CWP WG - Machine Learning

For further edits please use: https://www.sharelatex.com/project/593a919f0f690d906afbfc76

Participants:

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)

For further edits please use: https://www.sharelatex.com/project/593a919f0f690d906afbfc76

Attilio Picazio
Aurelius Rinkevicius (CMS)
Ben Hooberman (ATLAS)
Benedikt Hegner (SFT)
Claire David (ATLAS)
Daniele Bonacorsi (not only as CMS in this WG) (CMS)
David Rousseau (ATLAS)
Dick Greenwood
Dorian Kcira
Douglas Davis
Dustin Anderson (CMS)
Elias Coniavitis
Federico Carminati (SFT)
Fernanda Psihas (NOvA)
Filip Siroky
Gabriel Perdue (MINERvA)
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
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)
Konstantin Kanishchev (AMS-02)
Konstantin Skazytkin (ALICE)

For further edits please use: https://www.sharelatex.com/project/593a919f0f690d906afbfc76

Kyle Cranmer (ATLAS)
Laurent Basara
Lindsey Gray (CMS)
Lorenzo Moneta (ROOT)
Louis Capps
Lukas Heinrich (ATLAS)
Luke Kreczko
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
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)

For further edits please use: https://www.sharelatex.com/project/593a919f0f690d906afbfc76

Vladimir Gligorov (LHCb)
Wahid Bhijmi (Daya-Bay)
Wenjing Wu
Xavier Vilasís-Cardona

Scope: Machine Learning algorithms play an important role in many facets of today's HEP data analysis, data-processing and detector applications. Machine-learning tools already form an important part of HEP software. To overcome the challenges related to data-processing and analysis of upcoming very large HEP data-sets, it is important to plan ahead for how HEP machine-learning software and tools develop. This group will work on both identifying the challenges related to machine-learning software in HEP and proposing possible solutions and a community roadmap towards better HEP-ML software.

Steps Towards a Roadmap

- First workshop in San Diego link
    - podA, podB, podC, podD
- Second workshop at IML in CERN link
- Third workshop at DS@HEP in May link
- HSF Workshop in Annecy in June

Questions/Challenges:

- How to use machine learning to do more physics within the existing computing and software budget and reduce the processing power, data transfer and storage space and human effort per unit physics?
- How to keep up with external software and hardware evolution in machine learning?
- How to best engage the machine learning community, develop interfaces to external machine learning capabilities and represent data in formats useful to the machine learning community
- How do we train our community in machine learning?
- Roadmap questions
    - How do we best learn from and interact with a rapidly changing ML community?
    - How can we best use domain knowledge?
    - How do we best contribute to the ML community?

For further edits please use: https://www.sharelatex.com/project/593a919f0f690d906afbfc76

1. Introduction

As we enter the post-Higgs discovery era of particle physics, our community faces new challenges brought on by increasing data sizes, volumes and levels of complexity. To address this challenge we need more powerful methods than currently available.

Machine-Learning (ML) offers a promising direction to explore solutions to this problem. ML methods are designed to exploit large datasets in order to reduce complexity and can be used to identify new features in data. 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.

1.1 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 particle and event identification, energy measurements 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.

For further edits please use: https://www.sharelatex.com/project/593a919f0f690d906afbfc76

2. 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.

2.1 History of HEP Machine-Learning Software

1991 - 1998

C. Peterson and T. Rognvaldsson, An Introduction to Artificial Neural Networks, Lectures given at the 1991 CERN School of Computing, University of Lund preprint LU TP 91-23, September 1991.

Early Feed-forward NN's B. Denby, Neural Computation 5 (1993) 505.

Kernel Density http://inspirehep.net/record/407093

Fisher discriminants & H-matrix http://inspirehep.net/record/1234704

1998-2005 Initial HEP Machine Learning Software development based on external algorithms lead to number of standalone packages (MultilayerPerceptron and others)

2005-2010 Abundance of comparably-performing methods lead to a consolidation into software toolkits such as Toolkit for Multivariate Analysis (TMVA) and StatPatternRecognition (SPR).

2008 SPR stops development and support, TMVA takes over

2010-2014 Wide use of TMVA in HEP, full integration in ROOT and switch to maintenance mode

2014-present Upgrade/modernization of TMVA, wider use of externally developed tools deep learning revolution.

2.2 Status of HEP Machine-Learning Software

What works:

- We have a good baseline with which any new machine-learning tools or methods can be compared with
- We are able to keep up and in some cases improve over external tools on HEP benchmarks

- GPU-capable deep-learning libraries
- ML is commonly used for event selections offline and also in software triggers
- First usages in event/track reconstruction and more “technical” areas: assignment of computing resources, anomaly detection, etc.

What does not work:

- Currently many tools store data in RAM, leading to problems for very large datasets.
- Not all external dataformats and options are fully supported
- Some model architectures and variants currently not supported

2.3 Software Methodology

Presently, there are two ML software methodologies in HEP. The first approach focuses on HEP-developed ML toolkits, such as 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.

- **Challenge: How to find balance between the two approaches.**

There is some recent convergence of the two approaches. Converters have been written for the reintegration of some externally trained models into HEP tools. Interfaces for more direct communication between HEP and external tools have also been developed. More effort is needed to extend these interconnecting efforts and integrate new useful tools when they become available.

- **L** Link to 2.6 Internal and external tools

2.4 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.

