Machine Learning Community White Paper

See page 21 for the full author list

June 2017

Contents

1 Introduction

As particle physics enters the post-Higgs discovery era, we want to exploit the full physics potential of the LHC. The high luminosity LHC (HL-LHC) will deliver data from 100 times the luminosity of the original LHC, bringing quantitatively and qualitatively new challenges due to event size, volume, and complexity. The physics reach of the experiments will be limited by computational resources and physics performance of algorithms. Based on achievements in data science, very generally, and HEP, more specifically, machine learning (ML) promises to provide improvements along both of these dimensions. Areas where significant improvements are needed include:

- Increasing the **physics performance** of algorithms, including reconstruction and analysis;
- Reducing the **execution time** of the most computationally expensive parts of event simulation, pattern recognition, and calibration;
- Reducing the **data footprint** by optimizing data compression and optimizing data placement and access.

Incorporating machine learning in HEP workflows to provide solutions in these areas will require significant research and development over the next five years. The research and development will go in defining the best algorithms for the challenges that will be faced (section 2), their software implementation in production framework (section 3), the computing resource needs (section 5), reaching out efficiently to the data science community (section 6) and educating HEP researcher on these new technique (section 7). Many of the topics that will require work in the coming years are also addressed in several other sections of the CWP, mostly data management, work-flow management, framework, computing model, training, and software integration.

1.1 Motivation

The experimental program of the HL-LHC revolves around two main objectives: probing the Standard Model with increasing precision and searching for new physics. Both tasks require the identification of rare signals in immense backgrounds. Significantly increased levels of pile-up at the HL-LHC will further complicate signal extraction.

Machine learning is already used in HEP. Applications include event and particle identification, energy measurement and pile-up suppression. Machine learning algorithms outperform traditional techniques in these areas. Despite their current advantage, machine-learning algorithms of today have significant room for improvement to maximally exploit the full potential of data.

Modern machine-learning tools are themselves undergoing a process of rapid evolution. The HEP community can benefit from these advancements by developing flexible software with HEP priorities in mind. Machine learning community benefits from rapid development and experimentation. Being in tune with latest progress would be very beneficial to HEP. A major challenge is how to engage the ML community and to build interfaces to ML developments.

1.2 Machine Learning and High-Energy Physics

There is an inherent interaction between the HEP and ML communities, whose priorities are not always aligned. The two groups can benefit from increased interaction. Efforts are underway to bring together HEP and ML experts but more work is needed to bring the two communities together. ML methods are designed to exploit large datasets in order to reduce complexity and can be used to identify new features in data.

We foresee that the use of ML will become even more important as we move towards the more challenging High-Luminosity LHC (HL-LHC) environment in the next 10 years.

1.3 Brief Overview of Machine Learning Algorithms in HEP

This section provides a brief introduction to the most important Machine Learning algorithms in High Energy Physics, introducing key vocabulary (in *italic*). Specific applications are detailed in section 2.

The most frequently used machine learning algorithms in HEP are Boosted Decision Trees (BDTs) and Neural Networks (NN).

Typically, variables relevant to the physics problem are selected and a *model* is *trained* for *classification or regression* using signal and background events (or *instances*). As typical for all ML algorithms, training is the most human- and CPU- time consuming. While the application of the model to classification of new events, the so called *inference* stage, is relatively inexpensive. BDTs are typically used for particle identification, or in analysis to identify specific final states. BDT are also more and more used for *regression*, where a continuous variable is learned, for example to have the best estimate of a particle energy based on the measurements from several detectors.

Neural Networks have been used in HEP for some time, however improvements in training algorithms and computing power have led in the last decade to the so-called Deep Learning revolution, which is now percolating into HEP. Deep Learning is particularly promising when there is a large number of inputs, and/or when the question asked is more complex than a simple classification or regression task.

One particular type of NN are Generative Models, when a Neural Network is trained to mimick input multidimensional distributions, and can generate any number of new instances respecting the input distributions. Variational AutoEncoders and more recent Generative Adversarial Network are to examples of such Generative Models.

A large set of Machine Learning algorithms are devoted to time series analysis and prediction. They are in general not relevant for HEP where events are processed as collections. However, there is more and more interest in these algorithms for Data Quality Monitoring or for Computing Infrastructure monitoring, where the time dimension is important.

1.4 Structure of the Document

Potential applications of machine learning algorithms are detailed in section 2 to present ideas, needs and physics drivers of such R&D. The section will establish how relevant ML is for the field of HEP, and even more so in the horizon of HL-LHC. The applications described are relevant beyond the LHC experiments and have been used and achieved ground-breaking potential in neutrino physics.

With a large amount of tools available outside of the HEP community and a long standing tradition of having homegrown HEP software, a significant amount of work will have to go in mapping the needs and requirements of ML into mutation in HEP production software. Section 3 underlines the key items for R&D. Section 4 discusses the interplay between internally and externally developed machine learning tools in HEP.

The advent of machine learning in recent decades was made possible partly by evolution in available hardware for training complex models. In section 5 we address, with reference to other chapters of the CWP, the specific requirements of training large models over large datasets.

Even though applying most ML techniques do not require external expertise, when the problems are simple, some of the challenges exposed in this document are very specific to the field of HEP and will require significant R&D in the methods themselves. Section 6 presents a road-map for reaching out to the data science experts for their collaboration. In particular, it will be critical to align priorities in both fields of research and current effort to bring together HEP and ML experts will need to continue throughout the coming years.

In view to the adoption of new mathematical and statistical techniques, widely used industry for the benefit of tackling HEP challenges, as well as training newcomers to technique that can be re-used in future steps in their curriculum, a detailed teaching plan will have to be elaborated. Section 7 describes the view of the community on how to achieve the dissemination of knowledge on the bleeding edge techniques that will be adopted.

2 Machine Learning Applications and R&D

The following chapter describes the science drivers and high-energy physics challenges where machine learning can play a significant role in advancing the current state of the art. These challenges are selected because of their relevance and potential and also because of similarity with challenges faced outside the field. Despite such similarities, major R&D work will go in adapting and evolving the methods to match the particular HEP requirements. The adaption process will require much interaction with data scientists. Such collaborations will advance both fields in both diversity and reach.

2.1 Simulation

Particle discovery relies on the ability to accurately compare the observed detector data with expectations of the detector response to discriminate between hypotheses of the Standard Model or models of new physics. While the subatomic process of individual particles interactions with matter are known, it is intractable to compute the detector response analytically. As a result high fidelity Monte Carlo based tools, such as GEANT [1], have been developed to simulate the propagation of particles resulting from a proton-proton interaction through particle detectors.

For the high luminosity LHC, on the order of trillions of collisions will need to be simulated in order to achieve the statistical accuracy of the simulations needed for precision hypothesis testing. However, such simulations can be extremely computationally costly. For example, simulating the detector response of a single ATLAS proton-proton collision event take on the order of several minutes [**WhereDoesThisComeFrom**]. Among the most costly aspect of this simulation process is the simulation of a particle incident on the dense

material of a calorimeter. The subsequent radiative and nuclear interactions result in the production of a multitude of secondary particles that collectively are referred to as a shower. The high multiplicity of the shower, and the high interaction probability of particles passing through the dense material, make the simulation of such processes highly computationally costly. This problem is even further compounded when the response of individual particle showers overlap, as may happen in the core of the collimated stream of particles produced by high energy quarks and gluons also known as jets.

Fast simulation is the process of replacing the slowest components of the GEANT chain with computational efficient approximations. Often such approximations have been done by simplified detector response parameterizations or through particle shower look-up tables, which are computationally fast but often suffer from insufficient accuracy for high precision physics measurements and searches.

