September 2019

Neuromorphic Computing in High Energy Physics

AUTHOR:
Bartlomiej Borzyszkowski
Gdansk University of Technology, Intel AI

SUPERVISORS:
Maurizio Pierini
CERN

Jean-Roch Vlimant
California Institute of Technology

CERN openlab

ABSTRACT

At particle colliders, more data are produced than what the experiments can store for further analysis. This is why the incoming collisions are processed in real time by a so-called trigger system. At the Large Hadron Collider (LHC), trigger systems are structured as a combination of two hierarchical layers: the first trigger level (or L1 trigger) operates with latency of O(10 μec), selecting the ~100K events/sec to be passed to the second stage. The second stage, called high-level trigger, runs a faster and coarser version of the offline reconstruction, to select the final 1000 events to be stored for analysis. The L1 trigger selection algorithms are deployed as signal processing through electronic circuits, usually implemented though ASICs or FPGAs. The HLT runs on a CPU farm, supposed to be powered by GPUs or FPGAs in the future.

Recently, an effort to deploy neural network on the L1 FPGAs was started, resulting in a Deep-learning-to-FPGA firmware deployment (HLS4ML) being developed specifically for low-latency inference. This is only the first step into an R&D program aiming to explore new computing architecture to process LHC data in real time. In this respect, we are interested to explore emerging technologies for fast inference. Neuromorphic chips are clearly part of this R&D program, being a natural environment to implement Spiking Neural Networks (SNNs).

Neuromorphic computing is an interesting candidate for signal processing at the High-Luminosity LHC, the next stage of the LHC upgrade (scheduled to start in 2025). For HL-LHC, existing particle detectors will be upgraded, that will allow to take a time-sequence of snapshots for a given collision. This additional information will allow to separate the signal belonging to the interesting collision from those generated parasitic collisions occurring at the same time (in-time pileup) or before/after the interesting one (out-of-time pileup). By powering the LHC real-time processing with SNNs, one could be able to apply advance and accurate signal-to-noise discrimination algorithms in real time, without affecting the overall system latency beyond the given tolerance.

We propose to investigate the potential of SNNs deployed on neuromorphic chips as a technological solution to increase the precision of the upgraded CMS detector for HL-LHC. This includes the characterization of a particle type (classification) based on the recorded features or image representation. These information can be used to solve a classic LHC problem and eventually lead to determine whether a particle belongs to the interesting collision or to one of the parasitic events. The study is based on simulations and real data collected during tests at particle beams, collected in the context of the upgrade studies for the CMS detector.

TABLE OF CONTENTS

1. INTRODUCTION

In recent years deep learning technologies have contributed to many spectacular successes of Artificial Intelligence (AI). Architectural principles behind Deep Neural Networks (DNNs) were strongly inspired by neuroscience, however at the implementation level, widely used models have only marginal similarities between the brain-like calculations [1]. One of the main reasons is absence of the time factor in operations of Artificial Neural Networks (ANNs). Computation in biology is whereas performed thanks to the asynchronous *spikes,* what allows to distinguish the appearance of some characteristic events by the level of action potential in each neuron. This phenomenon was utilized in neuromorphic computing using Spiking Neural Networks (SNNs) [2], which mimic the functioning of human brains and allow to bridge a gap between the artificial and biological intelligence.

Operations in SNNs are complex and their simulation involves factors such as fine-time-grained computation of the action potential of all neurons, signal propagation or extracting gradient information from discrete events in the learning process [3]. Because of these difficulties, SNNs achieve the most promising results on the dedicated neuromorphic hardware. They exhibit favourable properties such as low power consumption, fast inference, and an event-driven processing of information in a massively parallel fashion. In this project we participate in the Intel Neuromorphic Community and use the Loihi chip [4], which is considered a current state-of-the-art in the neuromorphic hardware. Due to a lack of publicly available training algorithms, losses during conversion processes of networks and mostly conventional frame-based datasets [5], SNNs do not reach the same levels of accuracy as ANNs on most tasks. We expect that stronger development of technologies in this domain will allow to overcome these limitations and take an advantage of various benefits of neuromorphic computing.

Conventional machine learning approaches and SNNs should not be considered as two solutions to the same classes of problems. It is possible to identify and exploit their task-specific advantages and in some applications also connect the work of both. Deep SNNs offer great opportunities to be used with new types of sensors, and thanks to their event-based processing can be the perfect candidates also in High Energy Physics (HEP) [6]. We find their significant potential in this domain and present the first successful application at CERN, the European Organization of Nuclear Research. Furthermore, we compare and discuss different software approaches, basing on selected examples. Finally, we develop an own model of SNN and successfully solve a Jet Tagging Task. Our results are also compared with currently used DNNs and debated in terms of different computation possibilities. In conclusion, we discuss the future opportunities of neuromorphic computing in HEP and present several ideas for further applications in this domain.

2. SPIKING NEURAL NETWORKS

Structure of SNNs is asynchronous, which means that their neurons do not fire at each propagation cycle, as it happens with typical multi-layer ANNs [7]. Instead, neurons are activated only when a potential of their membrane reaches a specific value (Fig. 1). When a neuron fires, it generates a signal that travels to other neurons which, in turn, increase or decrease their potentials in accordance with this signal [8].

