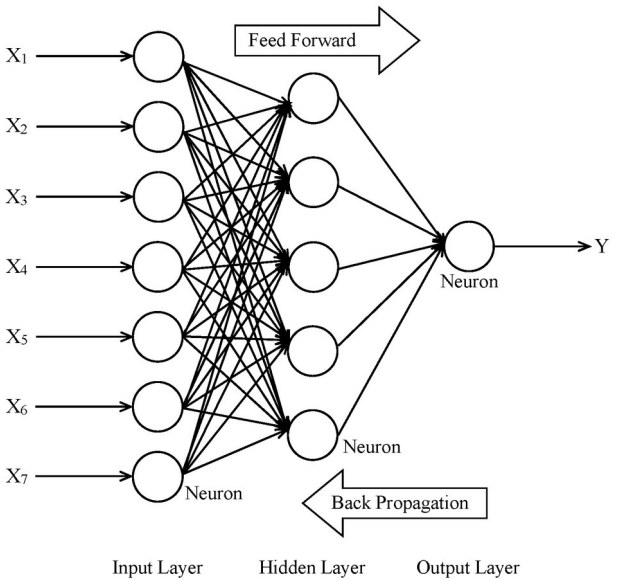
Notes on gamma / pi0,... separation

- With existing ECAL reconstuction chain, your first step could be to try to separate pi0 and gamma by using isolation criteria, and demand that high percentage of shower energy is contained in small area around tower (crystal) that triggered storage of an event.

- Next you can find more about shower shape in ECAL, and find some useful features of that shape, and apply cuts on those or use all features as input variables in Artificial Neural Network, like single layer feed forward ANN.

- The TMVA with collected multivariate methods appeared, including very useful Busted Decision Trees method.

- Currently, you do not need reconstruction, nor shower shape (or other) features, but let Convolutional Neural Network to train (and test) using events in form of whole surface image of ECAL, (and other detectors).



One hidden layer

Convolutional Neural Networks

Image depth: can be RGB channels, time slices...

