**Discovering New Physics**

**ATLAS Trigger Project**

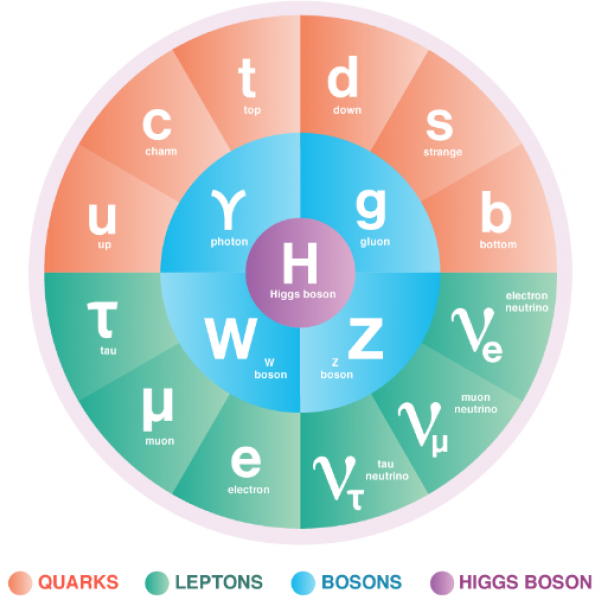
Tim Mironov and Ophir Shurany

**Abstract**. In this paper we propose a new method of predicting the tau-tau decay event vs. jet event using deep learning techniques on a raw calorimeter data. Predicting correctly the event would allow us to implement on-line algorithms in the trigger level and save only events that pass a trigger and delete all others. Next stage of ATLAS project will have higher collision rates, so efficient trigger mechanism is one of the main focuses in the current research.

1. **Introduction**

The Large Hadron Collider (LHC), located at CERN institute in Geneva, is the most powerful particle accelerator on the planet. It is used to collide protons with extremely high velocities and to inspect the results and physical properties of the by-products of the collision. At the point of collisions, the energies get as high as 13TeV with the rate of 40 million collisions every second.

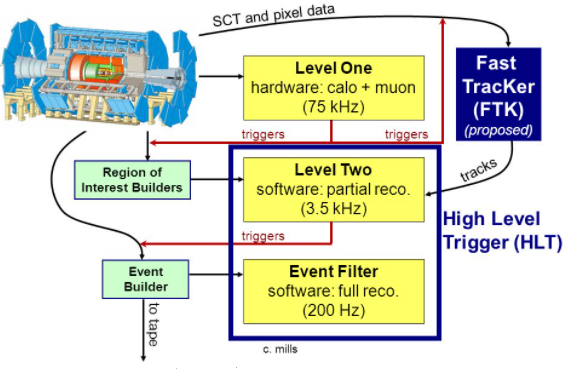
The Higgs boson is a special properties particle with zero spin and large mass. This boson particle was a missing part in the “Standard Model'' of physics and is responsible for the Higgs field and the mass creation of other particles. One of the main LHC’s purposes was the detection of Higgs Particle and measurements of its different interactions and the “new physics” it creates.



[*Figure 1:*](#figur_standard) *The “Standard Model” of Physics*

In 2012 the particle was discovered at CERN and the Nobel prize in Physics was awarded to Peter Higgs and François Englert for the theoretical predictions of the particle’s properties already in 1964. Since 2012 the project has kept upgrading and improving to be able to investigate the different interactions of particles and in particular the interactions and the physical properties of the Higgs Boson.

The experiment is producing large quantities of data due to the high number of collisions taking place inside the LHC. The rate of the data production is much higher than the possible capabilities of saving and analyzing the data in the system, thus some triggering decisions have to be made “on the fly” on whether to save the data or to discard and delete. The ATLAS trigger system was designed specifically to solve this problem and the Tau-Tau trigger is a part of the global system and is responsible for the detection of Higgs decay to two Tau particles [The CMS Collaboration, 2017][[1]](#footnote-0):



[*Figure 2:*](#figur_triggr)*The ATLAS Trigger System*

Tau particles are very heavy and thus decay fast to quarks and gluons. Their decay process leaves traces inside the detectors, which are similar to the background jets traces of particles resulting from other decay processes and residues from collisions.

The triggering decisions are crucial and limited in the amount of computational power they can consume; and the latency they can operate at, since these decisions must run “on-line” and on a predefined hardware. This hardware is decided upon years before the actual runs and the proposed algorithms for the triggering system are examined carefully to fit the specifications.

In this project we continue the work done in the field of novel machine learning and deep learning based algorithms proposed for the 𝛕𝛕 trigger system. We propose two separate methods for processing the raw calorimetric data of the ATLAS sensors, using fully connected and convolutional neural nets.