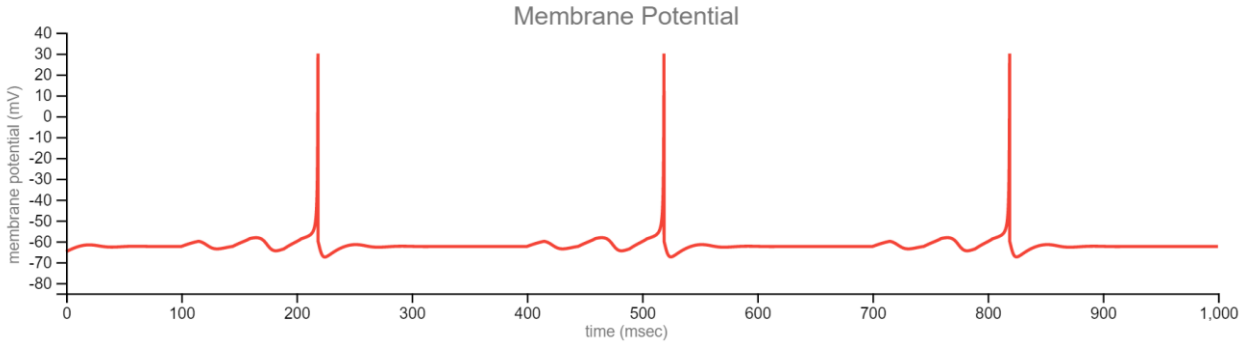

Figure 1. *Membrane potential of a single spiking neuron. The neuron is firing when the action potential achieves a specific threshold. After activation the signal returns to the low, regular value.*

SNNs allow to perform a bio-inspired learning (weight modification) that depends on the relative timing of spikes between pairs of directly connected neurons. The spike trains are represented formally by sums of Dirac delta functions (Fig. 2).

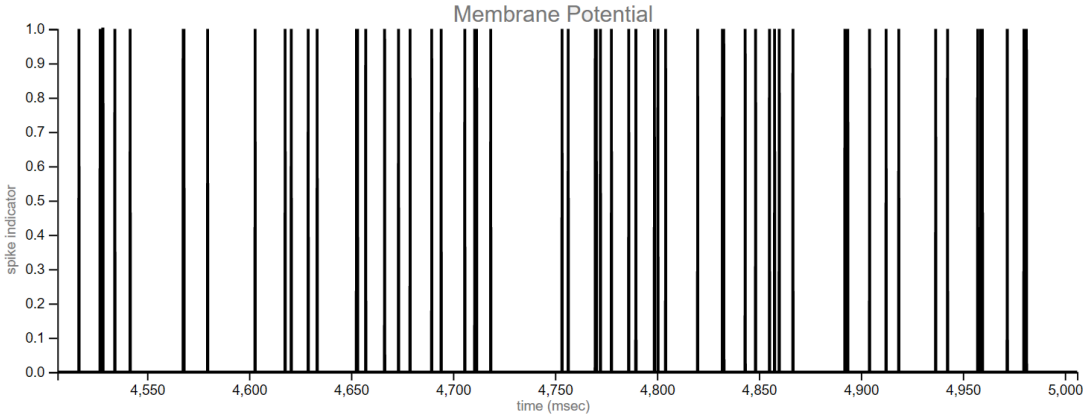

Figure 2. *Neural response of spike trains that shows a discrete output of different neurons over time.*

Because spiking topologies are not differentiable, we cannot train SNNs using gradient descent. Numerous methods to extract the learning information from discrete outputs are being developed [9], however this approach still remains a field of exploration. To circumvent the problem, conventionally trained DNNs can be converted into deep SNNs by adapting weights and parameters of the spiking neurons [10]. Moreover, neuroscientists have identified many variants of direct learning rules that fall under the term spike-timing-dependent plasticity (STDP) [11]. The key feature of this process is an adjustment of strengths of the connections between pre- and post-synaptic neurons, basing on relative timing of a particular neuron's output and input action potentials.

3. LOIHI – INTEL NEUROMORPHIC RESEARCH COMMUNITY

Simulating SNNs on von Neumann machines is typically inefficient due to an asynchronous activity of the networks. Ideally, each spiking neuron is an own processor without central clock, which is the design principle of neuromorphic hardware. To implement SNNs in a functional way, we use Loihi chip (Fig. 3), which is a fifth-generation of a self-learning neuromorphic processor introduced by Intel Labs in 2017.

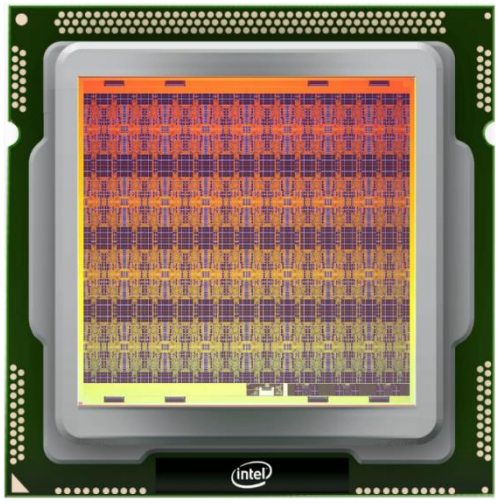

Figure 3. *Intel Loihi neuromorphic hardware*

Loihi includes a total of 130,000 neurons fabricated in 14nm process technology and supports both, training and inference of SNNs in a very efficient asynchronous way [12]. Because the hardware is optimized specifically for spiking topologies, it is also characterized by extremely low power consumption and impressive performance. Thanks to the highly parallel structure, sparse communication, and possibilities of scaling, Loihi is a big step forward in the work on neuromorphic hardware.