Machine learning may offer a solution. Recent progress in high fidelity fast generative models, which are able to sample high dimensional feature distributions by learning from existing data samples. Such techniques include Generative Adversarial Networks (GAN) and Variational Auto Encoders (VAE). The application of such techniques to LHC simulation may provide a computationally efficient approach to fast simulation which, unlike the simplified models used previously, can capture subtle correlations in the feature space resulting in reasonably high precision simulation.

The development of generative models for fast simulation may provide both a fast and high enough accuracy simulations for use at the HL-LHC. A simplified first attempt at using such techniques saw orders of magnitude increase in simulation speed over existing fast simulation techniques [2], but has note yet reached the required accuracy. Developing these techniques for realistic detector models and understanding how to reach the required accuracy is still needed, yet the fast advancement in the ML community of such techniques makes this a highly promising avenue to pursue.

2.2 Real Time Analysis and Triggering

The traditional approach to data analysis in particle physics assumes that the interesting events recorded by a detector can be selected in real-time (a process known as "triggering") with a reasonable efficiency, and that once selected, these events can be affordably stored and distributed for further selection and analysis later. However, the enormous production cross-section and luminosities of the LHC mean that these assumptions break down.[1] In particular there are whole classes of events, for example beauty and charm hadrons or low-mass dark matter signatures, which are so abundant that it is unaffordable to store all the events for later analysis. This is why, in order to fully exploit the physics reach of the LHC, it will increasingly be necessary to perform more of the data analysis in real-time.

This topic is discussed in some detail in the Reconstruction and Software Triggering chapter, but it is also an important driver of the use of machine learning in HEP. Many of the overabundant signals which must be analyzed in real-time are also the most computationally expensive to reconstruct in the first place. Machine learning methods offer the possibility to offset some of the cost of these algorithms from execution/inference to training, and may be the only hope of affordably performing the real-time reconstruction which enables the real-time analysis in the first place. This also connects to the increasingly heterogeneous computer architectures, since the most cost-effective architecture may be different for each algorithm, and the real-time analysis has to fit within a fixed budget so that the data are not lost. One illustrative example RD project is HEP.TrkX, which is looking into training DNNs (e.g. CNNs or RNNs) on large resource platforms (e.g. HPCs) and subsequently use them to allow fast inference in online systems (e.g. on FPGAs). Real-time analysis also poses certain specific challenges to machine learning algorithms, in particular how to design algorithms to be as insensitive as possible to the underlying detector performance which may vary over time. For example, LHCb already uses NNs for fast fake-track and clone rejection, but it will be important that these approaches maintain performance for higher detector occupancies across the full range of tracks used in physics analyses.

In addition, the increasing event complexity particularly in the HL-LHC era will mean that Machine Learning techniques may also become ever more important to maintaining or improving the efficiency of traditional triggers which select full events for saving offline. Examples of where ML approaches might improve performance are triggering of electroweak events with low-energetic objects; improving jet calibration at very early stage of reconstruction allowing jet triggers thresholds to be lowered; or supernovae and proton decay triggering at DUNE. Another related application is speeding up the reconstruction of beauty, charm, and other lower mass

[1] They may well also break down in other areas of high-energy physics in due course.

hadrons, where traditional track combinatorics and vertexing techniques may become too computationally expensive at higher detector occupancies.

2.3 Object Reconstruction, Identification, and Calibration

The physical processes of interest in high energy physics experiments occur on time scales too short to be observed directly by particle detectors. For instance, a Higgs boson produced at the LHC will decay within approximately 10^{-22} seconds and thus decays essentially at the point of production. However, the decay products of the initial particle, which are observed in the detector, can be used to infer its properties. Better knowledge of the properties (e.g. type, energy, direction) of the decay products permits more accurate reconstruction of the initial physical process.

Particles are observed in a detector through the energy they deposit when traversing material, which is subsequently digitized. Reconstruction is the process of converting the raw digital signals in the detector into the physical properties of particles. Particle Physics Detectors are usually composed of several sub-detector, each taking advantage of specific interaction mechanism to detect passage of usually a specific types of particles and then measuring the properties of the particles. There is a variety of sub-detector technologies, but most belong to one of three categories:

- Tracking Detectors: These detectors measure the trajectory of charge particles by spatially locating ionization. Usually trackers are placed in a magnetic field, so that the particle momentum can be inferred from the curvature of the trajectory. Very precise tracking detectors, such as those that employ silicon, provide sufficient spatial resolution to enable locating the particle creation and/or decay point. The ionization also allows identifying the particle type.

- Calorimeters: These detectors measure the particle energy by causing them to interact and loose the energy in material and counting secondary particles. Highly segmented calorimeters allow measure the profile of the energy deposition with enable identifying the type of the particle.

- Particle Identification: These detectors are aimed at determining the particle type using a variety of techniques.

Algorithmic reconstruction typically involves several steps that turn "raw" measurements from detectors into "features":

- Feature Extraction: The "signal" from the passage of particles through a detector element (e.g. cell in calorimeter) is observed above noise in the raw electronic output (e.g. voltage) associated with the element. This signal is then characterized (e.g. size and time).

- Pattern Recognition: The pattern of signals in geometrically adjacent detector elements is associated with the passage of a signal or group of particles. In calorimeters, this step is commonly referred to as clustering.

- Object Characterization: Properties of the objects are measured. In tracking detectors, this step means fitting a pattern of "hits" to a helix. In calorimeters, this step extracts the energy, location, and other properties of the cluster that for example characterize the shape of the cluster.

- Combined reconstruction: Objects in different are associated together to create a particle candidate.

Often the passage of a particle is interpreted differently by different reconstruction algorithms. An early step of physics analysis usually involves selecting a single interpretation from the outputs of these algorithms. This step is usually referred to as particle identification. Machine learning can in principle be applied at any of these steps. For example, experiments have trained ML algorithms on the features from combined reconstruction algorithms to perform particle identification for decades. In the past decade BDTs have been one of the most popular techniques in this domain. More recently, experiments have been able to extract better performance with Deep Neutral Networks in some cases.

An active area of research is performing particle identification and extracting particle energies on the output of Feature Extraction using DNNs, in particular for calorimeters or time projection chambers (TPCs), where the data can be represented as a 2D or 3D image and the problems can be cast as a computer vision tasks, in which neural networks are used to reconstruct images from pixel intensities. These neural networks must be adapted for particle physics applications by optimizing network architectures for complex, 3-dimensional detector geometries and training them on suitable signal and background samples derived from data control regions. A particularly important application is to Liquid Argon TPCs (LArTPCs), which is the chosen detection technology for U.S. flagship neutrino program. Here, algorithm reconstruction has proven to be difficult while early attempts at

Convolutional Neural Networks (CNNs) have shown better performance with little effort. Other applications include identification and measurements of electrons and photon from electromagnetic showers, jet properties including substructure and b-tagging, taus and missing energy. Promising deep learning architectures for these tasks include convolutional, recurrent and adversarial neural networks.

For tracking detectors, pattern recognition is the most computationally challenging step. In particular, this step becomes computationally intractable for the HL-LHC. Finding a solution to this problem is an active area of research. The hope is that Machine Learning will provide a solution here. To this end, an effort known as TrackingML is underway to create datasets and establish a challenge to elicit solutions from the broader ML community, while the HEP.TrkX project investigates Deep NN pattern recognition algorithms that have the potential to scale linearly with LHC intensity and to run efficiently on many-core processors.