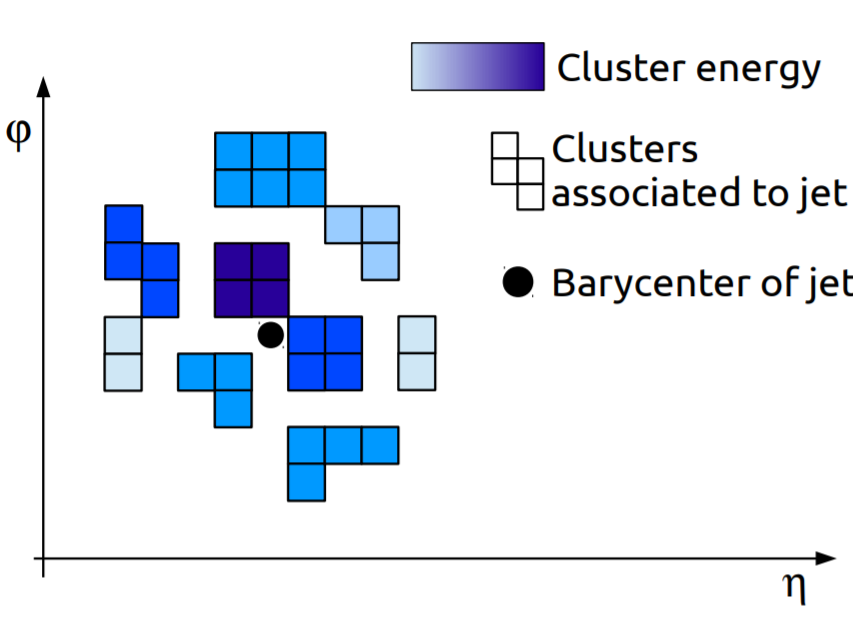
To access the performance of the proposed methods, multiple architecture approaches were **tested and implemented from scratch,** using **TensorFlow/Keras software**. We were able to show similar performance of our implementations to Pytorch implementations demonstrated in other works.

In accordance with the TAU ATLAS research group we focused our work specifically on Tensorflow/Keras to be able to simulate VHDL implementation of the trained models via HLS4ML library. Such simulations were successfully performed and showed a **proof-of-concept for the trigger system based on NN implementation.**

1. **Related Work**

Currently deployed algorithms use a high energy threshold on Tau (~100GeV) and the goal is to “catch” less energetic tau particles as well. These benchmark algorithms use the calorimeters data to decide whether to trigger or not, when the basic principle is the same for all: summing up cells' energy in a specific manner and thresholding based on this summation. In order to decide on the summation patterns research teams study both the signal and the background processes and come up with elaborate shapes to match the 𝛕𝛕 decay event.

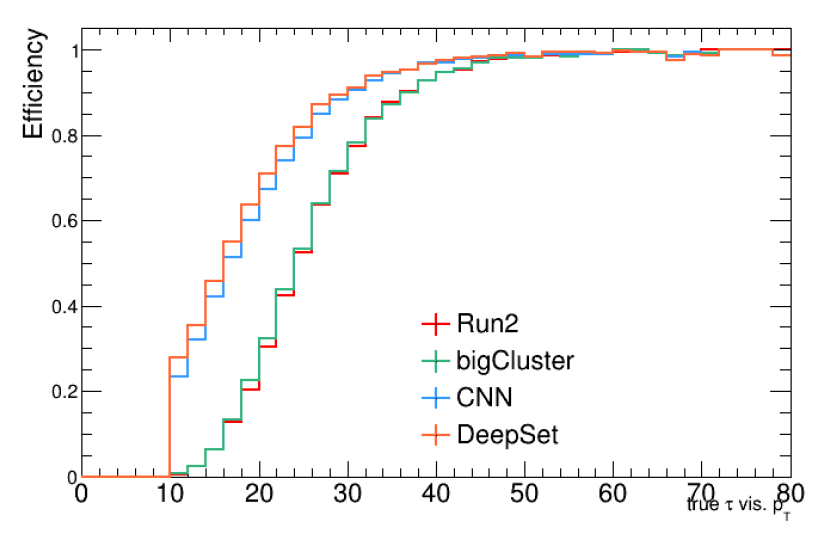
For example, one team of researchers [Christian Limbach, 2014][[2]](#footnote-1) proposed clustering of the energy cells on EM levels, since 𝛕 decays to pions, which in turn immediately decay into photons. These photons would leave a trace in the electro-magnetic layers, while background events would have a stronger hadronic signature left in the HAD sensor:



[*Figure 3:*](#figur_cluster) *Clustering Mechanism for Trigger*

Another approach to the triggering problem is focused on self-learning algorithms that use the raw data provided from the sensors. This approach doesn’t require elaborate research based on physical properties of the events, but rather allows the algorithm to learn the most relevant features from the sensor data directly.

Our project was based in it’s large part on the research done in the TAU ATLAS group. Methods proposed in [Cohen and Etzion, 2020][[3]](#footnote-2) included CNN and Deep-Set implementations on PyTorch; and were tested on the same simulated data that was used in our training process.



[*Figure 4:*](#figur_prevresults) *Results of the previous research*

The results of CNN and DeepSet models were compared vs. the Run2 and BigCluster algorithms with a significant gain in trigger efficiency in the low part.