For further edits please use: https://www.sharelatex.com/project/593a919f0f690d906afbfc76

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
        i. HyperParameter
        ii. Model Parallel
        iii. 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:

Models Management
1. Providence
2. Common representation of architecture and weights

Experiment SW Framework
1. Model inference
2. In framework training (e.g. for anomaly detection or rolling calibration)
    a. Iteration (same events for each epoch)
3. Condition like Model storage/booking service
4. Monitoring/validation tools
5. Better code framework automatic templating and/or FW autocompletion.
6. Better debugging tools: automatic parsing of logs, dynamic community driven FAQ (like StackExchange), etc.

Requirements to Workflow Management system
1. Allocation of multi-node / hybrid resources

Requirements to Facilities   (link to Resources)
1. Specialized [mini]-HPC aimed at specified tasks (e.g. ML training)
2. High bandwidth between accelerators
3. High throughput IO
4. Support for different data access patterns
    a. Hyperparameter scan / Model parallel- same data on lots of nodes
    b. Data parallel- different data on all

2.5 Technical Considerations

2.5.1 Programming Languages

Although particle physics has been reliant on C++ over the past decade, the machine learning community is increasingly leaving C++ behind towards python-based tools and ecosystem.

2.5.2 I/O

Given the size of the data used for machine learning, efficient I/O becomes critical.

- **Challenge: how to improve I/O of ML software in HEP**

For further edits please use: https://www.sharelatex.com/project/593a919f0f690d906afbfc76

2.5.3 Parallelization

Training machine learning algorithms takes a significant time and parallelization is one of the important areas of ongoing research. Additionally, the inference of machine learning algorithms significantly benefits from parallelization as well. For example, in particle physics trigger systems the stringent latency requirements impose constraints on the type of algorithms that can be useful under those requirements and distributed algorithms play a key role.

2.5.4 Interactivity

Availability of interactive machine learning tools, for example via Jupyter notebooks, allows for rapid prototype development, interaction with the model and the data (connect with Visualization Group)

2.5.5 Software Interfaces to Acceleration Hardware

2.5.6 Sustainability

2.6 Internal and external ML tools

Contributors: Juan Pedro Araque, Nuno Filipe Castro, Thomas Keck, Stefan Wunsch, Igor Lakomov, Konstantin Skazytkin, Luke Kreczko, Przemysław Karpiński, Claire David, Hans Pabst, Attilio Picazio, Giles Strong, Gordon Watts, Lorenzo Moneta, Matthew Feickert, Mark Neubauer, Daniele Bonacorsi, Mauro Verzetti (CMS), Seth Moortgat, Sergei Gleyzer, Fernanda Psihas, Elias Coniavitis, Gilles Louppe, Valentin Kuznetsov (CMS)

Session 1 Summary by Claire and Hans, original version is here:
https://indico.cern.ch/event/595059/contributions/2498444/attachments/1430083/2197780/IML_Discussion_Group1_HPabst_CDavid_summary.pdf

The problem

- Current dataformats (ROOT) not compatible with industry ML frameworks
  - Root_numpy and root_pandas exist to convert flat ntuples into numpy arrays/Pandas dataframes
  - HEP data is difficult to fit into these formats due to the nature of N objects per event where N is not constant, however HEP-ML applications usually fit well into the scope of established ML formats (data-frame, ndarray).
- Workforce fluctuates over time, difficult to maintain all packages

For further edits please use: https://www.sharelatex.com/project/593a919f0f690d906afbfc76

The situation

- People convert in an out from ROOT to external tools, then back into ROOT for publishing and archiving policies in HEP

HEP tools vs external tools

- In HEP our data format (ROOT) is unique. Not seen anywhere else. Moreover, it's too commonly accepted to hope for a change in the near future.
- But ROOT isn't modular enough (yet) for machine learning.
- If data were in another format (protobuf, dataframe, HDF…) it would be easier.

- Advantages of the external tools:
  - huge community to offer help (StackOverflow) and support (trainings)
  - given the competition for permanent positions, physicist leave HEP for industry and have to learn new tools. Using common tools will avoid an additional learning curve.
  - moving forward as fast as industry in ML field, increase productivity
- Disadvantages of external tools:
  - danger of deciding on a tool now (tensorflow) but it may loose its trend / become obsolete / go into an unadapted direction for ML
  - worse case: no maintenance (unlikely but possible)
  - is it easy to contribute? Are HEP-patches actually taken by maintainers? And will HEP-specific features be supported and maintained over time?
- Interfacing external tools from TMVA is an option and already deployed for scikit-learn, R and keras
  - Pro: it eases the comparison of TMVA methods to external tools (no need to convert data files or reproduce benchmark methods)
  - Con: it creates the need to maintain interfaces to tools which already have maintained interfaces.
  - Con: Deploying TMVA specific interfaces to external tools renders external documentation (e.g. on stackoverflow) inapplicable for HEP users.
  - Con: Adapting to upstream changes delays the release cycle from upstream to the HEP user
  - Con: Such an interface would split responsibilities between TMVA and the external tools (where folding for cross validation happens, controlling of training steps) and thereby constrain what options of the external tool can actually be accessed from the TMVA interface

Advanced solution: A middleware:
In this context a double-wrapper:

1. from HEP tool to the external tool
2. from the external tool to the HEP tool (after ML training)

For further edits please use: https://www.sharelatex.com/project/593a919f0f690d906afbfc76

Caution: *this middleware doesn't imply that it hides completely the other layers. Here one needs knowledge on both inputs and outputs to make a bridge between the HEP tools and external tools.*