2.4 Sustainable Matrix Element Method

The Matrix Element Method (MEM) is used in [scope]. In the MEM, an *ab initio* calculation is performed to compute a per-event (X) signal (S) probability $P(S|X)$ given by (using Bayes' theorem)

$$P(S|X) = \frac{\sum_{iy} \alpha_{Si} \mathcal{P}(\mathcal{X}|\mathcal{S}_\rangle)}{\sum_i \alpha_{Si} \mathcal{P}(\mathcal{X}|\mathcal{S}_\rangle) + \sum_\rangle \alpha_{\mathcal{B}\rangle} \mathcal{P}(\mathcal{X}|\mathcal{B}_\rangle)} \tag{1}$$

Here, S_i and B_j, denote all signal and background processes that are being considered. The *a priori* probabilities α_{Si} and α_{Bi} are given by the expected fraction of events of each process in the selected candidate events. The value of $P(S|X)$ is taken as the discriminant used in signal extraction. The likelihood values $P(S|Xproc)$ for a given process H_proc can be computed by means of the factorization theorem from the corresponding partonic cross-sections of the hard scattering process and is given by (see **??** for details)

$$P_X(\vec{x}) = \frac{1}{\langle\sigma\rangle} \int \frac{d\sigma(\vec{y})}{d\vec{y}} \, \epsilon(\vec{y}) \, G(\vec{x}, \vec{y}) \, d\vec{y} \tag{2}$$

where σ_{Hproc} and $|M_{Hproc}|^2$ are the total cross section and scattering amplitude, respectively, for the given process. The mapping between the measured event X and the parton (initial) state is implemented by transfer functions $W(X, y)$ which take into account the detector resolution functions and reconstruction efficiencies. The ME approach is very powerful because it does not require training samples (being an *ab initio* calculation) and incorporates all available kinematic information into a single discriminant $P(S|X)$. However, it is computationally intensive since it involves performing high-dimensional integration for a large number of events, signal and background hypotheses, and systematic variations. In practice, evaluation of definite integrals by the ME approach involves randomly chosen points over a large domain in phase space. Further complicating accurate evaluation is the fact that many of the integrands are sharply peaked (a consequence of imposing energy/momentum conservation in the processes) and therefore the importance sampling Monte Carlo integration algorithm VEGAS **??** is employed. In addition, the transfer functions rely on tediously hand-crafted fits to full simulated Monte-Carlo events.

Despite the attractive features of the MEM analysis approach, the computational burden of the calculations limits the range of its applicability. This is especially true when one considers that numerous systematics and sample iterations required for any practical analysis. The MEM calculations for a single physics process are computationally intensive – when folds in different processes, systematics, sample iterations, etc., MEM analysis are often not a practical approach to complete a time search analysis or measurement.

The idea is to use machine learning techniques to approximate the evaluation of the integral shown in Eq. 2. The approach of applying neural networks to multidimensional integration problems is plausible (see **??**, for example). The challenge is to design the network to be sufficiently rich (e.g. DNN) to encode the complexity of the MEM calculation over the phase space relevant for each process such that the DNN is a good approximation to the true value of Eq. **??**.

The approximation of MEM calculations by DNNs would greatly facilitate the use of MEM techniques to the benefit of the LHC physics program. Once trained, fast-forward execution of the MEM DNNs would be used to rapidly develop the analysis as it evolves. A related benefit is preservation and portability of the approximated MEM calculations, providing a *sustainable* Matrix Element Method.

[Describe specific R&D items]

2.5 Monitoring Detectors, Hardware Anomalies and Preemptive Maintenance

Data-taking of current complex HEP detectors is continuously monitored by people taking shifts to monitor the quality of the incoming data. Typically, hundreds of histograms have been defined by experts and the shifters are alerted when deviation with respect to a reference occurs. It regularly happens that a new type of problem is unseen in a timely manner because it has not been foreseen by the expert. A whole class of

Machine Learning algorithms called anomaly detection can be used to cope with this problem. Such algorithms are able to learn from good data and alert when deviation is seen; one difficulty being that normal drifts in environmental conditions can induce normal drifts in the data. By monitoring many variables at the same time and spot correlated changes, they would be sensitive to subtle signs forewarning of imminent failure, so that preemptive maintenance can be scheduled. Such techniques are already used in the industry.

Beyond just reporting a problem, the natural next step is to connect anomaly detection algorithm to an IA algorithm which would trigger the appropriate action: restart an online computer, contact directly an on-call expert...

In the long term, the hardware and data structures of future detectors should be designed upfront to facilitate the operation of anomaly detection algorithms.

2.6 Learn and Reject the Standard Model

New physics may manifest itself as unusual or rare events. One approach is to accurately identify the Standard Model processes and search for anomalies. Classifying the Standard Model events is challenging task, as it consists of many complicated physics processes, but machine learning algorithms are well-suited for this multi-class classification problem. Once an event is classified as likely a known physics process it can be filtered out and remaining events can be analyzed further. An additional approach is to apply unsupervised machine learning techniques to cluster events together and can be further applied to the filtered events. Such an approach would also possibly identify detector problems and would be a useful in that regard.

2.7 Resource Optimization

Computing systems for LHC experiments developed together with Grids worldwide. The data volume in Grid transfers is only one of the aspects of the data challenges posed by Run-2 activities to the current experiment's computing system. In several areas of computing operations, it becomes clear that margins of optimization may come from a deeper understanding of resource utilization both the Grid components and each layer of the experiment-specific software stack. The enormous among of metadata information collected by application stack components, e.g. job information, failures, file accesses, etc., represent a significant challenge.

In the current infrastructure LHC experiments relies on in house solutions for managing their data. While these approaches work well the usage of data insight can automate and improve the overall system throughput and reduce operational costs.

Machine Learning studies can be applied in many areas of computing infrastructures, workflow management and data management. For instance, optimization of dataset placement and reducing transfer latencies can lead to a better usage of site resources and an increased throughput of analysis jobs.

The usage of opportunistic resources provided by commercial cloud providers becoming a standard practice within HEP community. This brings new challenges in scheduling resources in the most cost-efficient way where ML can provide insight on best ways to fit the processing needs to the available resources within constrained budgets.

Finally, networks are going to play a crucial role in data exchange across Tiers and data delivery to scientific applications in HL-LHC era. The network-aware application layer and tuning its configuration may significantly affect experiment's daily operations. The usage of ML may be applied to the following areas: network security by identifying anomalies in network traffic; predicting network congestion; bug detection via analysis of self-learning networks, and WAN path optimization based on user access patterns.

2.8 Control and Monitoring of Production Workflows

The idea is to apply ML to predict specific workflow or computing site behaviour or system performances, and feedback this into an active and adaptive modelling of LHC experiment workflows.

Currently, there is plenty of data on computing operations archived, but rarely accessed that can be used to predict various quantities such as data on-site performance and failures, services behaviour, data transfers and access to datasets, what information is used within datasets.

2.9 R&D Challenges

As we strive to use information that is rawer and further upstream in the event processing chain, we challenge our present computing models and computing platforms. The size of the data will present a significant challenge as well as the ability handle increasingly complex machine learning models required to achieve top performance.It is likely that combinations of CPUs and GPUs will allow us to survive in the next 5-10 years, but disk space is going to be a very hard barrier.

3 Machine Learning Software and Tools

Machine learning does not exist without software. There are a large variety of algorithms written in different programming languages and general software frameworks that combine many classes of methods into one package. The following sections focus on specific topics and challenges related to machine learning software design in HEP.

3.1 Software Methodology

Presently, there are two machine learning software methodologies in high-energy physics. The first approach focuses on HEP-developed ML toolkits, such as the Toolkit for Multivariate Analysis (TMVA) in ROOT, while the second approach relies on externally developed software, of which there are many examples. Historically, a variety of approaches and competition among them has led to important breakthroughs in the field. On the other hand, having too many choices increases repetition and leads community segmentation and possible issues with reproducibility.

3.2 Programming Languages

Although particle physics has been reliant on C++ over the past decade, the machine learning community is increasingly leaving C++ behind towards a python-based ecosystem of tools. Crucially, several of these python-based tools have an optimized C/C++ backend. Moreover, ML tools often provide different sets of APIs to develop and train the models in one language, and various bindings to use trained models in another programming language. This can be adapted as a model for certain HEP applications, e.g. using ML models in trigger systems.