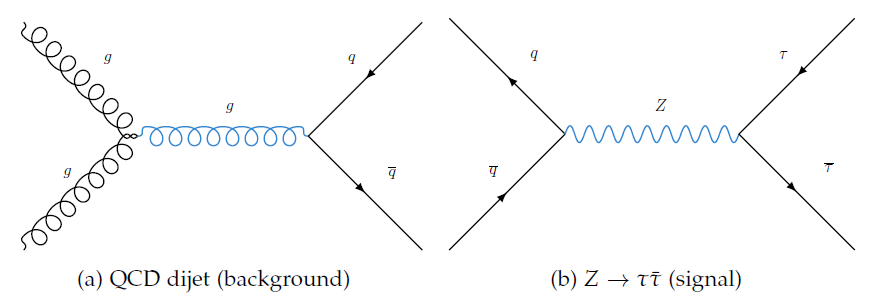
In our work we proposed a range of NN based solutions: from previously not tested Fully Connected net - to complex ResNet solution, similar to one proposed in [Cohen and Etzion, 2020]

We implemented the solutions in Tensorflow/Keras software in accordance with the TAU research team, this way the models created can be tested as a prototype to FPGA implementation. The requirement of TF/Keras comes from the HLS4ML library usage in the ATLAS research groups, which allows fast VHDL prototyping of NN models [Thea Aarrestad et al. (2020)][[4]](#footnote-3).

1. **Data:**

**3.1 Simulated TOBs Data:**

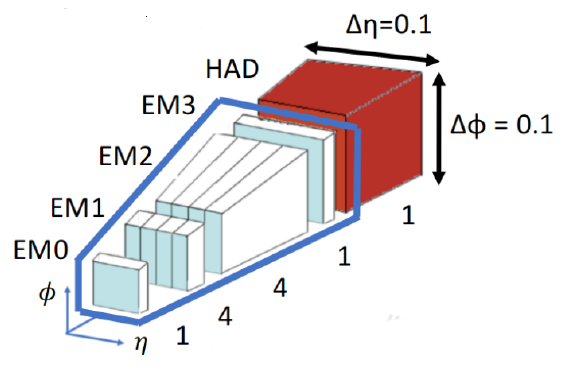
The data which was used for this project is a simulated collisions data from the ATLAS experiment. The simulated data was produced by Athena algorithm and focused around the 𝛕𝛕 decay of the Higgs boson:



[*Figure 5:*](#figur_Feineman1) *Feynman diagram for 𝛕𝛕 decay*

The Athena algorithm enables powerful Monte-Carlo simulations of the collisions and records the passing of the resulting by-products inside the planned detectors. These records are structured as the calorimetric cells, simulating the layered composition of the ATLAS electromagnetic and hadronic sensors.

There are a total of five layers in the L1Calo trigger tower, first four EMx sensors are electromagnetic and the last one HAD is a hadronic calorimeter:

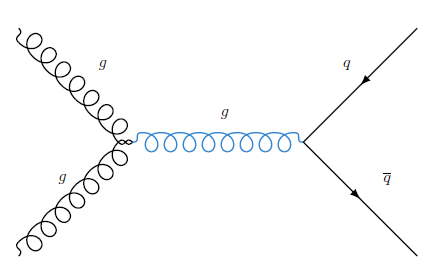


[*Figure 6:*](#figur_triggertower) *Trigger tower sensors*

The data recorded in the calorimetric cells is then processed with CERN proprietary data-processing software ROOT. First in order to zoom into the regions of interest, similar to the pre-trigger processing in ATLAS; and then to extract and form trigger objects (TOBs) which are represented by the raw cells of that region, plus additional measured properties of the event, such as (momentum of the event) for example.

TOB layers have different granularity - with EM0, EM3 and HAD being coarse layers of 3x3; and EM1, EM2 being finer layers of 12x3. Thus in total every event is a vector of 99 energy cells:

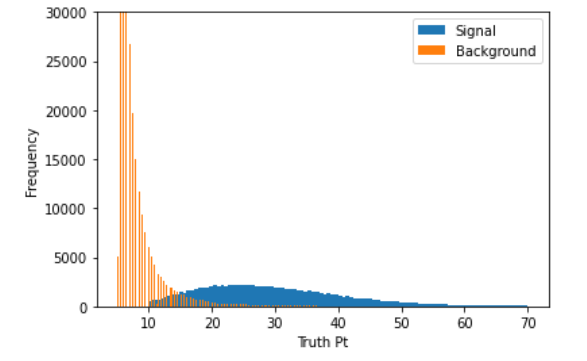
The simulated data contains 129,482 such simulated 𝛕𝛕 decay events. Since the trigger algorithm is tested as a detector of the actual 𝛕𝛕 events against background “noise” events, another 372,619 events of QCD “jets” (quantum chromodynamics) were generated to serve that purpose. These “jets” represent a hard scattering process with high transverse momentum levels and thus can be mistakenly classified as 𝛕𝛕 decay event with existing algorithms:



