hep x ml

Luke de Oliveira

Lawrence Berkeley National Laboratory

Vai Technologies

@lukede0 @lukedeo lukedeo@vaitech.io https://ldo.io

Today

- (Necessarily reductionist) Overview of ML in HEP
- Thoughts on future of engineering for HEP-ML as former HEP & current industry
- Collection of (potentially interesting) research ideas, drawing heavily from existing ML work

Importance

- HEP community has a long, multi-decade history of using ML

- ML plays a role in many different corners of the field and at LHC experiments

- The cross-over field is still growing, and will become **increasingly important** over the next decade

The Current Status

Current Practices

The use of ML in HEP analyses has become commonplace over the past two decades, and the most common use case has been in signal/background classification. The

...

commonplace in reconstruction and real-time applications. Examples where ML has already been deployed in a limited way include charged and neutral particle reconstruction and identification, jet reconstruction and identification, and determining a particle's production properties (flavour tagging), based on information from the rest

HSF Whitepaper (ArXiv:1712.06982)

- Most *deployed* ML applications in HEP are **optimizations of models**

- Much current *research* is on **optimizations of representations**

The future of HEP x ML is

optimizing workflows

Building Workflows

- The key 2022 R&D goals for ML set by **HSF** include simulation, detector anomaly detection, tracking, and sustainable MEM calculation, to name a few

- **These are workflows,** not individual tasks

The future of HEP x ML will involve large engineering projects to ensure this can be sustained

Engineering in HEP x ML

HEP ML growth

- From 2013 to now, different world!
- Recognition (from ML / industry community) that HEP problems are unique and interesting
- Recognition (from HEP community) that other tools (e.g., TensorFlow, scikit-learn) can be useful for training models