3.3 I/O

The shear amounts of data accumulated by HEP experiments require a close look at data access optimization. When applying ML techniques to these data, efficient I/O becomes critical, especially for what concerns data training. It is worth to mention that I/O performance is very dependent on data formats. Moreover, support for reading data in different formats, e.g. row- or column-wise, can be required for certain use-cases. In particular, data streaming, locality and partitioning will impact training and cost of data processing.

Exploration of new file systems and methods to improve I/O limitations become important. We hence foresee that the following R&D studies should take place:

- Explore new file systems to assess I/O limitations;
- Use alternative approaches such as Google BigQuery to explore various data access patterns;
- Explore parallel data processing platforms such as Apache Spark for ML training.

3.4 Parallelization

Training ML algorithms takes a significant amount of time and parallelization at various levels is desired. For instance, the model parallelism addresses parallelizaion of the computations within a single model, while data parallelism targets processing phase of the training, both in terms of data partitioning and model training using

distributed workers. These approaches can be beneficial in particle physics trigger systems where stringent latency requirements impose constraints on the type of the algorithms.

3.5 Software interfaces to acceleration hardware

Machine learning software must be capable of using hardware accelerators such as FPGAs, GPUs, TPUs and many-core systems in general. At the same time, ML users should not be forced to write platform-dependent code. We need various interfaces to different hardware architectures in order to make efficient use of provided computing resources.

3.6 Interactivity

Availability of interactive frameworks, for example Jupyter notebooks, allows for rapid prototype development and testing of ML tools. Such frameworks also ease the connection between the description of models and the data, providing straightforward means of visualizing models and data [link to the Visualization Group]. Initiatives in HEP have started exploring interactive frameworks [link to CERN's SWAN].

4 Internal and External Machine Learning Tools

4.1 Internal and external ML tools

As described in section **??**, internally developed tools such as the Toolkit for Multivariate Analysis (TMVA) have been developed to apply a variety of machine learning algorithms to HEP challenges. Currently, most published HEP analyses with machine learning have made use of TMVA. There are also tools developed in HEP, such as NeuroBayes and RuleFit, which have gained popularity outside of HEP.

At the same time, the ML landscape has evolved and many different ML tools and toolkits have emerged and gained popularity. There is a growing number of published results based on externally developed tools. The latter, often developed directly by industry for specific applications, are constantly undergoing development, incorporating the latest algorithms from academia. Currently, both internal and external tools are used by the HEP community. TMVA has also undergone significant development in recent years.

This begs the question: what aspects of ML development and use should the HEP community focus on in the next 5-10 years. There are several aspects to consider including data formats, community size, and interfaces.

4.1.1 Machine Learning Data Formats

Unfortunately, HEP and ML communities currently make use of different data formats. HEP heavily relies on the ROOT software framework for data storage, data processing, and data analysis. The machine learning community uses a large variety of formats, as shown in Figure 1. This figure also shows the relationship of machine learning data formats with ROOT: ROOT file format is very flexible, though requires a significant investment to properly use. Table 1 summarizes the current machine learning tool kits and file formats they use.

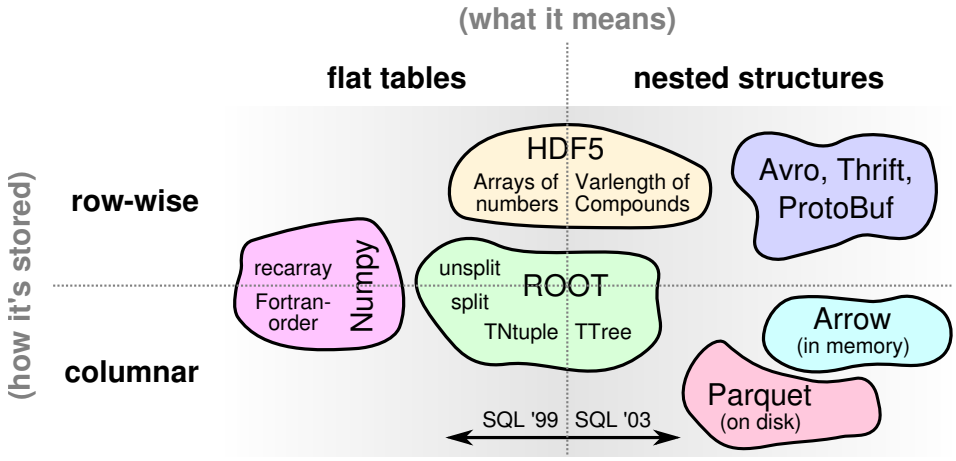

Figure 1: Existing data-formats used by ML communities (diagram is courtesy of Jim Pivarsky).

Table 1: This table lists various data-formats (rows) and ML tools (columns). The ✓indicates that there is a native solution, while × means that conversion is from one data-format to another is straightforward. The following notations has been used to denote the data-formats: **T** Trees, **F** flat tables, **M** sparse matrices, **R** row-wise arrays, **C** column-wise arrays **S** static data structures

	TMVA	TensorFlow	Theano	Scikit Learn	R	Spark ML	VW	libFM	RGF	Torch
ROOT [**T, C**]	✓									
CSV [**F**]		✓	✓	✓	✓	✓	×	×	×	✓
libSVM [**M**]							×	✓	×	
VW [**M**]							✓			
RGF [**M**]									✓	
NumPy [**R**]		✓	✓	✓	✓	✓	×	×	×	✓
Avro [**S, R**]					✓	✓				
Parquet [**S, C**]					✓	✓				
HDF5 [**S**]										✓
R df [**R**]					✓					

The de-facto HEP data format is ROOT, used by current HEP experiments and software systems. Although it is very well suited to store and process physics events, it is seldom used outside of HEP.

4.1.2 Desirable HEP-ML software and data format attributes

A desirable data format should have the following attributes: high read-write speed for efficient training, sparse readability without loading entire dataset into RAM, compression and common use by the machine learning community.

HEP machine learning applications require high performance and flexible algorithms to address the variety of use cases. Some applications, such as triggering, also have to work under tight latency constraints of the order of a few microseconds and below. The data sets are extremely large, which comes with I/O challenges described in section 3.3. This is expected to become even more challenging, as the LHC continues to ramp-up and deliver increasingly large amounts of data. All of these constitute requirements on ML tools used by the HEP community.

There are further requirements. The HEP community, for example, is looking to elevate Python to a first class library. Machine Learning tools use a number of languages as well as data formats. If use of the external tools is important, it will be important to offer support for these languages (and data formats). Finally, the result of the training must be imported back into the HEP world. This might involve supporting C++ converters or similar tools to make sure the training result can be efficienctly evaluated.

Advantages of using the **external tools** are the size of the community that uses and supports them, being able to easily keep up with progress in the industry and profit from the forefront of the ML research. A disadvantage of using external tools: too many choices that are not guaranteed to be supported over the lifetime of particle physics experiments, difficulty of adaptation to HEP specific requirements not among the priorities of the ML community.

Advantages of using **internal tools** are decisions about long-term support remain in the community, can be adapted to specific needs of HEP. Disadvantages include challenges in incorporating new algorithms and ideas on a timely basis and possible lack of resources for long-term maintenance.

4.1.3 Interfaces and middleware

One approach to bridge the gap between internal and external tools is by building interfaces. A number of interfaces have already been built between TMVA and external machine learning tools, allowing for their use and direct comparison between their performance. Another approach is building middleware solutions that export HEP-specific formats like ROOT to formats used by external machine learning tools. The existing middleware solutions are shown in Table 2.

]

The fast paced improvements of external tools and the large number of researchers and practitioners working on them means that it will be almost impossible for the HEP community to keep up with the improvements. As a result, external machine learning tools will be used by HEP physicists to maximize their physics output.

Approaches to bridge the different languages and data formats inside and outside HEP include providing interfaces or building middleware solutions that translate HEP-specific data formats to those used in external ML tools ("bridges and ferries"). It is a topic of research to determine the most efficient solution.