Access to the scalable Loihi-based infrastructure is provided thanks to the cloud platform being part of Intel Neuromorphic Research Community (INRC). This collaborative research group brings together various scientific and industry entities for further development of neuromorphic computing technologies. Members of the community agree to share the outcomes of their research and participate in common scientific activities.

Until now, Intel has developed a series of systems based on Loihi which scale to large number of neurons and synapses [13]. The main platform used in the project is named Nahuku and carries up to 32 bare Loihi chips. Nahuku is not an independent processor but a scalable Arria10 FPGA (host) expansion board. It communicates with a standard “super host” (CPU) which can be used to send commands to the board and to the management core on the chips themselves. Connection with the super host is available through INRC using the dedicated research cloud.

4. SOFTWARE OVERVIEW

Brain-inspired computing systems require substantial and very complex software for various aspects of their operation. Tools which are commonly used in conventional machine learning usually do not satisfy these needs, hence dedicated software packages play a key role in computations with SNNs. Rapid developments in this area and the following integration with neuromorphic hardware allow to verify new methods of training and thus achieve groundbreaking results in this field. To explore different opportunities, we use several software packages for various purposes in the project.

a. NxSDK

The Neuromorphic (Nx) Software Development Kit is a complete toolchain developed by Intel for working with Loihi platform. It is an intuitive set of APIs allowing algorithms developers and application engineers to quickly program Intel's Neuromorphic Hardware with support for different programming paradigms.

b. Nengo

Nengo is a graphical and scripting based Python framework working on the top of Tensorflow [14]. It allows to define own types of neurons, learning rules, optimization methods and reusable subnetworks. Simulation of large-scale neural networks with optimization of model parameters is possible in the framework thanks to the Nengo-DL package.

While the main part of Nengo is based on Tensorflow, the big advantage of this framework is an extension package named Nengo-Loihi, which allows to run SNNs on the neuromorphic boards. It contains an emulator backend for fast model development and easier debugging, and a hardware backend for executing code on the chip. In the background, it uses Intel's NxSDK API to interact with the host and configure the Loihi board.

c. SNN Toolbox

The SNN conversion toolbox contains functions to transform rate-based ANNs into SNNs and to simulate them [15]. It automates the conversion of pre-trained models and provides tools for testing SNNs in a spiking neuron simulator. Current support for input networks incudes Keras, Lasagne, and Caffe, however for the moment interface to NxSDK is not available, hence deployment of SNNs with toolbox on the Loihi chip is not possible [16].

5. JET TAGGING TASK

Jets are collimated showers of particles that result from the decay and hadronization of quarks and gluons [17]. At the LHC, due to the high collision energy, a particularly interesting jet signature emerges from overlapping quark-initiated showers produced in decays of heavy standard model particles. Jet substructure at the LHC has been a particularly active field for machine learning techniques as jets contain O(100) particles whose properties and correlations may be exploited to identify physics signals. The high dimensionality and highly correlated nature of the phase space makes this task an interesting testbed for machine learning techniques. There are many studies that explore this possibility, both in experiment and theory. One of them is the Jet Tagging Task, where five types of particles are being classified: gluon (**g**), quark (**q**), W boson (**w**), Z boson (**z**), top quark (**t**). This research is a part of HLS4ML, which presents a case study for neural network inference in FPGAs [18]. The classifier for jet substructure would enable, among many other physics scenarios, searches for new dark sector particles and novel measurements of the Higgs boson.

A dataset for this problem consists of a set of physics-motivated high-level features. Jets could be represented as an image or as a list of particles. These different representations are used to train different kinds of networks while solving the same classification problem.

An effort to train a deep neural network for Jet Tagging Task has been made, resulting in two simple network architectures (Fig. 4). The fully-connected neural network with three hidden layers (left) is characterized by categorical cross-entropy loss function, which is minimized using the Adam algorithm with an initial learning rate of 10^{-4} and a minibatch size of 1024. Moreover, a simpler architecture with one hidden layer (right) was considered, when identifying top quarks. Models were trained with Keras, resulting in the accuracy discussed in Sec. 6.

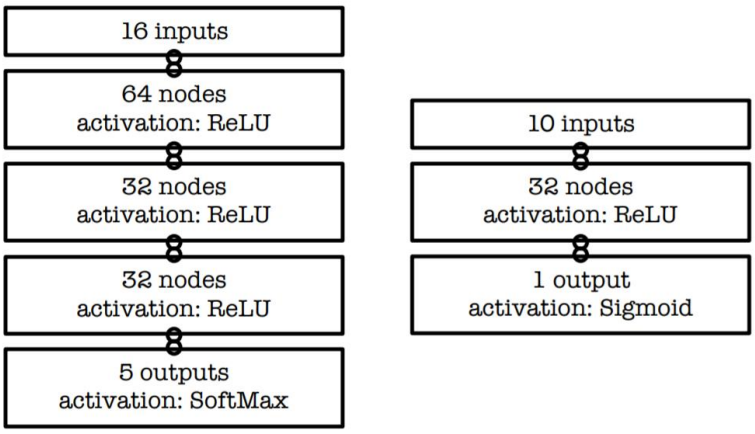

Figure 4. *Two neural network architectures for jet substructure classification. (Left) A three-hidden-layer model used to categorize five classes of jets (q, g, W, Z, and t). (Right) A one-hidden-layer model used to identify top quarks.*