HEP x ML workflow is converging to industry standards

Evolution of HEP x ML Engineering

Evolution of HEP x ML Engineering

Data Layer

Evolution of HEP x ML Engineering

Data Layer

Loading Layer

Evolution of HEP x ML Engineering

Data Layer

Loading Layer

Training Layer

Evolution of HEP x ML Engineering

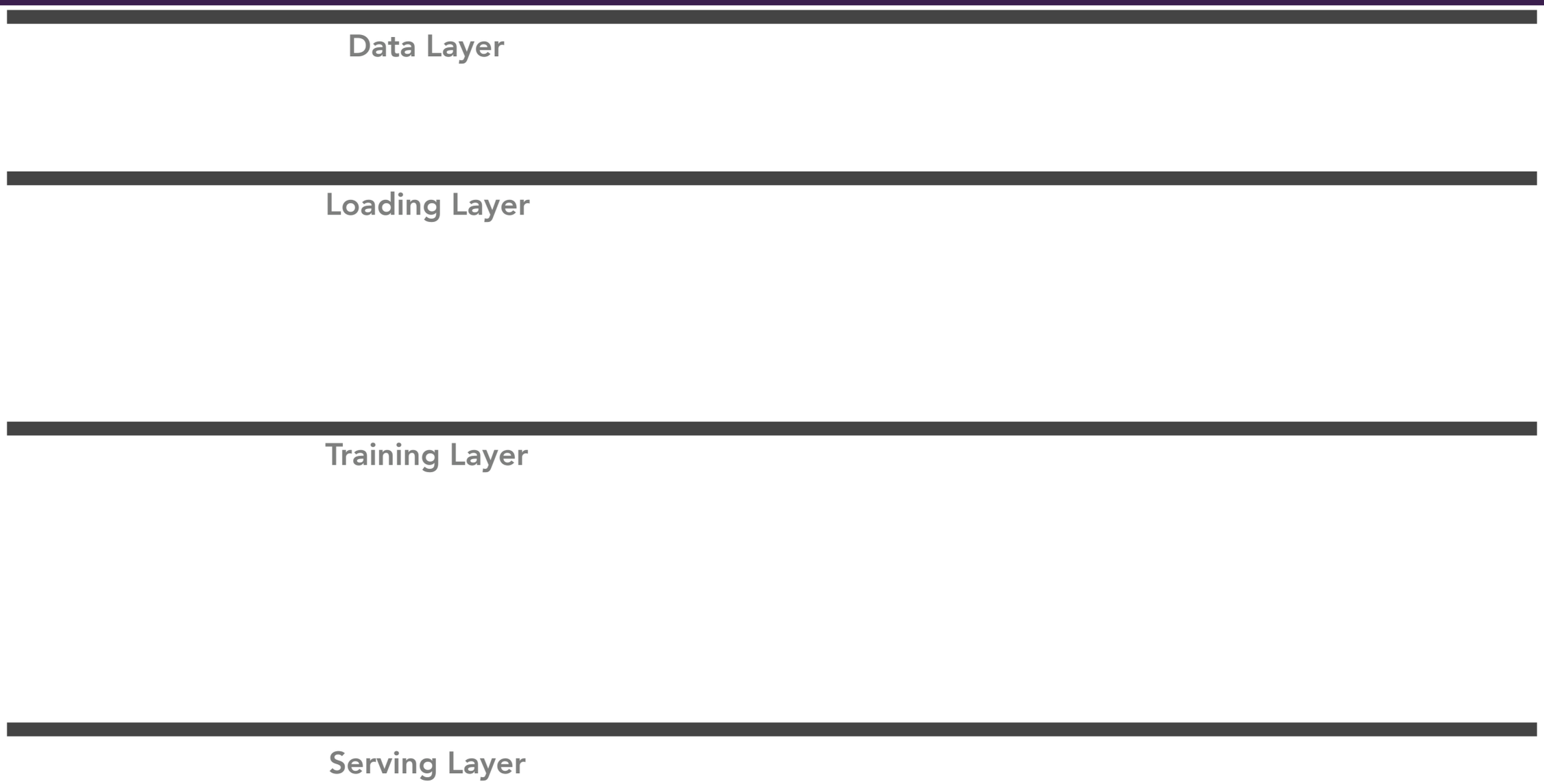
Data Layer

Loading Layer

Training Layer

Serving Layer

Evolution of HEP x ML Engineering



HEP (Circa 2013)

Evolution of HEP x ML Engineering

Data Layer

ROOT Files

Loading Layer

Training Layer

Serving Layer

HEP (Circa 2013)

Evolution of HEP x ML Engineering

Data Layer

ROOT Files

Loading Layer

Ad hoc ROOT
ETL logic

Training Layer

Serving Layer

HEP (Circa 2013)

Evolution of HEP x ML Engineering

Data Layer

ROOT Files

Loading Layer

Ad hoc ROOT
ETL logic

Training Layer

TMVA

Serving Layer

HEP (Circa 2013)

Evolution of HEP x ML Engineering

Data Layer

ROOT Files

Loading Layer

Ad hoc ROOT
ETL logic

Training Layer

TMVA

Serving Layer

Deployment Target
(TMVA)

HEP (Circa 2013)

Evolution of HEP x ML Engineering

Data Layer

ROOT Files

Loading Layer

Ad hoc ROOT
ETL logic

Training Layer

TMVA

Serving Layer

Deployment Target
(TMVA)

HEP (Circa 2013)

HEP (Circa 2018)

Evolution of HEP x ML Engineering

Data Layer

ROOT Files

ROOT Files

Loading Layer

Ad hoc ROOT ETL logic

Training Layer

TMVA

Serving Layer

Deployment Target (TMVA)

HEP (Circa 2013)

HEP (Circa 2018)

Evolution of HEP x ML Engineering

Data Layer

ROOT Files

ROOT Files

Loading Layer

Ad hoc ROOT ETL logic

Numpy / HDF5 Converters / Loaders

Training Layer

TMVA

Serving Layer

Deployment Target (TMVA)

HEP (Circa 2013)

HEP (Circa 2018)

Evolution of HEP x ML Engineering

	Data Layer	
	ROOT Files	ROOT Files
	Loading Layer	
	Ad hoc ROOT ETL logic	Numpy / HDF5 Converters / Loaders
	Training Layer	
	TMVA	Keras, TensorFlow, PyTorch, XGBoost, scikit-learn, …
	Serving Layer	
	Deployment Target (TMVA)	
	HEP (Circa 2013)	HEP (Circa 2018)

Evolution of HEP x ML Engineering

	Data Layer	
	ROOT Files	ROOT Files
	Loading Layer	
	Ad hoc ROOT ETL logic	Numpy / HDF5 Converters / Loaders
	Training Layer	
	TMVA	Keras, TensorFlow, PyTorch, XGBoost, scikit-learn, …
	Serving Layer	
	Deployment Target (TMVA)	Deployment Target (lwtnn, TensorFlow, TMVA wrappers)
	HEP (Circa 2013)	HEP (Circa 2018)

Evolution of HEP x ML Engineering

	HEP (Circa 2013)	HEP (Circa 2018)	Industry
Data Layer	ROOT Files	ROOT Files	
Loading Layer	Ad hoc ROOT ETL logic	Numpy / HDF5 Converters / Loaders	
Training Layer	TMVA	Keras, TensorFlow, PyTorch, XGBoost, scikit-learn, …	
Serving Layer	Deployment Target (TMVA)	Deployment Target (lwtnn, TensorFlow, TMVA wrappers)	

Evolution of HEP x ML Engineering

Data Layer		
ROOT Files	ROOT Files	DB / HDFS etc.
Loading Layer		
Ad hoc ROOT ETL logic	Numpy / HDF5 Converters / Loaders	Numpy / HDF5 Converters / Loaders
Training Layer		
TMVA	Keras, TensorFlow, PyTorch, XGBoost, scikit-learn, …	Keras, TensorFlow, PyTorch, XGBoost, scikit-learn, …
Serving Layer		
Deployment Target (TMVA)	Deployment Target (lwtnn, TensorFlow, TMVA wrappers)	Deployment Target (TensorFlow Serving, SageMaker, etc.)
HEP (Circa 2013)	HEP (Circa 2018)	Industry