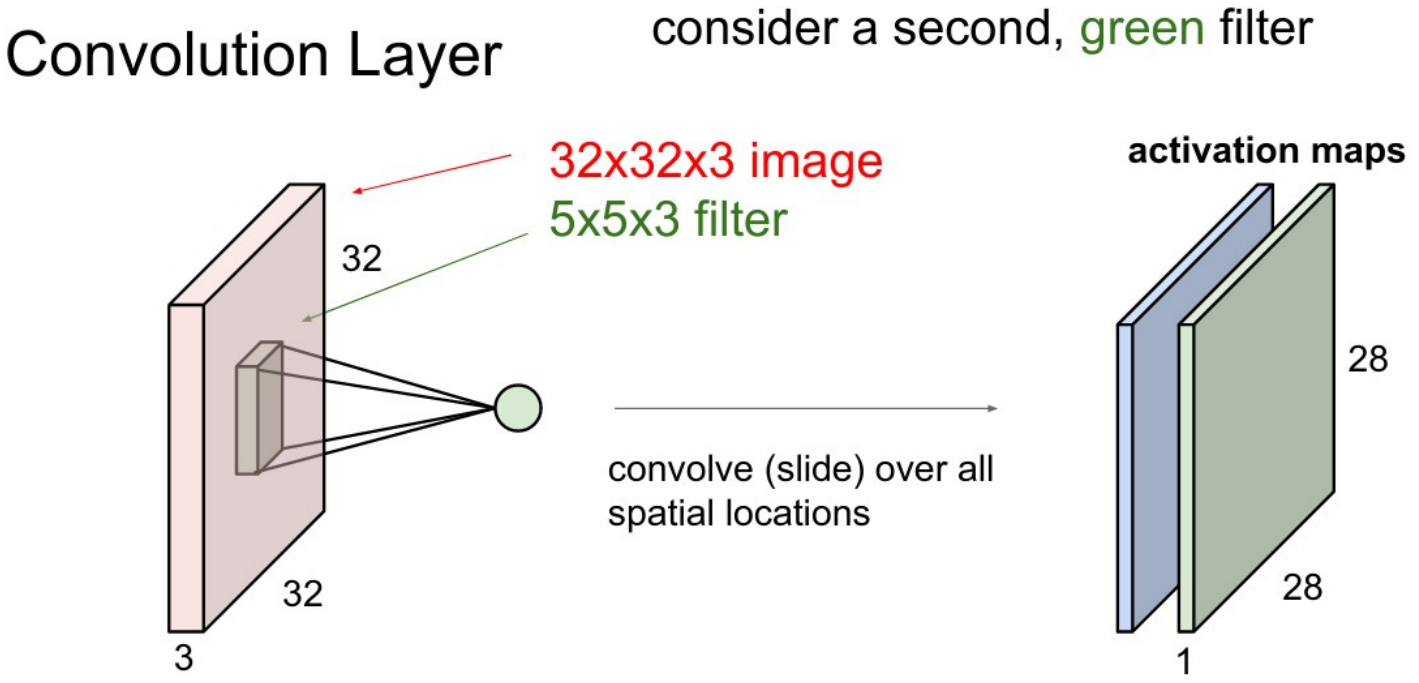
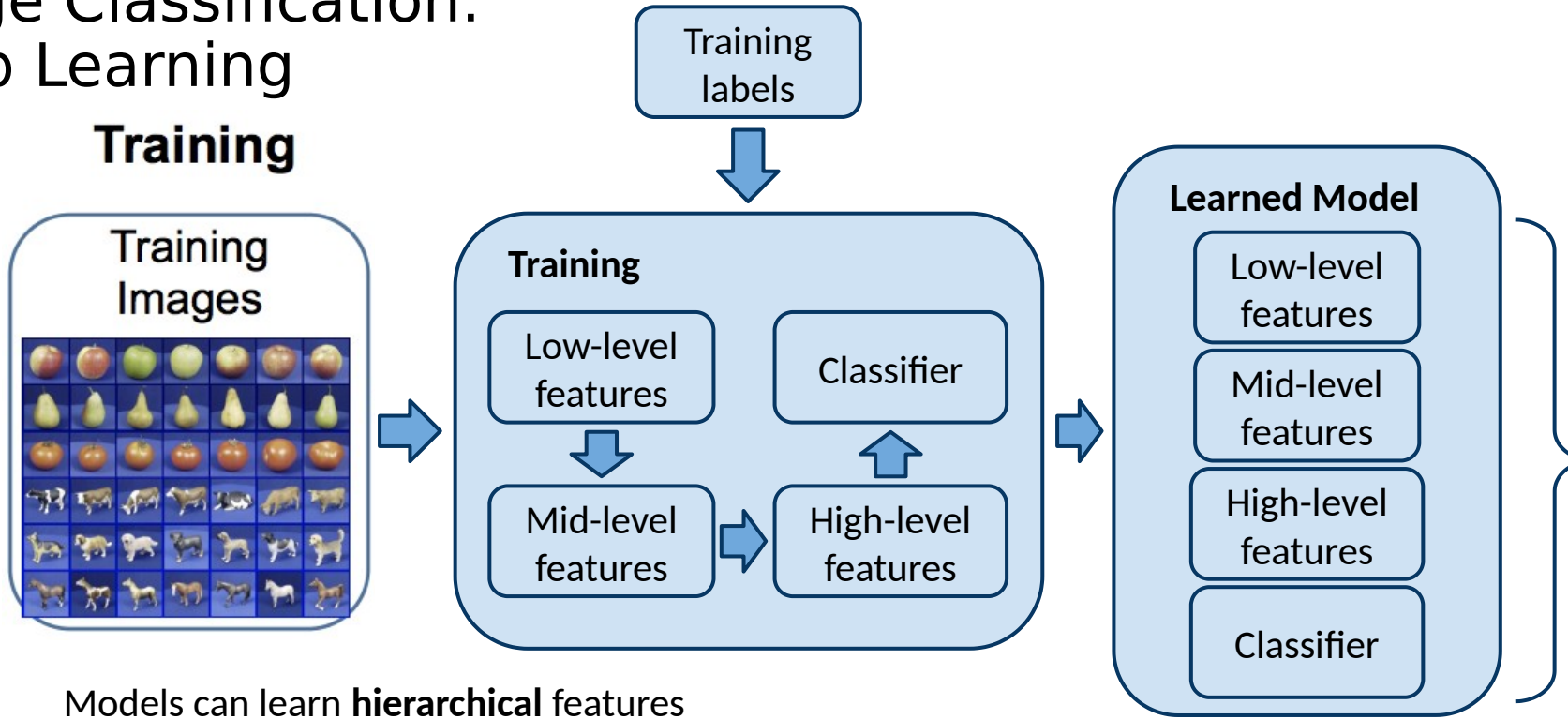