Table 2: Existing middleware solutions to translate ROOT data-format into other ML data formats.

PyROOT	Python extension module that allows the user to interact with ROOT data/classes [3].
root_numpy	The interface between ROOT and NumPy supported by the Scikit-HEP community [4].
c2numpy	Pure C-based code to convert ROOT data into Numpy arrays which can be used in C/C++ frameworks [5].
root4j	The hep.io.root package contains a simple Java interface for reading ROOT files. This tool has been developed based on freehep-rootio [6].
root2hdf5	Converts ROOT files containing TTrees into HDF5 files containing HDF5 tables [7].

5 Computing Resources

A typical HEP data model consist of a hierarchy of increasingly refined data stores. Each store provides an increasingly refined view of a list of "Events", self contained records capturing the state of the detector at the time at which an interesting particle interactions occurs. At the bottom of the hierarchy is the raw data store, a bytestream contaning the data as they were readout by the detector electronics, at the top are multiple "high-level" physics objects data stores, containing objects like electrons or jets, providing descriptive information about the quality and topology of physics events and is immediately ready for data analysis. These data stores are processed by independent copies of identical code processed in batch computing queues. The result of this processing is filtered data and extracted physics parameters.

At present, training of Machine Learning algorithms is done using dedicated or private resources of varying configuration and processing power, depending on the size of the data and complexity of the algorithm. The evaluation of algorithms, for a given event, is performed on a single core usually producing some single discriminator or regressor output. In order to progress to evaluation of complex machine learning models that implement pattern recognition, more computing power is needed in both the training and evaluation stages since larger amounts of data are needed to feed models with tens (or hundreds) of thousands of parameters. This implies expansion of the current computing model to include architectures that are well suited to machine learning such as MIC, GPUs, TPUs, etc. and eventually a departure from the single-core or few-core jobs. These architectures significant speed improvement in computation for both training and evaluation of ML algorithms, but require dedicated hardware, drivers, and software configuration.

Similarly, the locality and bandwidth of large data stores will need to be optimized in order to avoid bottlenecks in training and evaluation for analysis. The optimization of data placement and the need to use dedicated hardware indicate a transition to HPC, or HPC-like, architectures may be needed to achieve the best performance for these future algorithms. Given the presently large amount of synergy with the direction of industry in this respect, use of commercially available resources should be considered for future computing models in addition to vertically integrated models like the tiered data stores mentioned previously.

In the following sub-sections we will discuss the resource needs for the physics drivers mentioned earlier: Fast Simulation, Real Time Analysis, Object Reconstruction, Object Identification, Object Calibration, and Event Analysis, the limitations of the current computing model on achieving these physics drivers, and how those needs can be met in the future.

5.1 Resource Requirements

5.1.1 Resource requirements for training and inference

The current popularity of deep learning methods is to large extent due to the possibility to train deep networks in a reasonable amount of time thanks to large scale parallelism. Training requires repeated simultaneous access to many data elements. Moreover, in future experiments, the datasets used to train ML algorithms will be of order petabytes or larger, requiring optimized access to those data to feed the algorithms.

In contrast, inference can be an operation applied to a single data element at a time and needs to be performed only once. Inference has less demands on I/O and is limited only by the computing power and model complexity. Because inference may need to happen in real time, time constraint is the main challenge.

5.1.2 HPC: MIC, GPUs and TPUs

MIC, GPUs, and TPUs are useful for speeding up the computation of most machine learning algorithms. These include Deep Learning Neural Networks, GANs, Autoencoders and in particular help address the following physics needs: object and event reconstruction, real-time analysis, fast simulation, event analysis.

HPC also provides computing resources already integrated with MIC and GPU accelerators running massively multi-threaded jobs with common memory pools as opposed to many independent single core jobs. Initial

exploratory studies have been conducted on running ML applications for HEP and are promising. In order to deploy machine learning techniques effectively in future experiments, investment needs to be made to expand the availability of high-performance resources in HEP.

5.1.3 FPGAs

FPGAs allow an efficient and low-latent application of machine learning algorithms directly at the level of hardware. The following algorithms are suitable for FPGAs: boosted decision trees, random forests and rules ensembles. FPGA-based applications are useful for real time trigger systems where low-latency inference is needed. There are already boosted decision trees applied in the LHC trigger systems. Further research and development in this area is needed to apply more advanced machine learning techniques like deep learning in the hardware.

5.1.4 Data Storage and Availability

Data storage will have a major impact on machine learning applications. Currently, it has been possible to take advantage of existing increases in statistics for simulation by non-ML uses. Further progress in machine learning may require more simulated data than what is available from non-ML use-cases and how to produce and store it becomes a challenge (Link to Data Storage WG)

The data availability at PByte/EByte scale represents a challenge for ML community. A good solution must provide access to large data volume for hundred or thousand of users simultaneously. The success of Apache Spark and Google BigQuery platforms may serve as a model. Data streaming, transformation and readout in mini-batches may be required to train HEP ML models over large datasets. Therefore support of object, column, row wise data should be available as described in section 4.1.1.

5.1.5 Software deployment

To efficiently use resources described in previous subsections, machine learning software needs to be available on these resources. Tools like containers can be useful to provide homogeneous software environments across the different systems.

5.1.6 Opportunistic Resources

The current HEP model is based on tiered structure where computing resources are mostly large data centers providing CPU resources for collaboration. Although existing resources are gradually moving towards supporting GPUs, it is unlikely to reach all HEP computing centers in the near future. Therefore opportunistic resources are a possible option for training machine learning applications. Costs of these resources should be compared with the costs of procuring these resources independently.

- lxplus, lxbatch, analytix clusters at CERN or lpc at FNAL
- Tier 1 and Tier 2 centers
- HPC centers
- Experiment trigger farms during accelerator downtime
- cloud providers

Currently, cloud solutions provided by the industry run ML workflows on dedicated hardware and offer interfaces to train machine learning models. The scientific community should work closely with cloud providers in order to harmonize our analysis computing needs and data access patterns with their business models.

5.1.7 Machine Learning As a Service

Current cloud providers rely on machine learning as a service model allowing for efficient use of common resources and use interactive machine learning tools. Specialized HEP services for interactive analysis may play an important role in adaptation of ML tools within HEP workflows.

In order to use these tools more efficiently, sufficient and appropriately tailored hardware resources are needed.

5.2 Resource Evaluation

Existing and upcoming resources such as GPUs, TPUs, MIC available through HPC centers or otherwise, need to be evaluated in the context of realistic benchmark particle physics applications.

In particular, the evaluation the processing speed of ML techniques for inference is the major determining factor for applications like real time processing. If different ML techniques can achieve equivalent physics performance but require different processing power to achieve the goal, can achieve most of some ultimate performance, then it is of foremost importance to quantify what is bringing the performance gains and what level of performance is maintainable given some computing power or latency requirement.

6 Collaborating with other communities

6.1 Introduction

Before building collaborations, it is important to have a coherent strategy and vision of how to engage the ML community. In the future, we would like to achieve vibrant collaboration between domain experts in data science and high-energy physics fields speaking a common language and working together to further science. The HEP community has much to gain from greater interaction within the ML community - new research directions and applications of machine learning domain expertise, novel algorithms, and direct collaboration on HEP challenges.

The ML community can benefit as well from greater interaction with the HEP community: a diverse set problems with an assortment of unique challenges, both in scale and complexity, and a large community of researchers that can expand machine learning horizons by contributing directly to the problems relevant to both communities. For example, the treatment of systematic uncertainties is an important consideration in HEP and of growing importance to the ML community. By working together on common challenges further understanding of such topics can be acheived.