[*Figure 7:*](#figur_feinmanjet) *Feynman diagram for gluonic/quark jets*

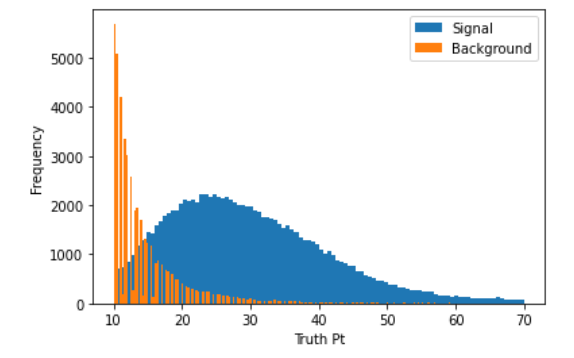
**3.2 Pre-processing with :**

The joint distribution of the 𝛕𝛕 event and the QCD jets represented by 502,101 samples was inspected against and outliers with were dropped, representing less than half of the percent of the data. Further inspection of the data reveals that background data of can be dropped as well:



[*Figure 8:*](#figur_pt1) *Initial Histogram of Signal vs. Background against levels*

Signal events for are non-existent and this drop equalizes the ratio of the signal against background to 69%. The resulting histogram further reveals the real objective of signal classification in lower levels:



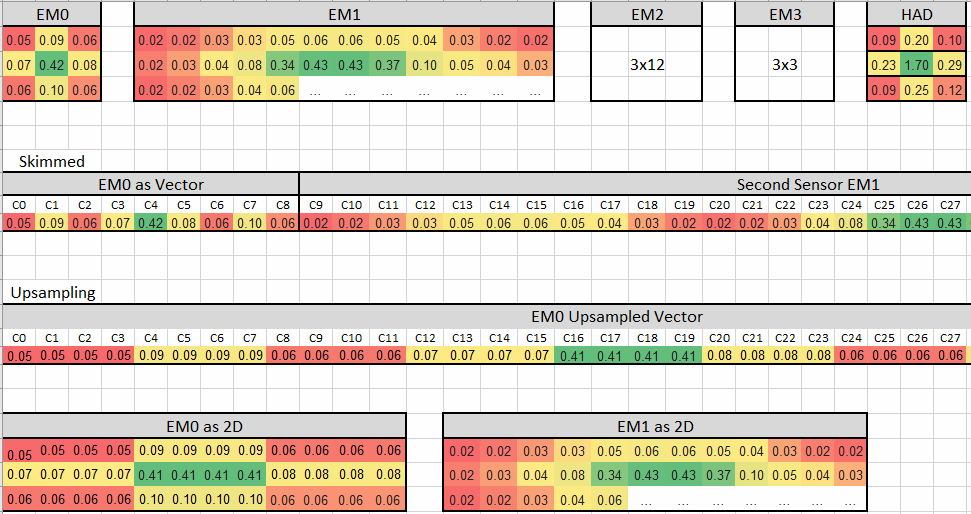
[*Figure 9:*](#figur_histogram2)*Final Histogram of Signal vs. Background against levels*

**3.3 Splitting and Transformations:**

The resulting 183,186 samples we splitted to training and testing datasets with a ratio of 80%-20%. The datasets were shuffled and checked to assure the adequate signal and background representation. The test dataset was saved and later used to evaluate the performance of multiple architectures against the same input distribution.

In order to improve the training process and to address the different energy levels recordings in the TOBs layers, we have normalized the cells readings across every cell deposit. The resulting scaler was saved and used in the evaluation process as well.

In order to test the performance of convolutional neural networks over the dataset, we decided to transform the calorimetric data vector to “image-like” representation of 5-channel matrices. To stack the multiple layers of TOB together, the coarse layers were upsampled to match the 3x12 granularity of the finer layers:



[*Figure 10:*](#figur_upsample) *Multi Channel Stacking*

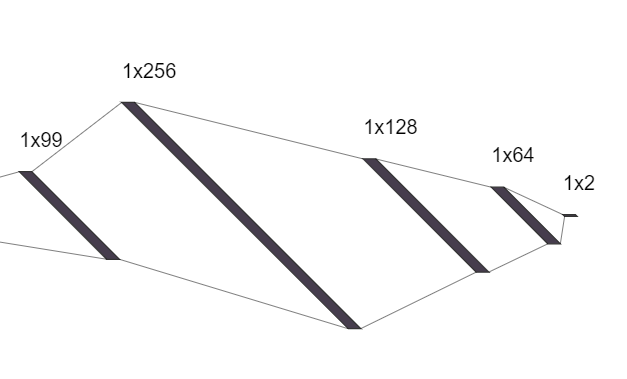
In all the CNN architectures this final representation of 5x3x12 was used, while the fully connected network was supplied with a flattened vector of the data.