Image Classification: Deep Learning



Models can learn **hierarchical** features

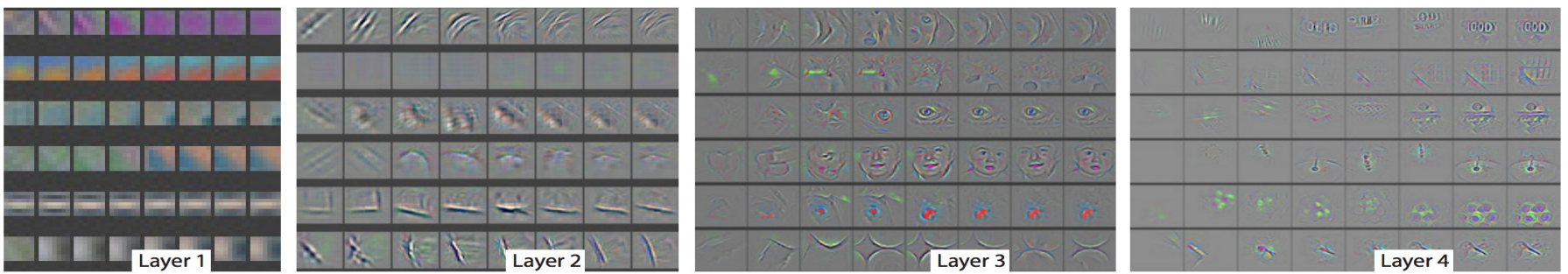


Figure credit: Zeiler and Fergus, "Visualizing and Understanding Convolutional Networks", ECCV 2014

IMAGENET Large Scale Visual Recognition Challenge

Year 2010

NEC-UIUC

Dense grid descriptor: HOG, LBP

Coding: local coordinate, super-vector

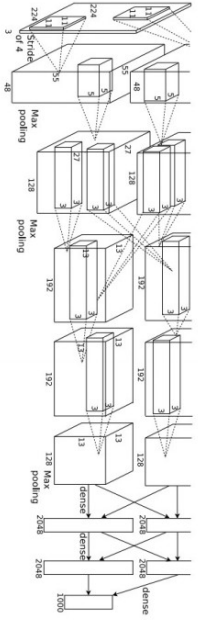
Pooling, SPM

Linear SVM

[Lin CVPR 2011]

Year 2012

SuperVision



[Krizhevsky NIPS 2012]

Year 2014

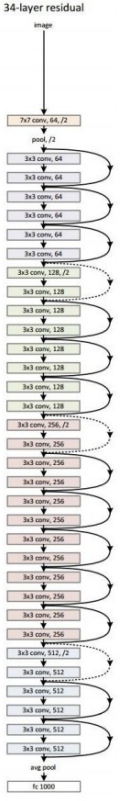
GoogLeNet

VGG

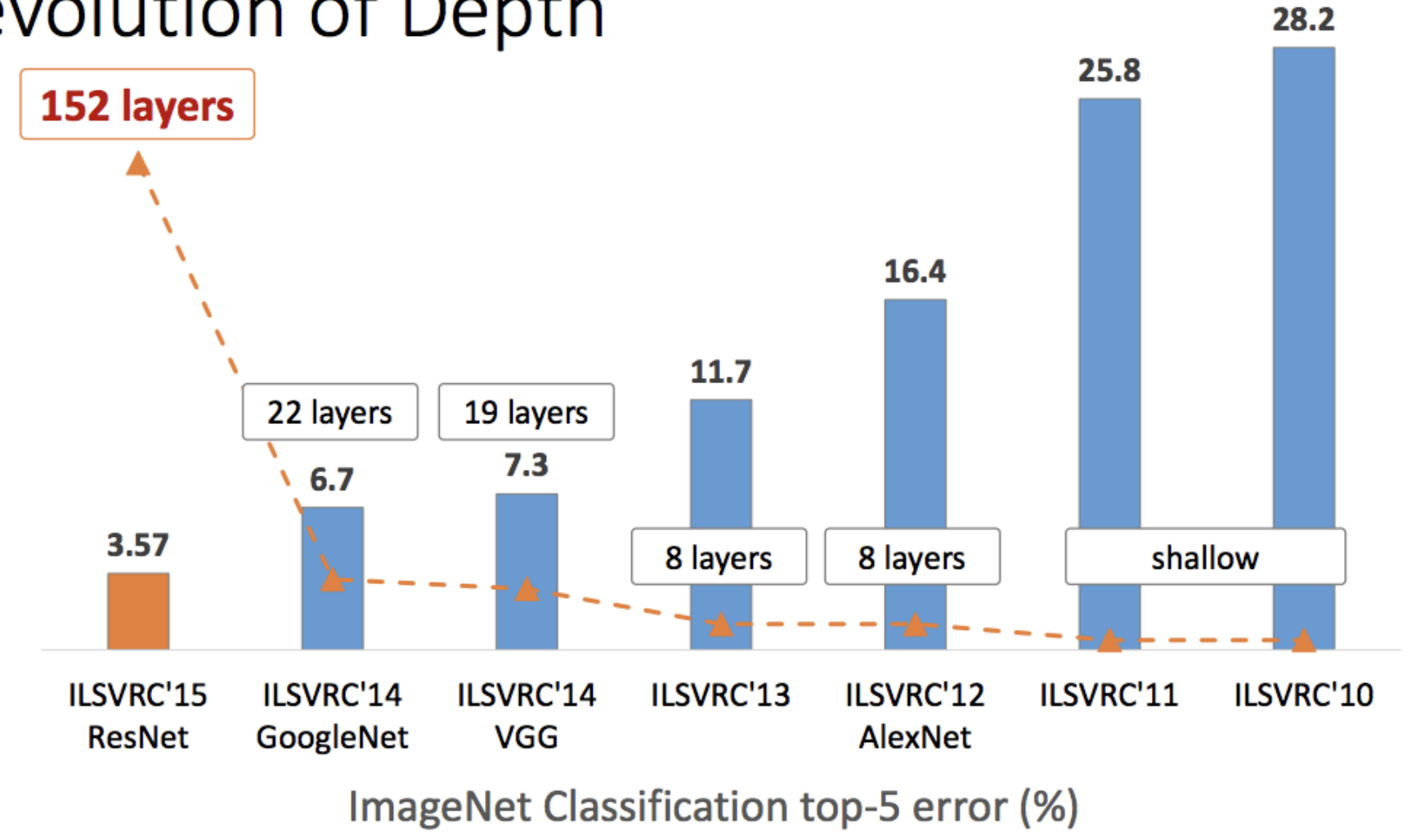
[Szegedy arxiv 2014] [Simonyan arxiv 2014]