There are existing examples of collaboration between HEP and ML that have produced fruitful results mostly through local connections [TODO references]. Both communities would benefit from expanding this further. The HEP community has a problem domain that offers rich avenues for intellectual reward. Discovery science provides a challenge that could attract brilliant minds eager to push the boundaries of scientific understanding of nature. Additionally, after investing heavily into producing highly-detailed and realistic simulations, the HEP community can provide the Machine Learning community with datasets with high statistical power to test algorithms and develop novel ideas.

Particle physics domain knowledge can present a barrier to collaboration. The HEP community needs to present its challenges in a way the ML community can understand. This may involve stripping the domain knowledge entirely, or retaining necessary information with clear and concise explanations as to its relevance. On the other hand, ideas and solutions provided by the machine learning community for the HEP community should be presented in an equally understandable way for HEP scientists without in-depth ML knowledge.Machine learning has a significant amount of domain knowledge. By understanding and speaking the same language, the HEP community can better collaborate and find solutions to existing and future challenges

6.2 Machine Learning Challenges (Competitions)

To engage the wider ML community, challenges such as the Higgs Boson Machine Learning (2014) or the Flavor Physics Challenge (2015) have been organized on Kaggle. This kind of challenges draw considerable attention from the Machine Learning community and new such challenges should be organised in the future.

The organisation of a challenge requires to build a well documented dataset, associated to a starting-kit giving out a simple solution and the score used to rank the participants. It forces the challenge organisers to simplify the problem as much as possible, while retaining its intrinsic difficulty. To have a single score as a figure-of-merit is part of the required simplification.

The drawback of challenges is that once they are launched, participants want to win the challenge, not develop useful algorithms for HEP, so that the winning solution might be useless. In addition, there is no incentive for collaboration between participants. It is thus important to foresee upfront forum and post-challenge workshop(s) where a diversity of competitive algorithms are exposed. Significant manpower should be foreseen to re-import the best algorithms into the full complexity of the problem. The challenge dataset and evaluation metric should be released publically so that further developments can continue.

6.3 Collaborative Benchmark Datasets

There is a strong incentive for HEP to develop public Benchmark Datasets, beyond just challenges. Whenever engaging a Machine Learning expert, being able to give him access to a dataset makes the discussion much more concrete and productive. Even within the HEP community a common Dataset allows to compare algorithms with a much better accuracy compared to papers, each using one's experiment data. These dataset could be built on purpose from public simulation engine, or released by experiments within the bonds of their data access policy. Even a small subset of an experiment simulated data can be the base of a very valuable dataset.

In High Energy Physics, only members of an experiment are allowed to access this experiment dataset, which hampers collaboration between physicists and with Machine Learning scientists. While a public Benchmark Dataset can be used for collaboration on cutting-edge algorithms and for dedicated publication. The same Benchmark Datasets can also be used for teaching, tutorials and training.

It should be noted that other scientific field like astronomy has had the policy to completely release its data after a short period of time, which has allowed collaboration between astronomers and Machine Learning to bud early.

The HEP community should organize and curate a variety of such Benchmark Datasets covering its current challenges. To improve reproducibility of results and algorithm comparisons, some of the data used for evaluation should be kept private.

To be maximally useful, the subsequent guidelines should be followed:

- Simplify the Dataset as much as possible (e.g. release fully calibrated quantities (e.g. 4-vectors) rather than raw data and calibration software)
- Document the Dataset to make it understandable by a non-HEP expert
- Create methodology and metrics (which can be several) for evaluation proposed solutions, and document them.
- Prepare an integration plan for incoming ideas and solutions
- Feedback results of successful applications

Another value of Benchmark Datasets is that they can be reused for completely different objectives. For example a public calorimeter simulation dataset foreseen to study computer vision algorithms for particle identification, can also be used to study energy regression, and also for simulation with Generative Adversarial Network.

6.4 ML Academic outreach

Conferences and workshops are a core aspect of the academic ML community, and organizing or contributing to key conferences is a means of gaining interest. Organizing sessions or mini-workshops within major conferences, such as NIPS, would increase the familiarity of HEP within the ML community and could jump-start future collaborations. At the same time inviting ML experts to HEP workshops dedicated to areas where external expertise is needed can foster greater long-term collaboration.

- Organize workshops and conferences open to external collaborators to discuss the applications, algorithms and tools
- Organize thematic workshops around topics relevant to HEP, for example "sparse data" workshop, which would attract plenty of interested people from other domains
- Engage machine learning experts by creating prestigious fellowships and software-related institutes. These can fit into existing initiatives, such as DIANA-HEP, Marie-Curie Training Networks and possibly be privately funded too.

6.5 Theoretical physics outreach

The theoretical physics communities have their own challenges where Machine Learning can have a deep impact. For example:

- Studying theoretical models with hundreds of parameters.
- NNPDF – Neural Network Parton Distribution Functions
- Quantum machine learning

- Links to low-level VM representation groups
- LLVM (ROOT uses LLVM)
- LIBXSMM by Intel
- XLA by Google

6.6 Science outreach

HEP should reach out to other scientific communities with similar very large datasets, for instance including astrophysics/cosmology (LSST), medium energy nuclear physics (JLAB). We could either host their data through the organization mentioned above and attempt to leverage interest in their data to attract researchers to our own datasets, or we could seek more active partnerships to work on each other's public datasets to better cross-germinate ideas, techniques, and algorithms.

6.7 Industry Engagement

Industry has been leading HEP when it comes to the development and adoption of machine learning techniques. The growing field of « Big Data » has led to tools and techniques for extracting information from datasets in industry and other academic fields that now rival high energy physics samples in size. One of the most promising areas of development is in the adoption of dedicated specialized hardware and high performance co-processors. GPUs, FPGAs, and high core count co-processors all have the potential to dramatically increase performance for specific sections of the HEP applications.

One of the challenges in the path towards adoption is gaining the human expertise for development and implementation. Industry brings both specific technology opportunities, but equally importantly access to specialised expertise that would be difficult to hire and support ourselves. The HEP environment has the potential to improve and evolve machine learning techniques with large and unique datasets and strict technical requirements.

There are at least five specific areas of development that groups from industry have expressed an interesting in pursuing with HEP or have already started. Automated resource provisioning, data placement, and scheduling are similar to industrial applications to improve efficiency and would be an excellent joint project that would bring short term results. Applications like data quality monitoring can be automated using techniques developed for other industrial quality control applications. This should also be a relatively straightforward implementation that saved effort. Detector health monitoring and scheduling preventative maintenance is being explored now, and is very similar to other types of industrial controls. There are two more forward looking areas which are using computing visualization techniques for object identification and real time event classification. These two are in many ways the most ambitious and are interesting to industry because they put the existing capabilities both in terms of complexity and latency.

6.7.1 CERN openlab

Engagement with industry in Machine Learning, as well as in a variety of technical computing and software topics, has also come through CERN openlab. CERN openlab is a unique public-private partnership that accelerates the development of cutting-edge solutions for the worldwide LHC community and wider scientific research. CERN openlab has established the infrastructure to maintain non-disclosure agreements, to arrange ownership of intellectual property, to define projects and hire supported effort, and generally manage the interactions between the research and industry worlds.

Through CERN openlab, CERN collaborates with leading ICT companies and research institutes. Within the CERN openlab framework, CERN provides access to its complex ICT infrastructure and its engineering experience — in some cases even extended to collaborating institutes worldwide. Testing in CERN's demanding environment provides the collaborating companies with valuable feedback on their products, while enabling CERN to assess the merits of new technologies in their early stages of development for possible future use. This framework also offers a neutral ground for carrying out advanced RD with more than one company.

CERN openlab is continuing its work to support CERN's research community, with a particular focus on the upgrades to the LHC and the detectors that will be carried out during LS2 and LS3. With the data rates from the experiments set to increase significantly, efforts have been focused on supporting the work to overhaul and modernise their data-acquisition systems, while also ensuring that the maximum benefits are gained from the hardware available to CERN's teams by making sure the software running on it has been fully optimised. Efforts have now begun to identify the ICT challenges that will be tackled in CERN openlab's sixth phase (2018 to 2020).