1. **Methods**

In general, we have divided our methods into two major approaches, which treat the experimental data from two different perspectives. One approach views the raw calorimetric data as an ordered one-dimensional vector. The other views the same data as an equivalent of a multi-spectral image, where every channel is an additional layer of sensors.

**4.1 Fully Connected Neural Net:**

Due to the computational restrictions of the ATLAS trigger system we intuitively started our research with the first approach and applied a shallow fully connected architecture as the initial method:



[*Figure 11:*](#figur_FCArch) *Fully Connected Architecture*

Our training samples are vectors of and we have tested multiple configurations of two/three hidden layers sequential nets. We have applied a dropout technique to prevent network overfitting with a 0.3-0.5 dropout coefficient between the hidden layers:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Layer 1 | Layer 2 | Layer 3 | Dropout | Best Val Loss |
| Type 1 | 64 | 32 | - | 0.3 | 0.4100 |
| Type 2 | 128 | 64 | - | 0.3 | 0.4072 |
| Type 3 | 64 | 64 | 32 | 0.3 | 0.4085 |
| Type 4 | 128 | 64 | 32 | 0.3 | 0.4060 |
| Type 5 | 256 | 128 | 64 | 0.3 | 0.4033 |

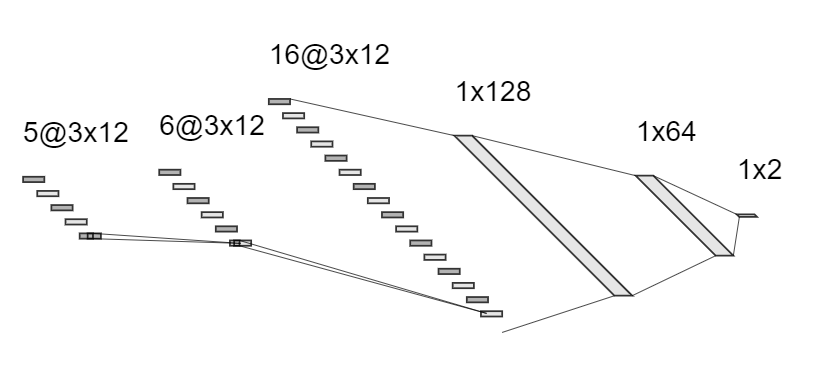
[*Table 1:*](#table_performance) *Performance of the Fully Connected Architectures*

Adam optimizer showed the best performance and we applied a varying 0.001-0.1 learning rate to train all the configurations. We have used a 32 batch size and trained all nets for a maximum of 30 epochs with early (not strict) stoppers, tuned on validation set cross-entropy loss.

**4.2 LeNet Convolutional Neural Net:**

After creating a baseline for the optimized fully connected performance we continued our investigation of the multi-layered calorimeters representation as a multi-spectral image and proposed a rather simplified initial CNN architecture.

We chose the LeNet classical architecture, modified to the input samples we have:



[*Figure 12:*](#figur_Lenet) *Initial LeNet Architecture*

Our input training samples are multi-channel matrices of and we have tested multiple configurations of “LeNet like” architectures:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1st Conv | 2nd Conv | 3rd Conv | Dense 1 | Dense 2 | Dense 3 | Val Loss |
| Type 1 | 6x3x12 | 16x3x12 | - | 128 | 64 | - | 0.4124 |
| Type 2 | 10x3x12 | 20x3x12 | - | 128 | 64 | - | 0.4084 |
| Type 3 | 15x3x12 | 30x3x12 | - | 128 | 64 | - | 0.4091 |
| Type 4 | 10x3x12 | 20x3x12 | 10x3x12 | 128 | 64 | - | 0.4078 |
| Type 5 | 10x3x12 | 20x3x12 | - | 256 | 128 | 64 | 0.4056 |

[*Table 2:*](#table_LenetPerf) *Performance of the Lenet Architectures*

We have used Adam optimizer with varying learning rates. 32 samples were chosen as the training batch size and the training was optimized to use Dropout technique.