Year 2015

MSRA



Revolution of Depth



Usage of CNN in HEP

- Convolutional Neural Network for high p_T jet tracking (CMS 2020)

- Tagging Hadronically Decaying Top Quarks with Deep Neural Networks PhD thesis 2019

- Implementation of Deep Neural Networks for the Level 1 Trigger system of the future High-Granularity Calorimeter (HGCAL) PhD thesis CMS October 2020

- End-to-End Event Classification of High-Energy Physics Data

  End-to-end approaches can be used for event classification to learn directly from detector-level data in a way that is completely independent of the high-level physics reconstruction

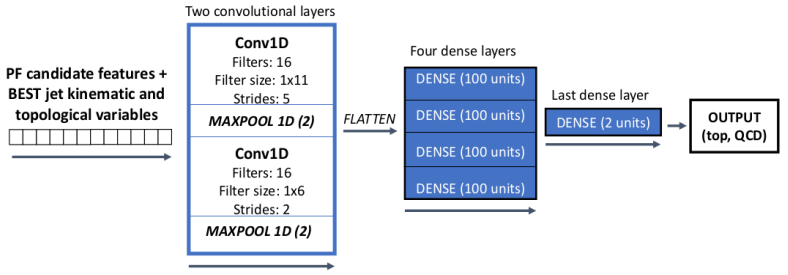
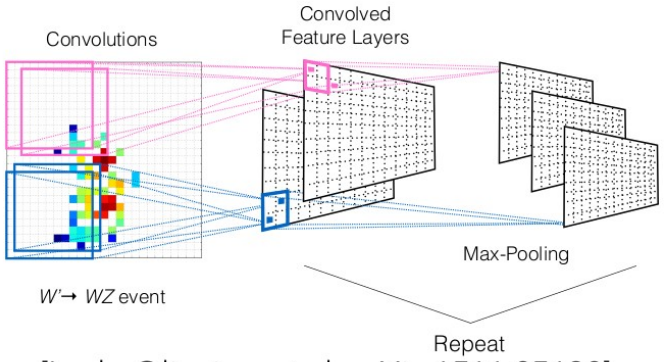


FIGURE 5.1: A schematic showing the general architecture of the PF+BEST neural network. This architecture was modified in a few tests, but this base network, with two convolutional layers, four "main" dense layers, and one final dense layer, was the primary structure used for training and testing.

- Many groups have also tried to apply machine learning to aid in the solution of this problem, such as
  - Convolutional neural networks using an analogy between calorimeters and images [arXiv:1407.5675, arXiv:1511.05190, arXiv:1704.02124]
  - Recursive neural networks built upon an analogy between QCD and natural languages [arXiv:1702.00748]



[L. de Oliveira, et al. arXiv:1511.05190]