HEP specific ML demands

- Highly performant algorithms that are flexible and tunable for maximum performance
- They (partially) need to work under the tight latency constraints of HEP experiments
- Training on large datasets (too big for the RAM)

General demands

- Maximize selection efficiency
- Maximise usage efficiency
- Minimize training effort (impedance to empower the physicists to do the science)
- Hardware agnostic interfaces would be a big plus (user writes code which works on GPU farms, many core systems, the grid, the cloud)

Technical requirements to tools

- should be "agnostic":
    - easily scriptable for productivity (e.g. python? or R?)
    - can support several languages
    - format agnostic? (a jet described in a rootfile or slha or … is still a jet)
- Yes but risk of being too abstract and need then to write more so that features can be used: this defeats the native purpose.
- "We should take the simplest and adapt it to the more complex" (as low as possible for the training part, but high for the use of the result)
- Modular, lightweight
- Flexible enough to adapt to future changes in the ML tool landscape
- possibility to contact through the alumni groups the HEP physicists who left for industry and get their recommendations on best external ML tools for HEP. Also some experiments have dedicated groups overlooking on how analyses are done.
- Long term key is automation:
    - avoid intermediary steps (and files)
    - make results reproducible for both experimentalists (PhD students, theorists…)
    - make experiment repeatable in a simpler & faster way
- Long term safety and maintenance must be granted:
    - HEP specific implementations of machine learning methods (or interfaces) create maintenance demand
    - Adaptability to future machine learning tools must be possible
- Trained machine learning models must be "exportable" into reconstruction and trigger software

For further edits please use: https://www.sharelatex.com/project/593a919f0f690d906afbfc76

Note: some people used interchangeably the word middleware and plugin. The latter implies the choice of a given framework with contribution from the HEP developers, but in this current discussion no differentiation is made.

Implementation

how to avoid (likely very heavy) intermediate data format while converting rootfiles?
should we contribute or rewrite?

News from TMVA (Lorenzo)

- lots of new development in the past year (e.g. on Deep Neural Networks, with even better performance than Theano)
- People learning (instead of only using) the algorithms are the best ones: they have the knowledge to make the tool better. Not throwing things into a black box.
- TMVA is good while dealing with weighted events and systematics

Pro: Central tool maintained by the community: newcomer can easily start and improve the tool
Cons: with HEP tools one needs to be an expert to start contributing (steep learning curve) whereas external libraries are easier to read and use.
Cons: non transferable skill
Cons: no support from other communities

Challenges

- Make a choice from the plethora of data formats
- Define a metric to evaluate data format, machine learning tools, machine learning method
- The machine learning landscape is evolving rapidly
- Lifespan of external tools is hard to predict
- For short latency applications, a software benchmark is needed
  - Discrimination power, error rates, latency of application - all of these are differential in cost (USD, CHF, ..) and driven by the physics goals
  - Automation useful - weak analogy with HEPSPEC06
- Machine learning method choices are entangled with hardware acquisition choices (GPU, Phi, …)

Conclusion

- ROOT might become more modular
- The least risky way: opening HEP to more/many alternatives to ROOT in parallel
- Decide the right level of abstraction to make it adaptable for many languages & tools
- Survey what local solutions have been already implemented with their local experts
  - get an overview on the trends
  - make a database of expertise within the HEP community

    - contact HEP people who left for industry and benefit from their double-expertise

  Let's bring a couple of people doing that? Relatively easy.
- Cope with the risk of obsolete language/tool/package with good archiving methods
- Software development effort is needed in any way
    - Maintaining HEP internal tools
    - Maintaining interfaces between internal and external tools
    - Contribution to external tools (for long term support or to provide HEP specific features)

Modularisation demands and ideas for ROOT

- Modular I/O (direct read/write to numpy/hdf5/…)
- Modular plotting (e.g. use matplotlib)
- Modular machine learning
- Modular math (e.g. eigen)

2.6.1 Existing middleware and converter solutions between ROOT and various ML data-formats

- **PyROOT** (https://root.cern.ch/pyroot): it is a Python extension module that allows the user to interact with ROOT data/classes
- root_numpy (https://github.com/scikit-hep/root_numpy), The interface between ROOT and NumPy supported by scikit-hep community
- **c2numpy** (https://github.com/diana-hep/c2numpy), pure C-based code to convert ROOT data into Numpy array, can be used in C/C++ frameworks
- **root4j** (https://github.com/diana-hep/root4j), The hep.io.root package contains a simple Java interface for reading Root files. This tool has been developed based on freehep-rootio one (http://java.freehep.org/freehep-rootio)
- **spark-root** (https://github.com/diana-hep/spark-root), directly read ROOT files as Spark DataFrames using root4j
- **root2hdf5** (http://www.rootpy.org/commands/root2hdf5.html), converts ROOT files containing TTrees into HDF5 files containing HDF5 tables.

For further edits please use: https://www.sharelatex.com/project/593a919f0f690d906afbfc76

3. Machine Learning Applications and R&D

Contributors: Mark Neubauer, Matthew Feickert, Piero Altoe (NVIDIA), Kyle Cranmer, Maria Girone, Sofia Vallecorsa, Johannes Junggeburth, Nicolas Köhler, Jonas Graw, Ben Hooberman, Michael Kagan, Mike Sokoloff, Daniele Bonacorsi, Gilles Louppe, Mario Campanelli, Michela Paganini, Paul Seyfert, Steven Schramm, Paolo Calafiura, Michele Floris, Jamal Rorie, Filip Siroky, Przemysław Karpiński, Alexander Radovic

Reminder about HL-LHC physics: We will be squarely in a luminosity-dominated era and the spectrum of analyses will evolve from now.