6. NEUROMORPHIC IMPLEMENTATIONS AND RESULTS

Performance of the current deep neural network classifier for the the Jet Tagging Task (Sec. 5) is presented in the confusion matrix (Fig. 5). The average accuracy of this model over five classes is **75.2%**. We propose an alternative approach to solve this task with neuromorphic computing and present successful implementations of corresponding SNNs which achieve comparable results with present conventional machine learning state-of-the-art in terms of accuracy. We use different software possibilities (Sec. 4) and deploy the model on the Loihi chip (Sec. 3), that allows for dramatic energy savings, increased speed of inference and gives new perspectives for further research in this domain.

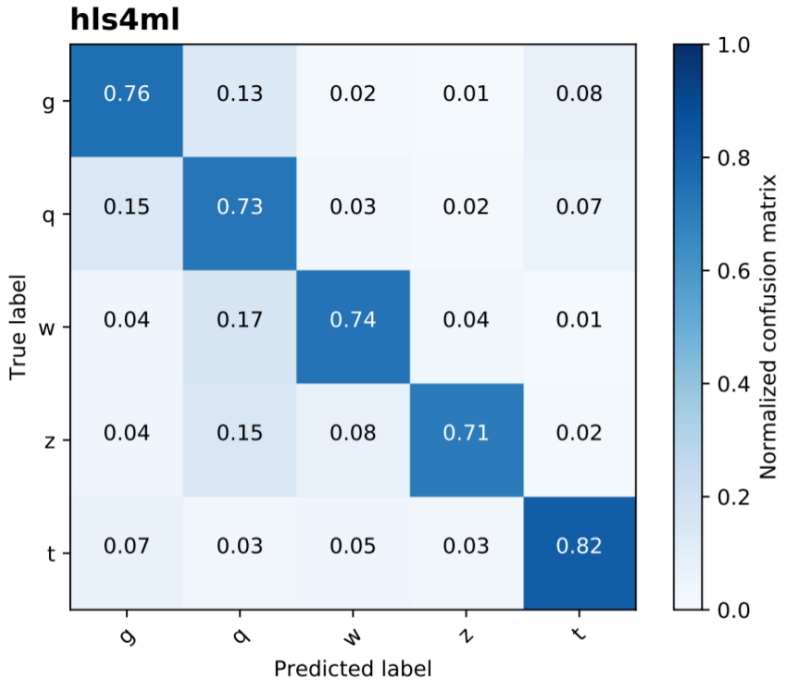

Figure 5. *The normalized confusion matrix of accuracy with DNNs at the Jet Tagging Task.*

a. Conversion and Simulation

As a first attempt, we propose to train and convert the DNN presented in Sec. 5. (Fig. 4. left). This approach allows to eliminate the problems of gradient descent in SNNs by adapting weights and parameters of the spiking neurons. The goal is to achieve the same input-ouput mapping with a deep SNN as the original DNN. The main advantage of this method is that conventional model can be trained without considering the later conversion process. Once the parameters of the DNN are known, conversion consists only of parsing and simple transformations, and thus adds only negligible training overhead. For this process we use SNN Toolbox (Sec. 3.c) with INIsim and Brian2 backends [19], which as input takes the exported Keras model with pre-trained weights in the h5 format. We prove in simulation that classifying jets with neuromorphic computing is possible and present our results in the Fig. 6. This straightforward way works very well for testing and research purposes. Nevertheless, the method comes with its flaws: first of all, at the moment SNN Toolbox does not support deployment of SNNs on the Loihi hardware. Secondly, the converted network directly depends on the conventional model, hence its performance could not be improved due to the mapping inaccuracies.