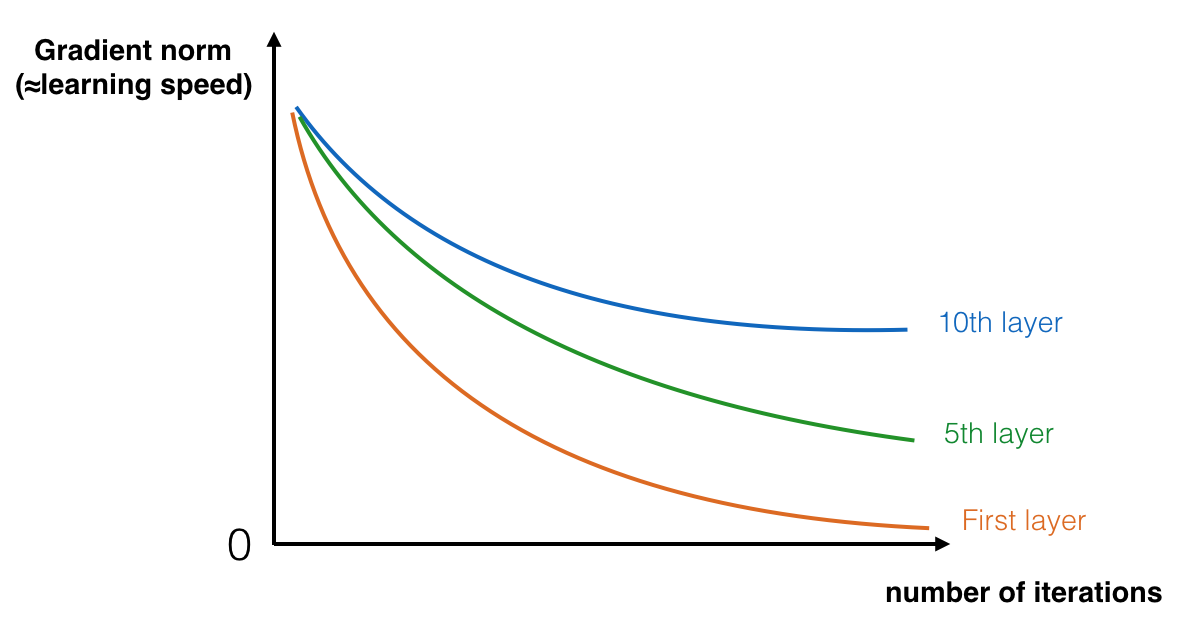
**4.2 ResNet Convolutional Neural Net:**

In theory, very deep networks can represent very complex functions; but in practice, they are hard to train. Residual Networks, introduced by He et al.,allow training much deeper networks than were previously practically feasible.

In recent years, neural networks have become deeper, with state-of-the-art. networks going from just a few layers (e.g., AlexNet) to over a hundred layers.

The main benefit of a very deep network is that it can represent very complex functions. It can also learn features at many different levels of abstraction, from edges (at the lower layers) to very complex features (at the deeper layers). However, using a deeper network doesn’t always help. A huge barrier to training them is vanishing gradients: very deep networks often have a gradient signal that goes to zero quickly, thus making gradient descent unbearably slow.

During training, we might therefore see the magnitude (or norm) of the gradient for the earlier layers decrease to zero very rapidly as training proceeds:



[*Figure 13:*](#figur_gradient) *norm of the gradient for the earlier layers*

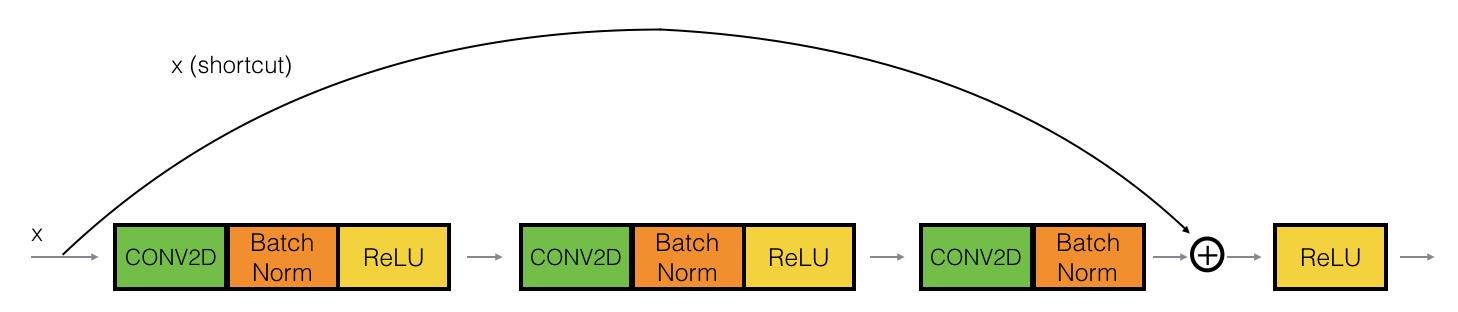
Data transformation

We restored the flatten data [samples,1,99] back to multidimensional images [samples,3,12,5] so that we could work on the database for the RESNET network.