3.1 Compelling New Applications of Machine Learning in HEP

In general, the goal is to identify potential demonstrators of machine learning applications with wide community interest and a mechanism for collaboration and reporting outcomes. The goal is to better involve the academia (computer science, statistics, data science) and private sector.

3.1.1 Simulations

- Improved fast simulation to (time) efficiently and accurately produce the massive simulated datasets needed for HL-LHC analysis
- Generative models (e.g. GANs), autoregressive models (Pixel RNN, etc), autoencoder models (split and use decoder for simulation)
- Could imagine some use of ML in helping estimate the uncertainty envelope of the simulations
- Bayesian optimization & sequential design for dramatically more efficient generation of monte carlo. As opposed to generating MC a priori, we can sometimes be much more efficient if we generate MC on the fly in areas where we need it.
- We are not entirely sure what the above two bullets mean but it is clear that fast simulation needs to improve in quality well before HL-LHC. Generative networks like CaloGAN, will help with that. Another promising avenue would be (as we think it is written above) to replace I/O intensive pile-up to simulate the underlying event with “just-in-time” simulation of min bias background events (or the whole min-bias background) using generative networks.

3.1.2 Triggering

- Examples of how trigger algorithm improvement may help analyses
    - Triggering of electroweak events with low-energetic objects
    - Lowering the thresholds for jet triggers
        → Improving jet calibration at very early stage of reconstruction

  - di-Higgs (e.g. hh→ 4b), for measuring self-coupling, triggering is a major problem on low energy jets
  - Fast b-hadron reconstruction
  - Supernovae and proton decay triggering at DUNE, pulling rare quiet signals out of potentially noisy detectors.

3.1.3 Accelerator Physics Applications

Possible application of ML in monitoring of the LHC. Study general usefulness ML in accelerator physics.

3.1.4 Offline Object Identification and Regression

Machine learning techniques offer promising avenues to improve the selection efficiencies and resolutions for physics objects. Identifying and measuring objects from energy depositions in individual cells bears a strong resemblance to 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. Possible applications include identification and measurements of electrons and photon from electromagnetic showers, jet properties including substracture and b-tagging, taus and missing energy. Promising deep learning architectures for these tasks include convolutional, recurrent and adversarial neural networks. These techniques including semantic segmentation are also promising for liquid argon reconstruction.

3.1.5 Exploring Deep Learning Applications

- Investigating and isolating the source of large gains from DL solutions to better inform future reconstruction/analysis efforts. Possible lines of investigation include auto-encoders and tSNE visualisations to find where any why we do better.

3.1.6 Fast Pattern Recognition and Tracking

use in triggering and offline

  - Training of DNNs (e.g. CNNs) on large resource platforms (e.g. HPCs) → Fast inference on FPGAs in online systems → HEP.TrkX

3.1.7 Monitoring Detectors, Hardware Anomalies, Preemptive Maintenance

- Automatization of the control shifters at HL-LHC?
  - Contact directly on-call experts.
- Use of the full list of trigger primitives in NN training (this information gets lost for events that don't pass the trigger).

- How can we influence hardware design and data structures so that raw data from hardware can be used for online anomaly detection?
    - Fork data from main path to be read by ML anomaly detection algorithms?
    - Industry already does that (eg. Siemens fault detection)
    - Data structures that can look at raw primitives from multiple subsystems

3.1.8 Control and Monitoring of Production Workflows

- 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
    - Plenty of data on computing operations archived - but rarely accessed - are a gold mine to measure how we ran ops, how to improve, and what can and needs to be predicted at some level (e.g. data transfers logs, job submissions info, data on site performances/failures, data on infrastructure and services behaviour, data on most accessed datasets in analysis, ..) → WLCG R&D working group
    - WLCG grid unique distributed computing architecture. Industry has similar but simpler (non-distributed) issues.
    - Data analytics on dataset location, what information is used within datasets.
    - This is potentially a great application for tools like OpenAI that may react to changes monitoring data and learn from the effect of their reaction.

3.1.9 Bayesian Optimization of Design

- Ingredients:
    - an objective function with a design - configuration of computing resources, trigger menu, analysis event selection...
    - A practical means of evaluating that objective function for different parameters (reproducible workflow)
    - One of many nice algorithms for efficient black-box optimization of expensive objective function. Eg. Bayesian optimization.
- Applications:
    - Optimization of Computing Resources and usage for the HL-LHC
    - Trigger menus
    - Event selection
    - Cost analysis
    - Detector design

- Another, perhaps more speculative contribution would be along the lines of application of symmetries. Convolutional neural nets rely on translational symmetry, and one can imagine expressing various physical symmetries in the design of neural networks for analysis. Extensive use of this technique could improve many different algorithms.

Connect with Reproducibility.

3.1.10 Learn and Reject the Standard Model (SM)

- Train some ML discriminant on the SM, and look for anomalies
- This will be an unimaginably complex structure, at least by current standards
- By the end of HL-LHC, may be a good way to find crazy new events which don't fit expectations
- Would mostly find detector problems and similar, but could find rare new processes if they exist
- Could start with very tight discrimination (look for 15 sigma discrepancies for example) and then loosen it
- Would need to carefully examine each event found by such a classifier
- Very difficult to do and not sure how possible it will be even with the full HL-LHC dataset, but also very interesting and likely the "limit" of what we can expect to occur within the next ~10 years
- This is something which is explicitly only possible (with current ideas) by using ML, we can't possibly do this type of search without ML. It also would likely have to be unsupervised learning, as we cannot train it on MC samples, that already adds a bias which may miss unexpected new physics.