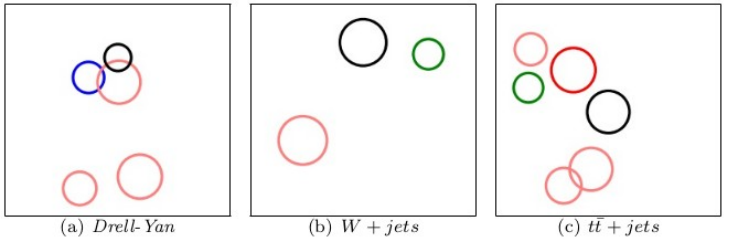
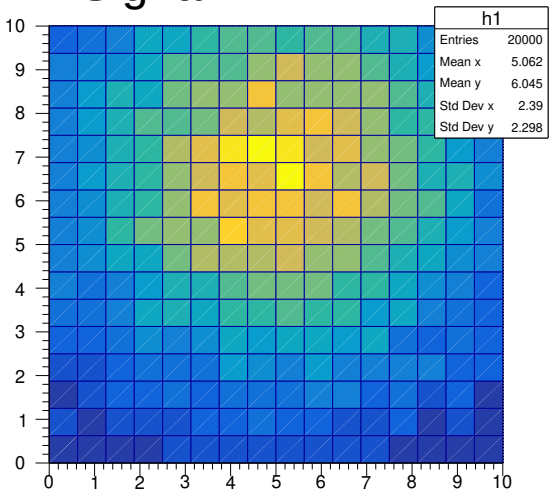
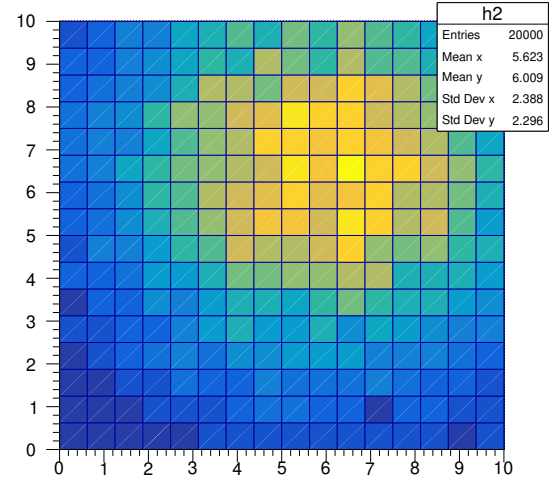


Fig. 6 Examples of images corresponding to the three different classes of collisions being classifed. The x-axis depicts the pseudorapidity η while the y-axis depicts the azimuthal angle φ.

Signal h1



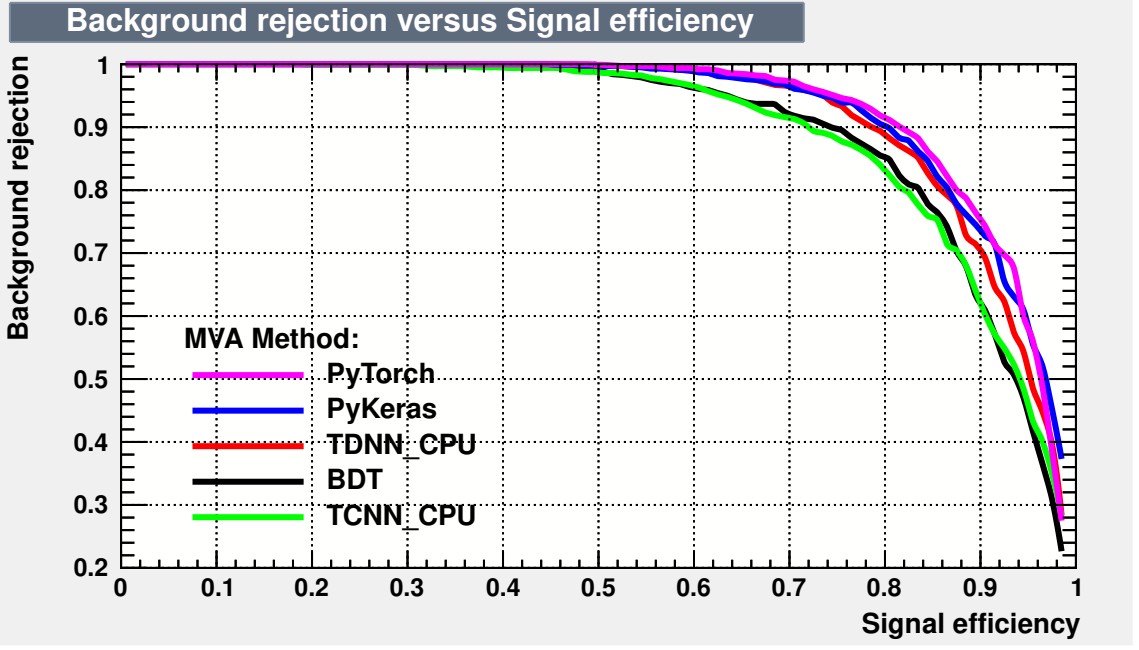
Background h2



My setup for NN training: Centos 8, openblas, python3.8, pip3.8, tensorflow, torch, root_v6.24, nvidia graphical card – still not used (did not get cudNN).