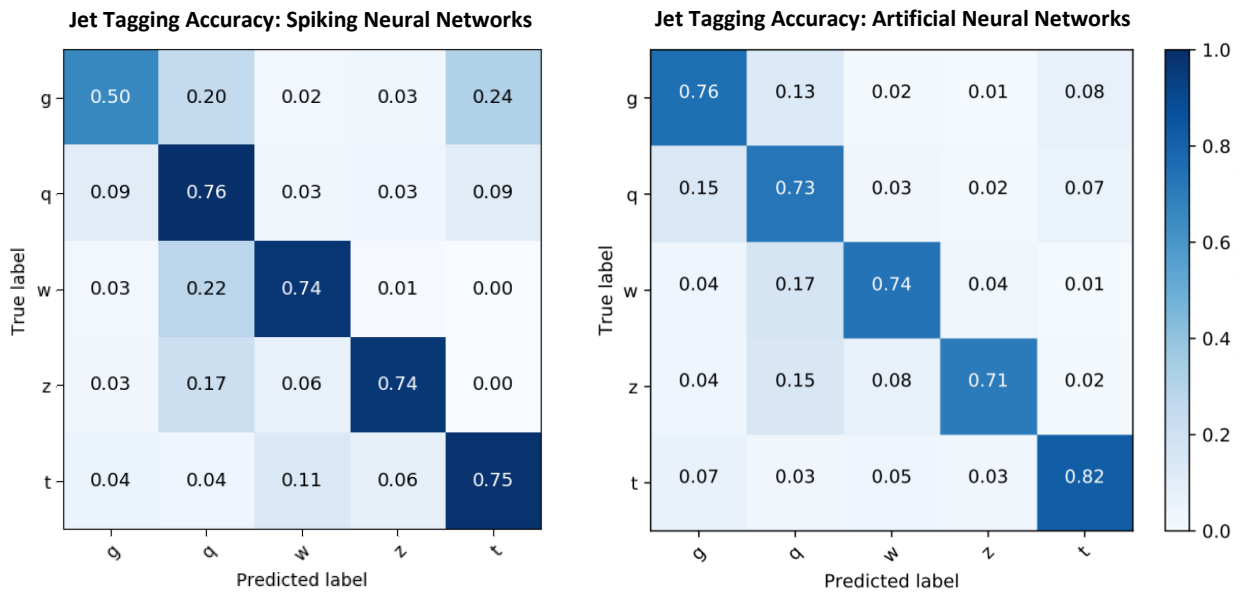

Figure 6. *Both normalized confusion matrices of accuracy at the Jet Tagging Task presented for simple comparison: Converted SNNs simulated with SNN Toolbox (left). ANNs ran on conventional hardware with Keras (right).*

The best accuracy over five classes achieved with converted network in SNN Toolbox is equal to **69.72%**. This result can be already considered imposing seeing that it proves that implementation of spiking models for jet classification can be done with satisfactory level of precision. Mapping process between the networks can lead to a small loss of information, however novel techniques that ensure that the actual SNN operation is in the loop during the conversion phase have been recently developed [20]. We hope that these possibilities will be also available in SNN Toolbox in the future and expect its integration with Loihi platform.

b. Deployment on dedicated hardware

Besides the simulation, we explore various possibilities for training of SNNs and running their inference on the neuromorphic hardware. We use the Nengo framework (Sec. 4.b) to develop our own spiking models, and NxSDK (Sec. 4.a) which manages the inference via Loihi board.

Training of the model happens within Nengo-DL in rate (artificial) neurons, and is further used by Nengo-Loihi, which allows to simulate or run the inference in the neuromorphic chip through the NxSDK. There are no changes to the network when it becomes spiking, other than the neurons emit discrete spikes rather than continuous rates. Information in the network is transmitted by the firing rates of spiking neurons, which are analogous to the rates that were used during training.

We develop an own model corresponding to previously converted DNN, with three, hidden, fully-connected layers. We use Spiking Rectified Linear neurons, where each neuron's activity scales linearly with current, unless the current is less than zero, at which point the neural activity will also stay at zero. Weights are initialized with Glorot method (also known as Xavier initialization) [21] and corresponding layers are connected by Nengo nodes. The connection between two objects is unidirectional, transmitting information from the first (pre), to the second (post) argument. To make neurons start at the same time, we adjust their max rate and intercept according to the defined signal amplitude (in our case 0.005):

```
net.config[nengo.Ensemble].max_rates = nengo.dists.Choice([1/amplitude])
net.config[nengo.Ensemble].intercepts = nengo.dists.Choice([0])
```

As an input the network takes sixteen high-level features and classifies five types of jets as an output.

We add probes to each hidden-layer that allows to count the number of spikes over the whole simulation and helps in the debugging process. We want to ensure that the neurons are firing fast enough, but on the other hand not too fast. An average rate around 100-200 Hz is sufficient. To translate features into spikes, first layer of the model is defined as an off-chip:

```
net.config[layer.ensemble].on_chip = False
```

After implementing the model we use Nengo-DL to perform the training. As we built the network, it has no synaptic filters [22] on the neural connection. This works well during training, however generates a significant error at the evaluation process. The achieved accuracy after training is equal to **46.83%**. We can improve this performance by adding synaptic filters, which optimize our trained model and are particularly useful for layers with lower firing rates [23]:

```
for conn in model.all connections:
        conn.synapse = 0.005
```

For data evaluation we are running the network over time using spiking neurons. For this purpose we repeat the input/output data for a number of timesteps, basing on the defined presentation time (in our case 0.1). Final accuracy of the optimized model over classes is equal to **65.08%**. Thanks to a vector of times which matches the probed output, discrete results of spikes can be interpolated to continuous functions for every specific event. Initially, the potential of all neurons is equal and their increasing activity decides about final classification. A plot of an output activity of spiking neurons that classify a gluon (g) is presented in the Fig. 7.