3.2 Supporting Future R&D

- Common libraries/code for people to use in order to start on new developments
- Centralized repository of data or generators to produce data, such that people who want to conduct future R&D studies have somewhere to begin
    - E.g. http://acts.web.cern.ch/ACTS/
- Also a repository of challenges for people to play with new concepts, "showed the tracking challenge from connecting the dots to my student and after he said that now he understands what tracking is"
- Cannot do challenges for all problems, and it's a lot of work, but good to try to target an outside audience
    - More "internal" challenges within physics to determine what can already be done by us
    - Leads to a small number of "external" challenges with the ML community to try to benefit from unifying both physics and ML knowledge to do something revolutionary
    - Need to balance the amount of overhead in making a public challenge (~2 years to make, longer to process results) with the need to receive input from the ML experts
- Can alternatively consider publishing benchmark data (or even code) associated with HEP ML publications so that others can play with it and learn the physics related to the paper
    - Example, can get a DOI (citable) when uploading data to Mendeley Data or Zenodo

  - Can't do this for applications within a collaboration, but useful for collaboration-independent publications
- There are current approaches which are very popular (such as deep learning), but there is no guarantee that this will remain the best approach in the future. How can we ensure that whatever software methodology is optimal, we can support it? (Note: explicitly ignoring the hardware side as that's left to group 4)
  - May be worth working toward having all usage of ML tools (TMVA, keras, or anything else) go through some well-defined common interface. The interface should be independent of any specific implementation or approach, and should define a generic means of defining how to "process" a given solution/classifier/algorithm/etc
  - This is in a sense a definition of a "ML interface language", which may take the form of config files, hooks in other languages, or similar
  - It may be too early to define exactly what form this should take at the moment, but within the next ten years this may be feasible, and we should start to plan how to get there
  - If we do this, we have to ensure that we are not imposing any "solution" on the community members. It should come from the bottom-up, or otherwise be constructed such that it is extensible in a way that does not exclude potential future developments.

3.3 R&D Challenges

- ***Computing models***: Challenge for computing platforms and resources for DL - as we strive to use information that is rawer and further upstream in the event processing chain, we challenge our present computing models. These kinds of workflow should inform future computing models.
- **Storage considerations** From discussion, it is likely that CPU+GPU will allow us to survive in the next 10 years, but disk space is going to be a very hard barrier. Aside from the hardware side, what can be done from software to ensure we can continue in the future to support new ML R&D efforts (which will require lots of data)
  - Maybe autoencoders? Unlikely to happen for raw data, may even be against rules
  - Smart identification of what information is needed to be reconstructed? Likely works for majority of analyses, but will break the use cases for more exotic searches (long-lived decays in the muon systems, etc)
  - We were unable to think of a good solution, but this will clearly be an important topic to address in the next 10 years

For further edits please use: https://www.sharelatex.com/project/593a919f0f690d906afbfc76

4. Computing Resources

4.1 Data Storage

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)

4.2 Training ML Algorithms

(CPUs, GPUs, SPARK, HPC)

4.3 Application

(CPUs, GPUs, SPARK, HPC, hardware)

4.4 Data availability

Resources and related challenges

Contributors: Paolo Calafiura, Konstantin Kanishchev , Laurent Basara, Aurelius Rinkevicius, Steven Schramm, Nuno Filipe Castro, Luke Kreczko, Giles Strong

- We need shared resources providing access to most commonly used hardware for ML. We could start by building on existing **swan-like** infrastructure for interactive usage, and openstack for batch training/inference.
- The analysis use case requires us to have a well-configured software platform (e.g. right version on library for given GPU), more than lots of high-end hardware.
    - Reduce “entry” problems like finding GPU-enabled machine, compiling/installing software, getting training data
    - Even for analysis if you want to do hyperparameter optimization you quickly need access to batch processing.
- Main use case for specialized hardware is training.
- Using optimal resources for each production task may require to do distributed sub-event processing.
- GAN training already requires multi-GPU training.

- Setup study group to collect statistics from communities like IML and DS@HEP on current and expected problem sizes (epoch to train, size of input/output, number of free parms, training batch sizes).
    - Establish need of specialized resources (GPUs today) on the grid.
    - this information is also important to get time on HPC, but potentially also to influence the specs and strengthen the case for next gen HPCs.
- Existing resources (HPC) - how to find out about them, how to engage them
    - CNAF HPC (nodes equipped with one or more GPUs) as an example
    - *Should GPUs be available at every Tier 1-2 on the grid?*
        - Do we need to make changes to current Computing Elements?
    - Maintain list of know ML-friendly resources, their software platform, and access restrictions
    - Develop a tool similar to hammercloud to show ML sites are production-ready and vv that ML software is compatible with existing hardware
- Pricing question -- is it cheaper to use a cloud v.s having own T1/T2?
    - Probably in 5 -10 years
- *How do we handle training sets with billions of events.?*
- *Are training datasets significantly different from AODs in access patterns?*
- *Storage and data access.*
    - *Consider industrial state-of-the-art technologies for distributed data storing/processing. In particular look at Google BigQuery (Dremel) https://cloud.google.com/bigquery/*
- *New question: how do we organize access to shared resources?*
    - *Is there a GPU cluster in lxplus? - I don't think so*