1.2. Detailed ResNet network

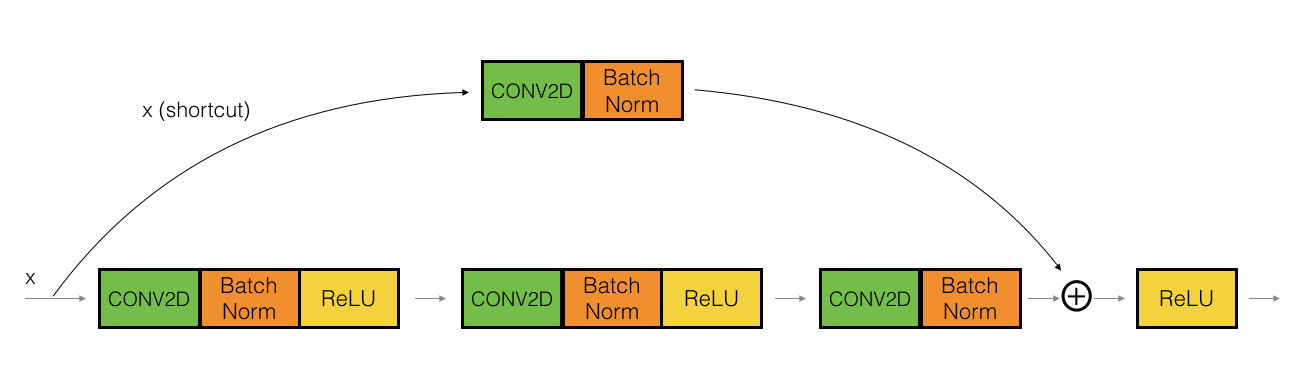
The identity block is the standard block used in ResNets, and corresponds to the case where the input activation (say a[l] ) has the same dimension as the output activation (say a[l+2] ).

In this chapter, we’ll implement a slightly more powerful version of this identity block, in which the skip connection “skips over” 3 hidden layers rather than 2 layers. It looks like this:



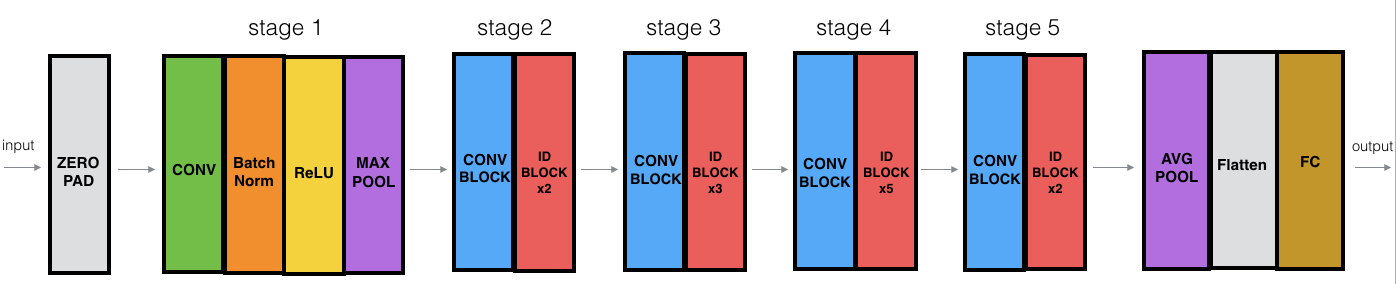
[*Figure 14:*](#figur_identity) *The identity block*

ResNet “convolutional block” is the other type of block. We can use this type of block when the input and output dimensions don’t match up. The difference with the identity block is that there is a Conv2D layer in the shortcut path:



[*Figure 15:*](#figur_convolution) *The convolutional block*

The following figure describes in detail the architecture of this neural network. “ID BLOCK” in the diagram stands for “Identity block,” and “ID BLOCK x3” means we should stack 3 identity blocks together.



[*Figure 16:*](#figur_resnet) *ResNet-50 model*

The details of this ResNet-50 model are:

* Zero-padding pads the input with a pad of (4,2)
* Stage 1:

· The 2D Convolution has 64 filters of shape (1,3)

and uses a stride of (1,1)

· BatchNorm is applied to the channels axis of the input.

* Stage 2:

· The convolutional block uses three sets of filters of size [64,64,256]

· The 2 identity blocks use three sets of filters of size [64,64,256]

* Stage 3:

· The convolutional block uses three sets of filters of size [128,128,512]

· The 3 identity blocks use three sets of filters of size [128,128,512]

* Stage 4:

· The convolutional block uses three sets of filters of

size [256, 256, 1024]

· The 5 identity blocks use three sets of filters of

size [256, 256, 1024]

* Stage 5:

· The convolutional block uses three sets of filters of

size [512, 512, 2048]

· The 2 identity blocks use three sets of filters of size [512, 512, 2048]

* The 2D Average Pooling uses a window of shape (2,2)
* The flatten doesn’t have any hyperparameters or name.
* The Fully Connected (Dense) layer reduces its input to the number of classes using a softmax activation.

We used ADAM optimizer and learning rate=0.005

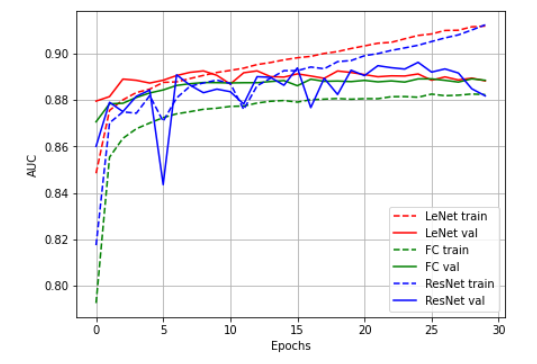
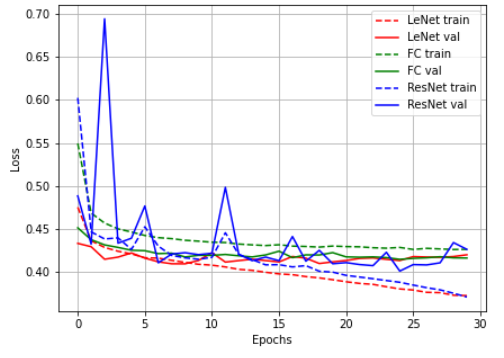
1. **Experiments**