Setup for MC simulations: Centos 7, spdroot 4.1.0

TMVA TensorFlow PyTorch



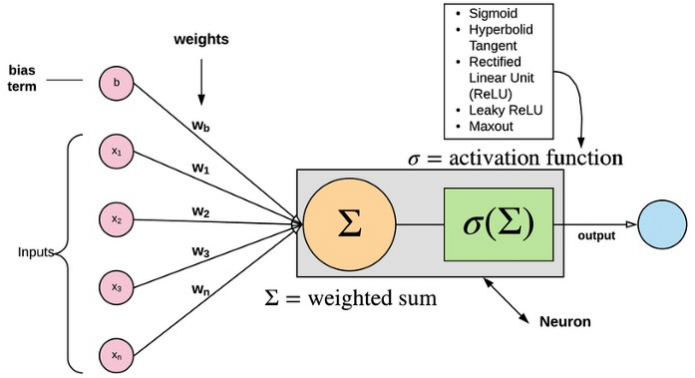
Optimize Neural network layout!

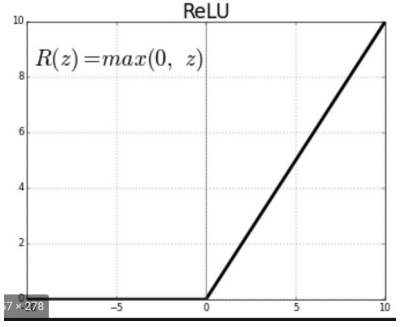
TMVA_**DNN**_CPU

Layout=DENSE|100|RELU,**BNORM**,DENSE|100|RELU,**BNORM**,DENSE|100|RELU,**BNORM**,DENSE|100|RELU,**DENSE|1|LINEAR**

DEEP NEURAL NETWORK: Depth = 8 Input = (1, 1, 256) Batch size = 100 Loss function = C

Layer 0	DENSE Layer:	(Input = 256 , Width = 100)	Output = (1 , 100 , 100)	Activation Function = Relu
Layer 1	BATCH NORM Layer:	Input/Output = (100 , 100 , 1)	Norm dim = 100	axis = -1
Layer 2	DENSE Layer:	(Input = 100 , Width = 100)	Output = (1 , 100 , 100)	Activation Function = Relu
Layer 3	BATCH NORM Layer:	Input/Output = (100 , 100 , 1)	Norm dim = 100	axis = -1
Layer 4	DENSE Layer:	(Input = 100 , Width = 100)	Output = (1 , 100 , 100)	Activation Function = Relu
Layer 5	BATCH NORM Layer:	Input/Output = (100 , 100 , 1)	Norm dim = 100	axis = -1
Layer 6	DENSE Layer:	(Input = 100 , Width = 100)	Output = (1 , 100 , 100)	Activation Function = Relu
Layer 7	DENSE Layer:	(Input = 100 , Width = 1)	Output = (1 , 100 , 1)	Activation Function = Identity





Optimize Neural network layout!

TMVA_**CNN**_CPU

InputLayout: "1|16|16" [The Layout of the input]

Layout: "**CONV|10|3|3|1|1|1|1|RELU**,**BNORM**,**CONV|10|3|3|1|1|1|1|RELU**,MAXPOOL|2|2|1|1,RESHAPE|FLAT,**DENSE|100|RELU**,DENSE|1|LINEAR" [Layout of the network.]

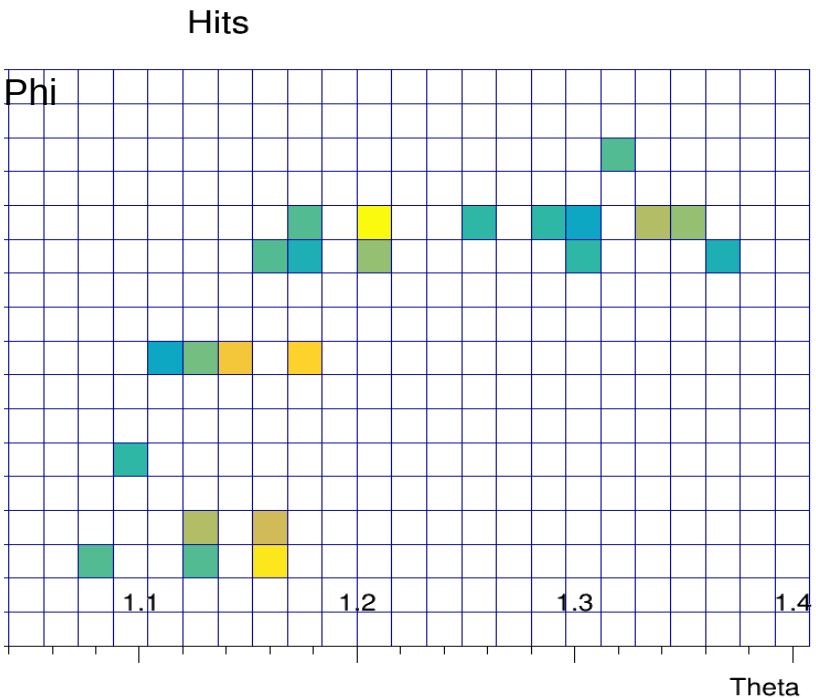
Layer (type)	Output Shape	Param #
reshape (Reshape)	(None, 16, 16, 1)	0
conv2d (Conv2D)	(None, 16, 16, 10)	100
batch_normalization (BatchNo	**(None, 16, 16, 10)**	**40**
conv2d_1 (Conv2D)	(None, 16, 16, 10)	910
max_pooling2d (MaxPooling2D)	(None, 15, 15, 10)	0
flatten (Flatten)	(None, 2250)	0
dense (Dense)	(None, 256)	576256
dense_1 (Dense)	(None, 2)	514