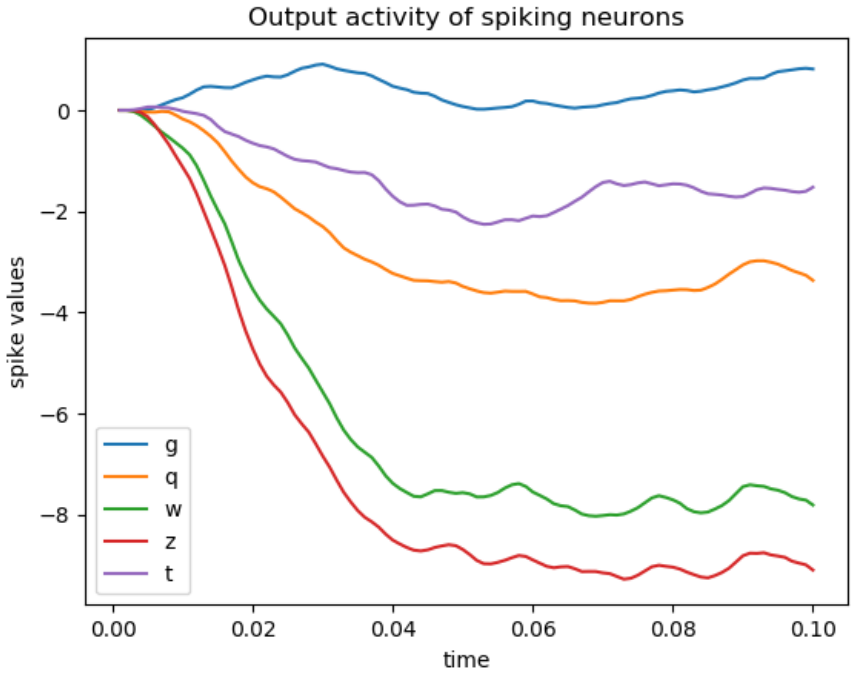

Figure 7. *Discrete output activity of spiking neurons that classify a gluon (g). Initially, all neurons start at the same level. Their increased firing after time demonstrates the final characterization of specific classes.*

By using the benefits of Nengo framework, after training and evaluating the model, we are finally able to load it to the Nengo-Loihi package and deploy on the neuromorphic hardware. To do so, we also increase the maximum number of input spikes per step:

```
if 'loihi' in sim.sims:
        sim.sims['loihi'].snip_max_spikes_per_step = 120
```

We finally obtain **69.8%** of accuracy over classes, as presented in the Fig. 8.

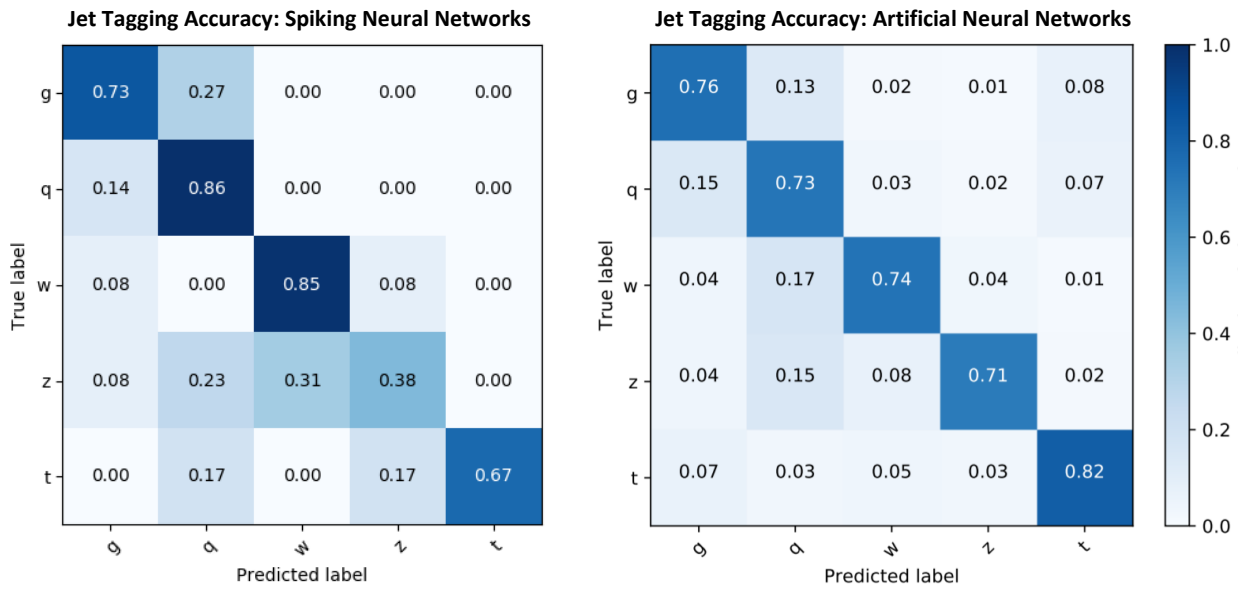

Figure 8. *Both normalized confusion matrices of accuracy at the Jet Tagging Task presented for simple comparison: SNNs deployed on the neuromorphic chip with Nengo-Loihi (left). ANNs ran on conventional hardware with Keras (right).*

The overall error of this result is only 5% bigger than the best performance of conventional DNNs on the task. It is worth to outline that only one class of particles (Z boson) is significantly misclassified, whereas the other types perform satisfactory if not better than in classical approach. The Z boson achieves the lowest accuracy in both cases, however its error with SNNs is clearly bigger. Nevertheless, the results of two classes (W boson and quark) were improved by over 10%, which can be considered as an impressive proof of work. We expect that rapid developments in this domain will allow to achieve even better results thanks to utilization of new computing possibilities in direct training of SNNs. Integration between software and neuromorphic hardware will also allow to implement bigger, more complex spiking topologies and result in an outstanding performance of SNNs trained on the event-based data.