- *How to use machine learning to do more physics within the existing computing and software budget and reduce the processing power, data transfer and storage space and human effort per unit physics*
    - Machine Learning/AI applied to resource monitoring (anomaly detection), and optimization (e.g. data transfers, job retries)

- *Can new filesystems (e.g. glustre) help to improve the I/O limitations related to the processing of huge datasets by exploiting better the locality of data?*
- *Private vs commercial cloud*
    - *How to feed you ML algorithm - moving data is costly*
    - *Last analysis mile only for commercial cloud?*

5. Bridges to other communities

Contributors: Igor Lakomov (ALICE), Konstantin Kanishchev (AMS-02), Konstantin Skazytkin (ALICE), Jonas Eschle (LHCb), Alexander Kurepin (ALICE), Andrey Ustyuzhanin (Yandex, LHCb), Michela Paganini (ATLAS), Gabriel Perdue (MINERvA), Ryan Reece (ATLAS),

For further edits please use: https://www.sharelatex.com/project/593a919f0f690d906afbfc76

Sean-Jiun Wang (CMS), Sergei Gleyzer (CMS), Meghan Kane (Industry - SoundCloud), Steven Schramm (ATLAS)

Section outline:

- Describe the need for a bridge and its benefits
- **Explain community differences**
    - ML rapid experimentation
    - ML and HEP Domain knowledge differs (need to be able to communicate)
- Roadmap to creating a bridge
    - Define the problem
    - Discuss a common approach within HEP for reaching out to ML
    - Reach out to ML
        - Become more involved members of the ML community
        - Support new collaborations between HEP-ML
        - Make benefits of collaborations accessible to the community

5.1 Introduction

Before building bridges, it is important to have a coherent strategy and vision of how to engage the ML community. In the future we would like to have vibrant collaboration between domain experts in both 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. This can lead to new research directions and applications of ML-domain expertise, novel algorithms, and direct collaboration on HEP challenges.

The ML community can also benefit 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 we can further our understanding of such topics.

There are existing examples of collaboration between HEP and ML that have produced fruitful results mostly through local connections [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 an enormous amount of time and effort into producing very 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.

5.2 Addressing domain knowledge challenge

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.

HEP also has domain knowledge that can present a barrier to external collaboration. Removing this barrier is a priority for future collaboration. By abstracting the problem, the HEP community can present it's challenges in a way that 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.

It is essential for the physics community to provide data and software to the ML community for future collaboration. In order to maximally benefit from this interaction, the subsequent guidelines should additionally be followed:

- Create the methodology and metrics to evaluate proposed solutions
- Prepare an integration plan for incoming ideas and solutions
- Feed-back results of successful applications

5.3 Communication and Outreach

A multi-prong strategy for engaging the ML community is presented, targeting different aspects of the community, from the academic to industry to general.

5.3.1 Conferences and Workshops

Conferences are a key aspect of the academic ML community, and organizing or contributing to key conferences is a means of gaining their 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 collaboration.

5.3.2 Research Centers, Fellowships and Institutes

Talented machine learning experts could be engaged by creating prestigious fellowships, software-related institutes and research centers. These can fit into planned initiatives like NSF S2I2 Software Innovation Institute, European Commission Horizon2020 or be possibly privately funded.

- Workshops/conferences/etc
    - Organize workshops and conferences open to possible external collaborators to discuss the applications, algorithms and tools.

- Organize thematic workshops at, e.g., NIPS → not a HEP workshop, but a "sparse data" workshop, for example, which would attract plenty of people not only from ML, but also from medicine, bio, … who also deal with sparse data and who might have already developed solutions we could borrow (or, conversely, we could lend them our solutions)

- Research centers/positions
  - Cross-communities research centers (based on CERN experience)
  - Future collaboration mechanisms: HEP Centers or Hubs for Computational Intelligence (potentially privately funded). A prestigious, way to hire top notch in-house ML faculty and researchers with an eye towards HEP applications.
  - Create a prestigious title/fellowship (named after an eminent computational scientist?) to attract experts from private companies such as Google and Facebook, who can often engage with the research communities with a small percentage of their time, either weekly or through a short sabbatical. We should make it more appealing for them to work with us at a lab or directly within an experiment, instead of seeking an adjunct position to help (for example) genomics research at their local university.
- General community outreach
  - Appear on Podcast for Machine Learning to explain how data science is used in HEP (at CERN / Fermilab)
    - This Week in Machine Learning Podcast
  - 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/
  - More cross-communities (Kaggle) challenges.
  - Reach out to Data Science community -- both academic and private sector.
    - https://cloud.google.com/genomics/

        - Bridge to industry (partnerships in applying externally developed techniques and tools to HEP problems anomaly detection against fraud transactions, spam filtering etc)
            - Formal (i.e. on company level)
            - Informal (directly with developers)
        - Investigating if it is possible to get theoretical physics communities interested.
            - E.g. 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