One of the most significant technical challenges facing CERN is how to provision computing for the LHC experiments and the accelerator in Run3 and Run4. In order to tackle said challenge, CERN openlab Phase

VI will choose projects that are intended to help close the gap and deliver the evolutionary and revolutionary changes needed for the success of the program. The exploration of new applications of Machine Learning techniques aligns with the goals of industry and the next phase of CERN openlab.

The work areas are divided into three sections that will be expanded below: research and development (RD) for data centre technologies and infrastructure, RD for software and computing performance, and RD for machine learning and data analytics. In particular, for its Phase VI, CERN openlab is in the process of investigating the opportunity of establishing a number of key projects in machine learning for data taking, triggering, simulation, reconstruction, detector and data quality monitoring with leading companies in deep learning technologies.

6.8 ML community at large outreach

- Appear on Podcast for Machine Learning to explain how data science is used in HEP (at CERN / Fermilab)
  - There is an abundance of machine learning podcasts with large base of listeners:
    * Linear Digressions (co-hosted by former ATLAS Ph.D. Katie Malone)
    * Partially Derivative
    * Talking Machines
    * Data Skeptic
    * Becoming a Data Scientist Podcast
    * Not So Standard Deviations
    * This Week in ML & AI

    Podcasts have shown to be a great vehicle for reaching a large audience. The listener base of common podcasts consists of machine learning experts from the community at large, not just high energy physicists. Listeners are keen to consume material that is outside of their immediate problem domain in a way that is easy to digest. Since people generally seem very interested in how physics is using technology to advance science, we have a captive audience. There have already been physicists from the community going on podcasts to talk about machine learning in physics to great welcome. Some examples are (Partially Derivative Episode: "Particle Physics and Machine Learning at CERN with Michael Kagan") as well as applications of programming and technology in physics (Talk Python To Me Episode #29: "Python at the Large Hadron Collider and CERN" with Kyle Cranmer).

    Bringing together tangentially related fields is commonplace in ML podcasts. For example, podcasts such as This Week in Machine Learning (TWiML) features machine learning experts across a wide variety of problem domains, spanning topics from a genomics to robotics to astrophysics.

    From our perspective, the amount of work involved in appearing on a well known podcast is much lower than another form of media, such as creating a YouTube series. Podcasts typically consist of a casual conversation from 30 minutes to an hour, while YouTube videos usually require multiple takes (translation: a lot of time) for one final recording. Therefore, there is a high yield in the form of listeners vs time investment required to release a podcast.

    Finally, appearing on podcasts can enable the HEP to foster an ongoing conversation about ML in HEP in the outside community. Podcast listeners expect a new episode to come out on a weekly or monthly basis, and podcasts are often followed by further conversations with their peers about the podcast content. Being able to reach a larger audience on a regular basis sets would enable the HEP community to have higher visibility across the ML community at large.

- There are a variety of channels which can be leveraged to increase the visibility of our problems and research opportunities in the ML community. These can be popular forums (such as reddit and hackernews), personal or official blogs, social media (such as twitter), and direct contact with influential personalities (such as Ian Goodfellow).

- Do presentations at Machine Learning Meetups across the world
  - Goal: To generate awareness of ML methods in HEP, engage community, foster cross pollination of ideas between HEP + industry
  - Meghan belongs to a few of them:
    * NYC: https://www.meetup.com/NYC-Machine-Learning/
    * Berlin: https://www.meetup.com/Advanced-Machine-Learning-Study-Group/
    * SF: https://www.meetup.com/SF-Bayarea-Machine-Learning/

- Create outreach-style blog posts to explain our public work in a way that is easy to understand by the public.
- Reach out to Data Science community – both academic and private sector.
  - https://cloud.google.com/genomics/

7 Training the community

In order to address the communication barrier and to speak the same language, the HEP community should be trained in ML concepts and terminology as part of a standard curriculum. The training should focus on well-maintained and well-documented software packages. It should provide lectures on general ML concepts and tutorials on specific tools, in addition to explaining a framework's API and providing examples. More details on the foreseen training methods are provided in the next section.

As discussed in the chapter on training, ensuring the development and availability of resources for knowledge transfer is essential to ML. The following resources are useful

8 Roadmap

:

When thinking the incorporation of ML into particle physics experiments, two time lines must be kept in mind: the first defined externally by the LHC schedule and funding agencies, and the second defined by the experiment need for extensive validation of the algorithms necessary to process and interpret data into statements about nature.

The current LHC schedule, which the funding agencies are matching, has Run 3 starting in 2021 and the HL-LHC, if approved, starting in 2026. As software processes and algorithms are re-imagined, their implementation must fit into these dates if they are to maximize their benefit to the physics. In order to fit this schedule, a newly proposed implementation or idea would need to show a demonstration in 2018 to prove viability. Two years later, in 2020, when HL-LHC computing TDR's will be drafted, the idea needs to have progressed to a level of maturity that it can be included in the TDR. The idea, turned project, should then be further refined towards a large scale test around the middle of Run 3, about 2022. Run 3 is scheduled to end in late 2023. The project must then be adapted to the HL-LHC software and physics analysis environment as it will be relied on by the experiment.

The path of taking a ML idea from conception to community-wide acceptance and deployment will entail several stages, as appropriate. As a function of

There are ample opportunities to make the process more efficient. For example, in many steps having common datasets, which are often large, etc.

1. Problem formulation and dataset preparation: Problem formulation is the first step in building an ML algorithm. The inputs and desired output of the ML need to be established. The training and validation datasets must be identified and perhaps created (in most cases simulated). In many cases, these datasets are large, and resources must be identified to possibly create and store the data. In most cases, the data needs to be processed into a form suitable for input into the algorithm. Since these steps are often lengthy, common datasets with well-defined problems are very helpful.

2. Feasibility/Demonstration: Given a dataset, appropriate ML algorithms need to be investigated and evaluated for ability to solve the problem. In some cases, such studies can be preformed on simplified datasets.

3. First application: An application of the solution to one or few specific physics analysis where the ML technique significantly improves the physics result. Here the incorporation of the technique into the workflow will likely be very specific to the application and require significant manual intervention.

4. Scaling/Optimization: Evolving from a demonstration to a general or production solution requires use of realistic datasets (e.g. full detector simulation, noise, etc.). Furthermore, the solution will also require optimization to achieve nominal physics and computing performance. A good practice at this point would be to apply the solution to a specific physics analysis. This stage is likely require significant computing resources to scale solutions full detector, use large datasets, and perform scan and optimize hyper-parameters.

5. Integration/Validation: The solution needs to be incorporated into the experimental software and workflow and validated.

As an example, consider the simulation physics driver. An effort has recently started to build generative models that could significantly accelerate simulation of particle showers in calorimeters. These early efforts are based on simplified datasets specifically created for this problem, without the irrelevant complications of realistic data and limited to a small section of calorimeters. The first papers[] use GANs to generate calorimetric data which are reasonably faithful, but still require tuning. The next step involves exploration of DNN architecture and systematic hyper-parameter scans on HPCs to achieve the required performance. The technique can applied to to searches at LHC that involve boosted objects, where the required simulation samples require CPU intensive full GEANT-based simulation and are therefore limited in statistics due to resource limitations. The process of employing the new technique in a publication will illicit scrutiny by the full experiment, effectively validating the technique. Once the technique is accepted, it can be then generalized beyond this first application and then incorporated into the experiment's software for use by others. Finally, as the technique is applied to an increasing number of physics analysis, the technique will be incorporated into the experiment's production workflows.