Total params: **577,820**
Trainable params: 577,800
Non-trainable params: 20

MC Simulation:

- using latest spdroot-4.1.0.
- first step in preparation of data for training of NNs is using isotropic generator to produce samples of single gamma and pi0 in central barrel region. Continue training with ECAL images of more realistic events. Use full simulation. ECAL reconstructed clusters for trainning.

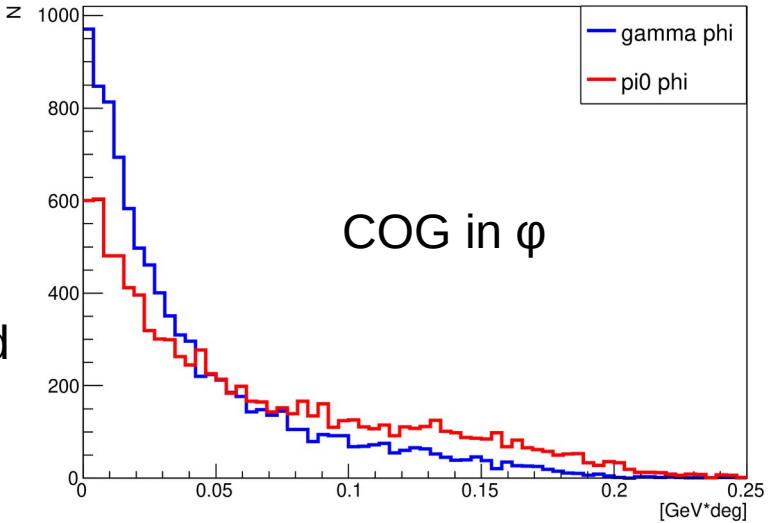
Event 1 particle_energy 0.868217
particle_pos -0.19639 0.734603 **clusterId 5** cluster_energy 0.890704
is_barrel 1 is_endcap 0 **No_cells 9** cells:
cell 44 **hit_energy** 0.16364 **hit_pos Phi** 1.14641 **Theta** 1.75628
cell 60 **hit_energy** 0.0125859 **hit_pos Phi** 1.04991 **Theta** 1.66021
cell 46 **hit_energy** 0.000327415 **hit_pos Phi** 1.04991 **Theta** 1.72831
cell 47 **hit_energy** 0.0444611 **hit_pos Phi** 1.09835 **Theta** 1.75622
cell 48 **hit_energy** 0.0108021 **hit_pos Phi** 1.09835 **Theta** 1.78859
cell 49 **hit_energy** 0.00466598 **hit_pos Phi** 1.14641 **Theta** 1.78865
cell 55 **hit_energy** 0.0117995 **hit_pos Phi** 1.45055 **Theta** 0.798613
cell 56 **hit_energy** 0.0659235 **hit_pos Phi** 1.45055 **Theta** 0.780733
cell 133 **hit_energy** 0.00785588 **hit_pos Phi** 1.45055 **Theta** 0.763627



Possible FC ANN variables

Shower Shape variables

8GeV, isotropic, central ECAL barrel region



Plots are indicative and not optimised

Fully connected feed forward, back propagation neural network with one hidden layer

Crystals' energy sums

$$p_4 = \frac{S_1}{S_9},\ p_5 = \frac{S_9 - S_1}{S_{25} - S_1},\ p_6 = \frac{S_4}{S_{25}}$$

$$p_8 = S_{6\text{-}ratio},\ p_9 = \frac{S_6}{S_9},\ p_{11} = \frac{M_2 + S_1}{S_9},\ p_{12} = \frac{M_2 + S_1}{S_4}$$

$$p_1 = \left|x_{cog}\right|_{25} = \left|\sum_{i=1}^{25} E_i x_i^{rel}\right|,\ p_7 = \left|y_{cog}\right|_{25} = \left|\sum_{i=1}^{25} E_i y_i^{rel}\right|$$