It is important to remember about numerous advantages of neuromorphic computing such as significant energy efficiency and impressive speed of inference. Even though recent results are comparable with DNNs in terms of accuracy, we can still benefit from these advantages and outperform currently used solutions thanks to the dramatic energy savings and lower time of data processing. These aspects are important especially in the context of triggering system at LHC, where the speed of filtering is crucial and could have the highest priority when a satisfactory level of accuracy is achieved.

Moreover, it is worth to mention that representation of convolutional layers with spiking neurons is likewise possible [24]. In this case we also tested a classification of particles on the pictures, however

the densely-connected model was used in the project as an ideal implementation for the natural representation of jets. Recently, SNNs are also successfully applied on most common image datasets such as MNIST or CIFAR. Their improving performance on these tasks sheds new light on applications of neuromorphic computing in classification of real image data.

Because the current study was realized on the Loihi chip through the remote (cloud) connection, we are unable to present benchmarks of the chip on energy efficiency. Nevertheless, we would like to refer to the analysis presented by the Applied Brain Research titled '*Benchmarking Keyword Spotting Efficiency on Neuromorphic Hardware*' [25]. We were likewise able to reproduce presented results on the Keyword Spotting Task with Nengo and run the model on the Loihi chip. Due to the very close construction of our model on the Jet Tagging Task, we expect similar results in performance of the networks in terms of energy cost and average inference speed. The selected benchmarks are presented in the Fig. 9.

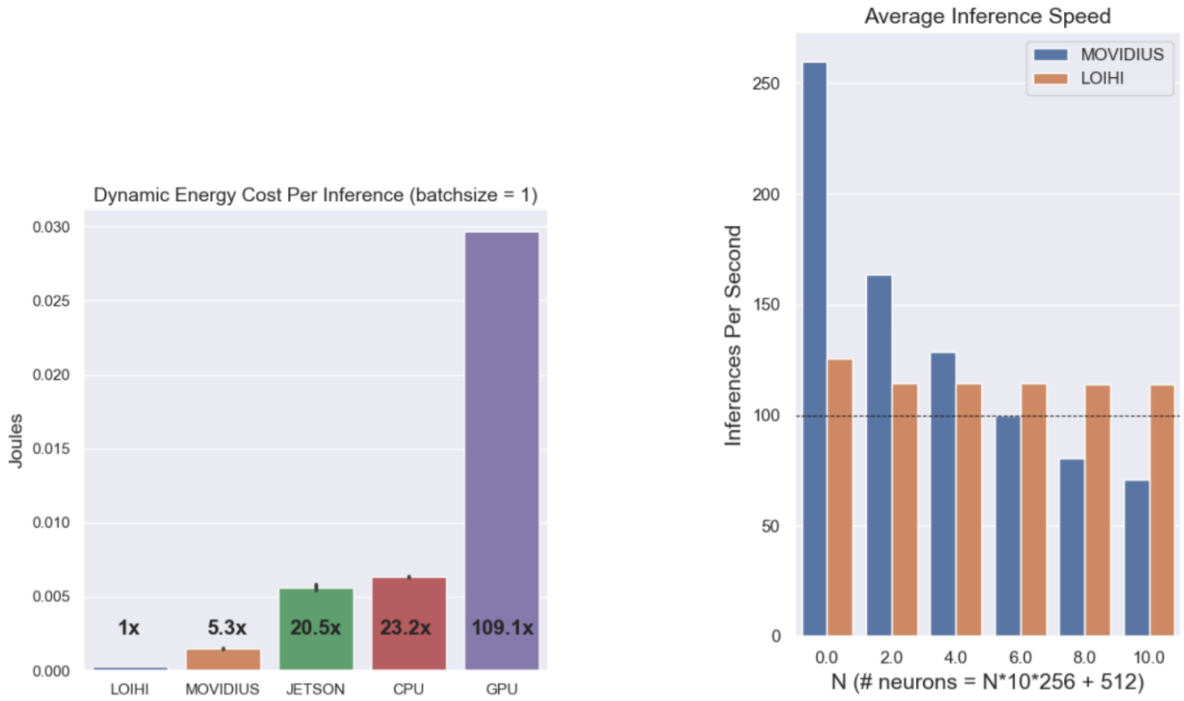

Figure 9. *Benchmark on the Loihi chip at the Keyword Spotting Task presented by Applied Brain Research. (Left) A Dynamic Energy Cost Per Inference between five devices. (Right) Average Inference Speed between Intel Movidius and Intel Loihi.*

We observe massive differences in the cost of energy per inference (Fig. 9. left). It is worth to outline that Loihi chip consumes over one hundred times less power in comparison to GPU on the same task. Moreover, SNNs are also very scalable, which means that their speed of inference on the neuromorphic hardware does not dramatically change with an increased number of neurons in a topology. This is also their advantage over classical networks, what can be presented on the benchmark (Fig. 9. right) with Intel Movidius Stick, which is known for its fast computations on matrices [26]. Various advantages of SNNs can be a very important and promising field of research in the following years, thus we believe that HEP can be a perfect place for testing and exploring upcoming cutting-edge technologies in this domain.