Here to write more aboth HLS4ML

1. **Results & Conclusions**

ADD the PT

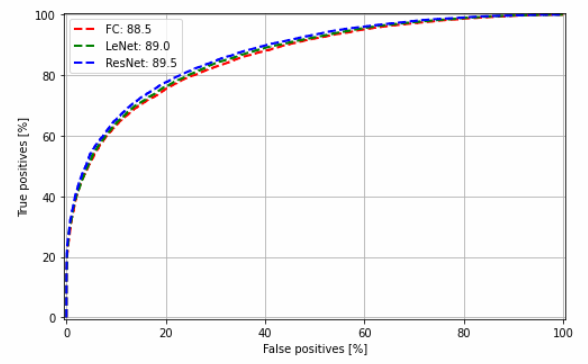
Training results for the loss (categorical cross-entropy) for all three architectures.

Due to the nature of the problem, we decided to present the results for the AUC (area under the curve) as well:

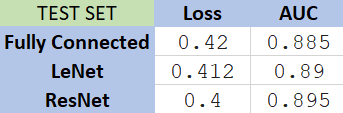
[*Figure 18:*](#figur_auc) *AUC results* [*Figure 17:*](#figur_loss) *Loss results*

Since the efficiency of the trigger system is compared to other algorithms already used in the ATLAS, ROC plot is a good way to look at the behaviour of the classifier for different False Positive rates.

We want to maximize the number of “Signal” samples saved in the experiment and to minimize the “background” irrelevant samples.



[*Figure 19:*](#figur_roc) *False Positive rates*



[*Table 3:*](#table_results) *Loss & AUC final results*

In conclusion, we were able to apply ML and DL techniques learned in the course on real physics problems and data; helping a TAU research team in their project.

We have implemented new architectures using TF (all previous attempts were made using PyTorch) and had a proof-of-concept on deployment of TF models to the Trigger System FPGA.

Proved that “image like” approach to the ATLAS data can yield better results with CNN techniques applied and will be even more relevant with the higher resolution of the next generation of sensor

1. **References**

[1] The CMS Collaboration - *Observation of the Higgs boson decay to a pair of τ leptons with the CMS detector (2017)*

[2] Christian Limbach *- Reconstruction and Identification of Tau Leptons in ATLAS (2014)*

[3] Hadar Cohen, Erez Etzion - *𝛕 Triggering with Deep Learning (2020)*

[4] Thea Aarrestad, Vladimir Loncar, Maurizio Pierini, Sioni Summers - *Fast Convolutional Neural Networks on FPGAs with HLS4ML (2020)*

[5] Taylor Faucett, Jesse Thaler, Daniel Whiteson - *Mapping Machine Learned Physics into a Human-Readable Space (2020)*

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun – [*Deep Residual Learning for Image Recognition (2015)*](https://arxiv.org/abs/1512.03385)

[7] Piotr Migdał -[*Simple diagrams of convoluted neural networks*](https://medium.com/inbrowserai/simple-diagrams-of-convoluted-neural-networks-39c097d2925b)*, Medium (2018)*

1. **Code Appendix**

All the methods, experiments and results are documented and produced using Google Colab and the following list includes all the notebooks with the source code of the project:

[1] ATLAS: Upsample to CNN - *Processing raw sensor’s data to meet CNN specs.*

[2] ATLAS: Fully Connected - *Training process of the FC NN trained on vector data.*

[3] ATLAS: LeNet CNN - *Training process of the Lenet CNN for multi-channel data.*

[4] ATLAS: Preprocessing - *Data preprocessing: Pt filtering and train-test splitting.*

[5] ATLAS: ResNet CNN *- Training process of the ResNet for multi-channel data.*

[6] ATLAS: Test Models *- Evaluation of all models on the same test set, results and graphs.*

[7] ATLAS: Create HLS Model *- Simulation and testing of HLS model from saved NN*

1. The CMS Collaboration - *Observation of the Higgs boson decay to a pair of τ leptons with the CMS detector (2017)* [↑](#footnote-ref-0)
2. Christian Limbach *- Reconstruction and Identification of Tau Leptons in ATLAS (2014)* [↑](#footnote-ref-1)
3. Hadar Cohen, Erez Etzion - *𝛕 Triggering with Deep Learning (2020)* [↑](#footnote-ref-2)
4. Thea Aarrestad, Vladimir Loncar, Maurizio Pierini, Sioni Summers - *Fast Convolutional Neural Networks on FPGAs with HLS4ML (2020)* [↑](#footnote-ref-3)