x^{rel} and y^{rel} are crystal's coordinates relative to S_1

Covariances

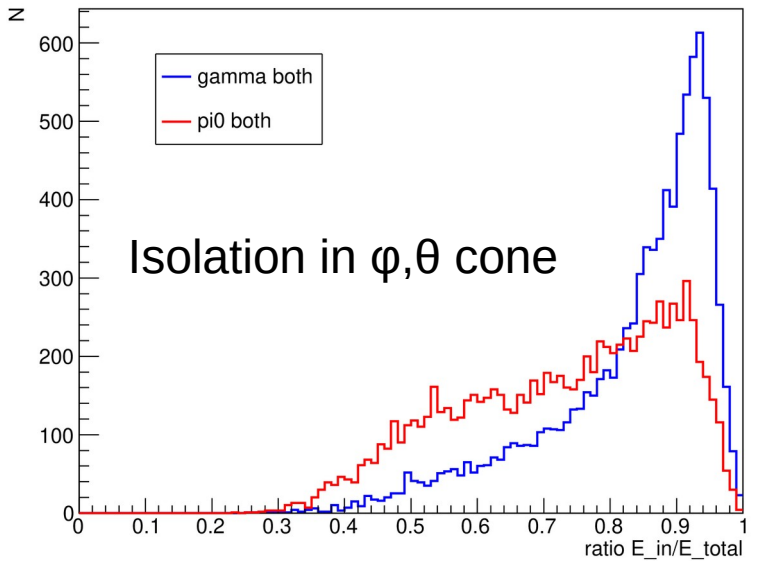
$$p_2 = \frac{\sigma_{\eta\eta}}{0.0004},\ p_3 = \frac{\sigma_{\varphi\varphi}}{0.001},$$

$$\sigma_{xy} = \frac{\sum_{c=1}^{25}(x^c - < x >)(y^c - < y >)\omega_c}{\sum_{c=1}^{25}\omega_c}$$

$$x, y = \{\eta, \varphi\} \qquad \omega_c = MAX(0., \omega_0 + \log(E_c / S_{25}))$$

$$p_{10} = \frac{\lambda_-}{\lambda_+}$$

$$\lambda_\pm = \frac{\sigma_{\eta\eta} + \sigma_{\varphi\varphi} \pm \sqrt{(\sigma_{\eta\eta} - \sigma_{\varphi\varphi})^2 + 4\sigma^2_{\eta\varphi}}}{2}$$



Software for translating neural networks to hardware.

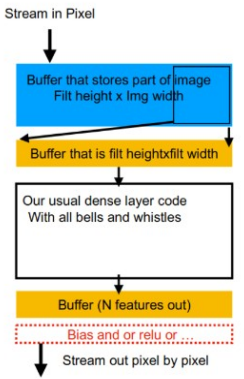
Need to follow progress of this kind of software, and look at <u>constrains</u> for memory.

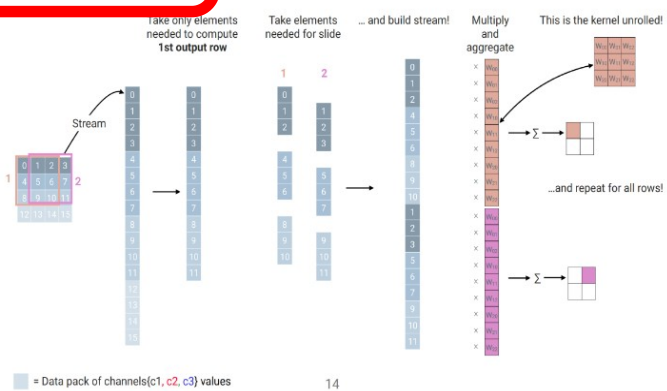
A package for machine learning inference in FPGAs. We create firmware implementations of machine learning algorithms using high level synthesis language (HLS). We translate traditional open-source machine learning package models into HLS that can be configured for your use-case!

- **hls4ml** is a package for translating neural networks to FPGA firmware for inference with extremely low latency on FPGAs
  - https://github.com/hls-fpga-machine-learning/hls4ml
  - https://fastmachinelearning.org/hls4ml/
  - `pip install hls4ml`

Coming Soon

- A few exciting new things should become available soon (this year):
  - Intel Quartus HLS & Mentor Catapult HLS 'Backends'
  - Convolutional Neural Networks
    - Much larger models than we've supported before
  - Recurrent Neural Networks
  - More integrated 'end-to-end' flow with bitfile generation and host bindings for platforms like Alveo, PYNQ





Next steps:

- Improve control of training by using learning transfer (using network parameters from previous training as starting point) This way we can move from simpler events training to more complex ones:
    - We can use single particle simulations to initially train a network.
    - Continue training at some other time with realistically simulated events, when they become available.

- On one input image all detectors can be placed, one beside the other or, naturally, each detector image is placed as a new layer, one on top of the other.

- Adding time slices if needed.

- Artificially increasing number of input images:
    for simulated isotropic events: new image – from old image, shifted in phi.

- Implement “middle step” (between classical and CNN based ECAL reconstruction): ANN using shower shape features for cross checks.

- Coordinate with ECAL experts. Follow new developments in FPGA, GPU hardware and software (ml algorithms, translating NN to hardware).

Thank you for your attention!