8.1 Machine Learning Workflows

The HEP computing model relies on high throughput computing (HTC). This means we operate on data under the assumption of independence between events, and divide our data across a very large number of homogeneous compute nodes that all operate in isolation. Results are drawn together as a sum in the end. This computing model is not natural for training machine learning algorithms where, in the simplest model, one training node must iterate over the entire dataset to fully leverage the statistical power it contains.

The next sections outline several important workflows - how individuals and groups accomplish machine learning tasks today. The purpose is to list all the steps and discuss needs for each of them and their possible future evolution.

Classical Training WorkFlow

1. Conversion/Extraction Tools
    (a) Conversion to ML format
    (b) Framework to ML format
    (c) Read Root to in memory ML format
2. Pipelining Disk to Memory
    (a) Parallel io / processing in batches
        i. Processing on device
3. Machine (Deep) Learning Framework
    (a) Use broader community and contribute back
    (b) Areas
        i. Model definition
        ii. Training (e.g. parallel)
        iii. Model output format
        iv. Plotting tools
        v. Analysis tools (e.g. ROC curves)
        vi. Notebooks
4. Hyperparameter scan/optimization "framework"
    (a) Workflow
    (b) Metadata
5. Integration with experiment Workflow and Data Management System
    (a) Parallelization
    (b) HyperParameter
    (c) Model Parallel
    (d) Data Parallel
6. Output to Experiment

Production Training Workflow

1. Distributed within one site/HPC
2. Automation
3. Monitoring validation

Production Usage of Training

1. Workflow like calibration-
2. Conditions Like Database for NN storage
3. Network Inference Service (potentially over network) (e.g. TensorFlow Server)
    (a) Memory management

User Analysis Workflow

1. Leveraging Production Tools?
2. Lightweight Tools
3. Exporting of Models/Weights from production
4. Model Sharing

Software

1. Models Management
    (a) Providence *PC Provenance?*
    (b) Common representation of architecture and weights
2. Experiment SW Framework
    (a) Model inference
    (b) In framework training (e.g. for anomaly detection or rolling calibration)
        i. Iteration (same events for each epoch)
    (c) Condition like Model storage/booking service
    (d) Monitoring/validation tools
    (e) Better code framework automatic templating and/or FW autocompletion.
    (f) Better debugging tools: automatic parsing of logs, dynamic community driven FAQ (like StackExchange), etc.
3. Requirements to Workflow Management system
    (a) Allocation of multi-node / hybrid resources
4. Requirements to Facilities (link to Resources)
    (a) Specialized [mini]-HPC aimed at specified tasks (e.g. ML training)
    (b) High bandwidth between accelerators
    (c) High throughput IO
    (d) Support for different data access patterns
        i. Hyperparameter scan / Model parallel- same data on lots of nodes
        ii. Data parallel- different data on all

Author list

- Aaron Sauers
- Aashrita Mangu (CS)
- Adam Aurisano (NOvA)
- Adrian Bevan (ATLAS)
- Alessandra Forti (ATLAS)
- Alexander Kurepin (ALICE)
- Alexander Radovic (NOvA)
- Alexei Klimentov (ATLAS)
- Amir Farbin (ATLAS)
- Andrey Ustyuzhanin (Yandex, LHCb)
- Antonio Limosani (ATLAS)
- Ariel Schwartzman (ATLAS)
- Attilio Picazio
- Aurelius Rinkevicius (CMS)
- Ben Hooberman (ATLAS)
- Benedikt Hegner (SFT)
- Bob Stienen
- Claire David (ATLAS)
- Conor Fitzpatrick (LHCb)
- Daniele Bonacorsi (not only as CMS in this WG) (CMS)
- Dario Menasce (CMS and INFN)
- David Rousseau (ATLAS)
- Dick Greenwood
- Dorian Kcira (CMS)
- Douglas Davis
- Dustin Anderson (CMS)
- Eduardo Rodrigues (LHCb)
- Elias Coniavitis
- Federico Carminati (SFT)
- Fernanda Psihas (NOvA)
- Filip Siroky
- Gabriel Perdue (MINERvA)
- Gaurav Kaul (Intel)
- Giles Strong (CMS)
- Gilles Louppe (ATLAS)
- Gordon Watts (ATLAS)

- Graeme Stewart
- Hans Pabst (Intel)
- Harvey Newman (CMS)
- Helge Meinhard
- Horst Severini
- Ian Stockdale
- Igor Lakomov (ALICE)
- Ilija Vukotic (ATLAS)
- Jamal Rorie (CMS)
- Javier Duarte (CMS)
- Jean-Roch Vlimant (CMS)
- Jim Kowalkowski
- Jim Pivarski (CMS)
- Jochen Gemmler (Belle2)
- Johannes Junggeburth
- John Harvey (SFT)
- Jonas Eschle (LHCb)
- Jonas Graw
- Jordi Garra-Tico (LHCb)
- Juan Pedro Araque Espinosa (ATLAS)
- Karen Tomko
- Kevin Lannon (CMS)
- Kim Albertsson (ATLAS)
- Konstantin Kanishchev (AMS-02)
- Konstantin Skazytkin (ALICE)
- Kyle Cranmer (ATLAS)
- Laurent Basara
- Lindsey Gray (CMS)
- Lorenzo Moneta (ROOT)
- Louis Capps
- Lukas Heinrich (ATLAS)
- Luke Kreczko (CMS, LZ)
- Maria Girone (CERN openlab)
- Mario Campanelli (ATLAS)
- Mario Lassnig (ATLAS)
- Mark Neubauer (ATLAS)
- Martin Vala

- Matthew Feickert (ATLAS)
- Mauro Verzetti (CMS)
- Meghan Kane (SoundCloud, formerly @MIT)
- Michael Andrews (CMS)
- Michael Kagan (ATLAS)
- Michael Williams (LHCb)
- Michela Paganini (ATLAS)
- Michele Floris (ALICE)
- Mike Sokoloff (LHCb)
- Nicolas Köhler
- Nuno Filipe Castro (ATLAS)
- Paolo Calafiura (ATLAS)
- Paul Glaysher (ATLAS)
- Paul Seyfert (LHCb)
- Pere Mato (SFT)
- Piero Altoe (NVidia)
- Przemysław Karpiński (CERN openlab)
- Rob Kutschke (Mu2e, Intensity Frontier)
- Ryan Reece (ATLAS)
- Savannah Thais
- Sean-Jiun Wang (CMS)
- Sergei Gleyzer (CMS)
- Seth Moortgat (CMS)
- Sofia Vallecorsa (SFT)
- Stefan Wunsch (CMS)
- Steven Schramm (ATLAS)
- Taylor Childers (ATLAS)
- Thomas Keck (Belle2)
- Tom Hacker
- Uzziel Perez (CMS)
- Valentin Kuznetsov (CMS)
- Vladimir Vava Gligorov (LHCb)
- Wahid Bhijmi (Daya-Bay)
- Wenjing Wu
- Xavier Vilasís-Cardona
- Omar Zapata (http://oproject.org)

references

[1] S. Agostinelli et al. "GEANT4: A Simulation toolkit." In: *Nucl. Instrum. Meth.* A506 (2003), pp. 250–303. DOI: 10.1016/S0168-9002(03)01368-8.

[2] M. Paganini, L. de Oliveira, and B. Nachman. "CaloGAN: Simulating 3D High Energy Particle Showers in Multi-Layer Electromagnetic Calorimeters with Generative Adversarial Networks." In: (2017). arXiv: 1705.02355 [hep-ex].

[3] *PyROOT*. URL: https://root.cern.ch/pyroot.

[4] *root numpy converter*. URL: https://github.com/scikit-hep/root%5C_numpy.

[5] *c2numpy*. URL: https://github.com/diana-hep/c2numpy.

[6] *root4j*. URL: https://github.com/diana-hep/root4j.

[7] *root2hdf5*. URL: http://www.rootpy.org/commands/root2hdf5.html.