7. CONCLUSION

Currently transistors are around 10-15 nanometers in scale and are expected to shrink to around 3-5 nanometers in the next few years, but that is seemed to be about far as we can go [27]. At that point, transistors are so small that physical limits prevent them from working properly. This challenge is also one of the reasons to explore novel possibilities of computations. Neuromorphic computing is certainly a part of this research and can become the future of computing not only at the European Organization for Nuclear Research, but the future of computing in general. We hope that our first implementation of SNNs in this domain using Loihi chip will lead to intense developments of neuromorphic technologies during the High-Luminosity LHC upgrade. We see significant potential in characterization of particle types basing on their signals and prove its correctness in the Jet Tagging Task.

High Energy Physics is a promising field of future work and an interesting candidate of applications for SNNs due to the number of event-based sensors and thus spike-friendly datasets. Various utilizations of time series and a crucial problem of signal-to-noise discrimination are just few examples for further research in this domain. We expect a stronger integration between the leading conventional deep learning frameworks, such as PyTorch and Tensorflow/Keras with dedicated neuromorphic software. Moreover, we hope that novel computing possibilities for direct training of SNNs, such as Spike Layer Error Reassignment in Time [28], Evolutionary Optimization Frameworks [29], or Surrogate Gradient Learning [30] as well as a close cooperation between hardware and software producers will allow to benefit from various advantages of SNNs and achieve even more innovative results. Collaborative research groups such as INRC or The Human Brain Project [31] can lead to better cooperation between scientists and result in more structured technological developments. All these factors show that although conventional AI approaches are effective and widely-spread, neuromorphic computing is still a very important field of science.

The repository of this project is public and fully-accessible at: github.com/borzyszkowski/SNN-CMS

8. REFERENCES AND BIBLIOGRAPHY

[1] Michael Pfeiffer, Thomas Pfeil, “Deep Learning with Spiking Neurons: Opportunities and Challenges”, 2018

[2] Amirhossein Tavanaei et al., “Deep learning in spiking neural networks”, 2019

[3] Wolfgang Mass, “Networks of Spiking Neurons Learn to Learn and Remember”, 2018

[4] Mike Davies et al., “Loihi: A Neuromorphic Manycore Processor with On-Chip Learning”, 2018

[5] Qian Liu et al., “Benchmarking Spike-Based Visual Recognition: A Dataset and Evaluation”, 2016

[6] Linghao Song et al., “Deep Learning for Vertex Reconstruction of Neutrino-Nucleus Interaction Events with Combined Energy and Time Data”, 2019

[7] Devin Soni, “Spiking Neural Networks, the Next Generation of Machine Learning”, 2018

[8] Jack Terwilliger, “(A Bit of) Biological Neural Networks – Part I, Spiking Neurons”, 2018

[9] Huanneg Qiu et al., “Evolving Spiking Neural Networks for Nonlinear Control Problems”, 2019

[10] Bodo Rueckauer, Shih-Chii Liu, “Conversion of analog to spiking neural networks using sparse temporal coding”, 2010

[11] Wulfram Gerstner, Werner M. Kistler, “Spiking Neuron Models: Single Neurons, Populations, Plasticity”, 2002

[12] David Schor, “Wikichip - Intel Loihi”, 2018

[13] Eric Hunsberger et al., “Nengo-Loihi overview”, 2019

[14] Trevor Bekolay et al., “Nengo: A Python tool for building large-scale functional brain models”, 2014

[15] Bodo Rueckauer et al., “Conversion of continuous-valued deep networks to efficient event-driven networks for image classification”, 2017

[16] Bodo Rueckauer, “SNN Toolbox documentation”, 2018

[17] Eric A. Moreno et al., “Interaction networks for the identification of boosted H→bb¯ decays”, 2019

[18] Javier Duarte et al., “Fast inference of deep neural networks in FPGAs for particle physics”, 2018

[19] Dan Goodman, Romain Brette, “The Brian Simulator”, 2009

[20] Abhronil Sengupta et al., “Going Deeper in Spiking Neural Networks: VGG and Residual Architectures”, 2019

[21] Xavier Glorot, Yoshua Bengio, “Understanding the difficulty of training deep feedforward neural networks”, 2010

[22] Peter Duggins, “Incorporating Biologically Realistic Neuron Models into the NEF”, 2017

[23] Aaron R. Voelker, Chris Eliasmith, “Improving Spiking Dynamical Networks: Accurate Delays, Higher-Order Synapses, and Time Cells”, 2018

[24] Ruthvik Vaila et al., “Deep Convolutional Spiking Neural Networks for Image Classification”, 2019

[25] Peter Blouw et al., “Benchmarking Keyword Spotting Efficiency on Neuromorphic Hardware”, 2018

[26] Albert Reuther et al., “Survey and Benchmarking of Machine Learning Accelerators”, 2019

[27] Shengman Li et al., “Nanometre-thin indium tin oxide for advanced high-performance electronics”, 2019

[28] Sumit Bam Shrestha, Garrick Orchard, “SLAYER: Spike Layer Error Reassignment in Time”, 2018

[29] Catherine D. Schuman et al., “An Evolutionary Optimization Framework for Neural Networks and Neuromorphic Architectures”, 2016

[30] Hesham Mostafa et al., “Surrogate Gradient Learning in Spiking Neurons”, 2019

[31] Katrin Amunts et al., “The Human Brain Project: Creating a European Research Infrastructure to Decode the Human Brain”, 2016