5.3.3 Collaborative Benchmark Datasets

- Make realistic public datasets with enough complexity and documentation and share them with the broader community.
- To the dataset release, add a “challenge” factor by providing benchmarks (without necessarily creating an official Kaggle challenge with prizes and stuff)
    - This will also help us keep track of all of our results published on that dataset
    - A good example of fulfilling the two previous bullets (documentation and benchmarks) is the MNIST dataset repository: http://yann.lecun.com/exdb/mnist/
- Create outreach-style blog posts to explain our public work in a way that is easy to understand by the public.
- Consider the creation of a central organization to curate and host public HEP datasets. One responsibility of this organization would be to hold and keep private the test (evaluation) data and then run algorithms supplied by researchers studying the dataset. This would improve the reproducibility of results and make it easier to compare different algorithms. Another responsibility of this organization would be to host a web presence with unified documentation and pointers to the data. The group should also attempt to keep track of ongoing efforts to analyze the datasets and refer interested people to those groups for collaboration opportunities, etc. (this effort would probably be based on information provided by the groups as opposed to active tracking effort).
    - We could additionally seek private/corporate sponsorship for this effort.
- Reach out to other 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.
- Common data formats

- CSV (CSV.GZ)
- HDF5
- ROOT
- AVRO

5.3.4 Training

In order to address the communication barrier and to speak the same language, the HEP community should be trained in ML concepts and terminology. This training should include accessible and well-maintained documentation explaining a framework's API and providing examples in addition to holding tutorials on specific tools and lectures on general ML concepts. More details on the foreseen training methods are provided in the next section.

6. Training the community in ML

Contributors: Fernanda Psihas, Justin Vasel, Michela Paganini, Douglas Davis, Savannah Thais, Martin Vala, Jonas Eschle, Xavier Vilasís-Cardona, Jamal Rorie, Sean-Jiun Wang, Liz Sexton-Kennedy, Sergei Gleyzer, Laurent Basara, Kurt Rinnert, Industry (SoundCloud): Meghan

Who is our audience?

- Beginners - New collaborators with no knowledge of the tools
- Intermediate users - HEP analyzers with some experience in machine learning.
- Advanced users - HEP ML experts who need to stay current in new developments

6.1 Motivation for training the community

- Lack of resources stalls progress
- Understanding the underlying algorithms makes applications more efficient, enables people to choose the right tool for a given application
- Rapidly changing field
- Efforts get duplicated in the community, this delays the learning curve to implementation time
- Transferrable skills (related to bridges to other communities)
- Training is as much as worthy investment as R&D and running
- Unifying efforts with bridging to the community + training can help find ML experts to help with HEP implementation of ML

Can we do some research into the motivations and advantages of existing training efforts? (From different experiments and from industry)

6.2 Knowledge that needs to be transferred

- Existing case studies beyond conference presentations

- Knowledge of algorithms
- Existing frameworks (most used)
- Evaluation metrics
- “Trust” metrics (data driven tests, etc)
- Various tools and their advantages in specific types of problems
- Specific software implementation training
- Good practices (preventing overfitting, etc.)
- Scripting / cleaning data

6.3 Implementation

Should target multiple formats of training and knowledge transfer: some like videos, some like twikis, some like lectures, some like nice visualizations, etc. We should not exclude people who learn in non-conventional ways from the opportunity of learning about ML.

Recognize that there are different stages of knowledge transfer.
Develop programs which encourage experts to share knowledge.

6.3.1 Community Global Training

- INITIATIVE: Massive Online Open Courses.
  Develop an open-source set of tutorials and tools
  Test and recommend existing online courses of e.g. a best of Udacity&Coursera
- INITIATIVE: Global knowledge base.
  Both experiment specific and global
  Add incentives for experts to contribute
- HEP ML Stack Overflow / Stack Exchange
- ML expert/tutor volunteer network (office hours style)
- Continue investing in periodic in-person, hands-on workshops.

6.3.2 Leveraging existing forums

- At existing conferences and schools
- Use the existing forums like experiment specific training
- Encourage global field summaries of recent applications of ML at large conferences and schools
- Include the CERN Knowledge transfer group type of activities → motivate tutors by providing them with more challenging problems, opportunities to learn something new

How do you create incentives for experts to share knowledge?

For further edits please use: https://www.sharelatex.com/project/593a919f0f690d906afbfc76

6.3.3 Link to the ML Community

- How to motivate the community to participate in developing programs? Can some of it come from the ML group & leadership?
    - see above

- How to deal with competitiveness in the field? Can't force people to be interested in forming a community. How can we incentivize this? Building a vibrant, welcoming community may help. Link to Bridges.

6.3.4 Tool Development

Maybe this belongs in "External and Internal ML tools"

- Open source implementations to be developed by the community
- Development of packages to transfer implementations across frameworks

6.3.5 HEP-ML Leaders Updates

- Hold regular (monthly?) meetings and workshops that targets leaders in the field to keep them current (but open to anyone)
- A regular newsletter that compiles recent news, tools, and tutorials
- Journal club that anyone can participate; similar initiatives have been successful in turning beginners into experts.
    - This is a great way to discuss how to move from theory (reading the paper) to practice (discussing it)
    - Some is chosen to present the paper and draft a brief write-up which is archived and used for people to review at a later time.

Training networks: two ML-focused ph-d training networks exist, can trained students train others

TO-DO: roadmap, long term goals to meet, assessment strategies, how to involve pre-existing training networks

One of the points that we thought were important to take into account is that the community to educate will be a substantial extension of the users community, which was primarily taken into account previously. If we want the results of modern machine learning (deep learning or else) to be, both used and accepted, given the fact they are quite new and unknown, by the rest of the experiment, the community at large needs to be educated.
GIven that, the practitioners (that mean, us, and the future ML users) will have to be able to justify their choices in front of community, use of pedagogy, and for this resources should also be dedicated. The community training needs some deep understanding of "proof-of-concept" that the nice fancy black-box tool actually provides usable results.

Put emphasis on statistics and uncertainty control, maybe develop a dedicated team / task force to assess it.

7. Challenges over the next 5-10 years:

(<u>A</u> Action Item, **<u>L</u>** Link to other WGs)

Designing a powerful and flexible model for machine-learning software in HEP

- Flexibility to accommodate different architectures and data formats
    - Import progress from the machine-learning community without investing a lot of time in development
    - Remove barriers of interoperability, transferability of machine learning models
        - **<u>A</u>** Build interfaces and flexible data formats
        - **<u>L</u>** Link to Event Processing Frameworks WG?
    - Efficient programming model
        - **<u>A</u>** Create more modular libraries
        - **<u>A</u>** Move away from storing everything in RAM
- Efficient use of computing resources:
    - Complex and data- and computationally-intensive algorithms, require optimal use of parallelized acceleration hardware, such as GPUs etc. (ML must be/stay fast enough for triggers)
        - **<u>A</u>** Abstract high-level machine-learning models from low-level implementations
            - **<u>A</u>** Reuse low-level interfaces and math libraries
                - matrix and tensor computations
            - **<u>L</u>** Math WG
        - **<u>A</u>** Keep up and support up-to-date improvements in acceleration hardware and neuromorphic computing
        - **<u>A</u>** Make powerful computing resources accessible for training and related software infrastructure (cloud et.c.)
        - **<u>A</u>** More interactive capability of machine-learning software
            - Jupyter, cloud-based?
- Long-term software maintenance and maintainability

   - Reproducibility of Machine learning models
   - **L** Preservation WG
- Establishing generative models for efficient use of computing resources for simulation tasks.
   - **L** Simulation WG
- Identification of standard/safe algorithms: which techniques are safe against noise and can be deployed in triggers

---

Comments:

Sergei, Paul: added the first draft

Mario Lassnig: Machine learning approaches so far have been mostly in the domain of physics, but there are many potential applications in computing as well. This should be extended, and there are already some demonstrator/incubator projects for bootstrapping such activities.

Alessandra Forti: if it picks up for physics it may change also the storage architecture?

Valentin Kuznetsov: I'd like to see discussion about integration of non-HEP ML toolkits into HEP workflows, e.g. from standard libraries in python (scikit-learn), R to DL frameworks as Theano, TensorFlow, etc. Their impact on HEP can be significant and may provide many benefits for community. We need tools to make a bridge from de-facto ROOT HEP data-format to more simple CSV, NumPy, etc. flat based formats in different physics/computing workflows. Also, the effort should be done in adaptation of distributed computing resources, such as Hadoop/HDFS/Spark platform, to ML needs.

Adrian Bevan: Educating the people is an issue. Some strategy for enabling people to get recognition for investing effort in algorithms is required if we want to address the imbalance between state of the art and the algorithms in use in the field; and some strategy for educating people in methods is required…

Adrian Bevan: Related to the discussion on ROOT re-engineering raised at CHEP '16 - how dependent do we want new core tools on the current ROOT implementations? Do we need to propose good practice design rules (enums not string comparisons etc, follow C++11 etc., interface guidlines) on tool developers to promote/reduce refactoring requirements down the line?

Adrian Bevan: What lifetime do we need/expect for our tools for a given version; we need to refactor on a regular basis to remain agile which is inconsistent with external forces on the community that wants us to focus on physics output and some consensus (if possible) would help discuss this issue with funding agencies.

Vava : Tools for machine learning which are safe for real-time applications (connects to trigger WG). This is both in terms of physics performance (being relatively insensitive to changing detector/accelerator conditions), implementation performance (training can be slow but application has to be instantaneous and with a small memory footprint), and reproducibility on different architectures (since the hardware used in the trigger may be quite specific for speed reasons).

Adam Aurisano: What tools and techniques can we use to understand and evaluate our methods? Can machine learning be used to improve reconstruction? How can we optimize network architectures and hyperparameters? Modern deep networks are typically large and relatively slow to evaluate on the grid. Can we use the features generated by one network as the input for multiple specialized networks?

Vava : In the last years HEP has largely become accustomed to ML approaches, not least because the gain from using e.g. a BDT instead of a cut-based analysis is huge in most cases. At the level of selections, there is rarely more than a few % difference between a well tuned BDT, well tuned NN, etc., much of the work comes down to feature building, and the choice of ML method comes down a lot to personal preference. In the future, however, we will have smaller or at least harder to achieve marginal gains, and we may well be dealing with problems where there is a much bigger difference between methods. Not only in terms of absolute theoretical performance but even more in terms of performance which can be achieved in the real world given e.g. the limited sizes of training samples, limited computing resources for training/executing the classifier/regressor, and so on. So I think we need to work not only on improving the availability of ML methods within our frameworks but also on getting a better understanding of what classes of HEP problems are suited to what kinds of ML methods.

Paul Seyfert: thoughts on sorting. “Internal and external tools” and “bridges to other communities” are somewhat closely related. They are both about the interaction with the non-HEP ML world. One about software, the other about knowledge and work. I thus suggest the sorting

1. Bridges to other communities (outlines that we need contact to the outline world)
2. internal/external tools
3. Training the community in ML (natural followup)

4. Application of ML and R&D (can be read as a semi-independent addon on the first three once the foundation of the first three is done)
5. Resources and related