Evolution of HEP x ML Engineering

Data Layer		
ROOT Files	ROOT Files	DB / HDFS etc.
Loader		
Ad hoc R[illegible]	Numpy / HDF5 Converters / Loaders	Numpy / HDF5 Converters / Loaders
Training Layer		
TMVA	Keras, TensorFlow, PyTorch, XGBoost, scikit-learn, …	Keras, TensorFlow, PyTorch, XGBoost, scikit-learn, …
Serving Layer		
Deployment Target (TMVA)	Deployment Target (lwtnn, TensorFlow, TMVA wrappers)	Deployment Target (TensorFlow Serving, SageMaker, etc.)
HEP (Circa 2013)	HEP (Circa 2018)	Industry

Reusability & Reproducibility

Evolution of HEP x ML Engineering

Data Layer		
ROOT Files	ROOT Files	DB / HDFS etc.
Loader		
Ad hoc R…	Numpy / HDF5 Converters / Loaders	Numpy / HDF5 Converters / Loaders
Training Layer		
	Keras, TensorFlow, PyTorch, XGBoost, scikit-learn, …	Keras, TensorFlow, PyTorch, XGBoost, scikit-learn, …
Serving Layer		
Deployment Target (TMVA)	Deployment Target (lwtnn, TensorFlow, TMVA wrappers)	Deployment Target (TensorFlow Serving, SageMaker, etc.)
HEP (Circa 2013)	HEP (Circa 2018)	Industry

Reusability & Reproducibility

Flexibility

Evolution of HEP x ML Engineering

	HEP (Circa 2013)	HEP (Circa 2018)	Industry
Data Layer	ROOT Files	ROOT Files	DB / HDFS etc.
Loading Layer	Ad hoc R…	Numpy / HDF5 Converters / Loaders	Numpy / HDF5 Converters / Loaders
Training Layer		Keras, TensorFlow, PyTorch, XGBoost, scikit-learn, …	Keras, TensorFlow, PyTorch, XGBoost, scikit-learn, …
Deployment Layer		Deployment Target (lwtnn, TensorFlow, TMVA wrappers)	Deployment Target (TensorFlow Serving, SageMaker, etc.)

Reusability & Reproducibility

Flexibility

Compatibility

Industry Ideas for HEP-ML Engineering

- Will only discuss two
- ML Experiment Management
- Universal Serving Layer

Why Care?

- Already under way!

- Important to see what growing pains other orgs where ML plays a big role had, and learn

- *ML as a Service (MLaaS):* current cloud providers rely on a MLaaS model exploiting interactive machine learning tools in order to make efficient use of resources, however, this is not yet widely used in HEP. HEP services for interactive analysis, such as CERN's Service for Web-based Analysis, SWAN [62], may play an important role in adoption of machine learning tools in HEP workflows. In order to use these tools more efficiently, sufficient and appropriately tailored hardware and instances other than SWAN will be identified.

HSF Whitepaper (ArXiv:1712.06982)

Experiment Management

- Log & record compute environment / hyperparams from ML experiment for reproduction
- Log & record metrics (AUC, rejection@eff, etc.)
- Common DB for persistence
- Searchable & sortable

Experiment Management in Industry

- Every company has their own way of doing it
- **But**, every company has a *centralized, large team* working on this
- This sounds easy, why is it hard?

Benefits of Experiment Management

- Principled way of making sure things are reproducible
- Makes for an easier time for people joining new projects
- It's hard / valuable, so very few OSS projects
    - Sacred [IDSIA]
    - StudioML [Sentient Tech.]
- Scales well

Experiment Management at CERN

- Should *at least* come from the experiment, if not the CERN level
    - Requires well thought out & centralized abstractions
- For organizations with *heterogeneous consumers* of machine learning tech., this *standardization* provides layer of guarantee
- Deep philosophical relationship to efforts from Diana
- 5-10 years from now, we will see > 5-10x more usage of ML at CERN - this can reduce growing pains

to another, and the tradeoffs of using ML algorithms compared to using more traditional software. These issues are not necessarily "factorisable", and a key goal will be to ensure that, as HEP research teams investigate the numerous approaches at hand, the expertise acquired and lessons learned, get adequately disseminated to the wider community. In general, each *team*, typically a small group of scientists from a collaboration, will serve as a source of expertise, helping others develop and deploy

HSF Whitepaper (ArXiv:1712.06982)

Experiment Management

Make people *training* ML models more productive

Universal Serving Layer

Make people *using* ML models more productive

Serving Layers for Machine Learning

- Often microservice oriented
- What is a microservice?
    - High level: break large application into smaller one
    - Usually backed by Docker / LXC
- Why?
    - Every “application” has too many dependencies

ML as an Application Layer

- Very common in industry - decouple runtime envs
  - Entire backend written in Go, but I need to use scikit-learn, now what?
- ATLAS / CMS / broader CERN community faces this with ML!

Building a ML Serving Layer

- GRPC [Google], low-latency network calls with type-safe data structures called protobufs (support for Python, C++, …)
    - How large portions of Google do ML
- Message Queues can allow the parent application process to submit data to a queue, and have the ML application layer dequeue and perform the task

Why not Wrappers?

- Custom wrappers are good for small, one-off things (many in use today!)

- Benefit of no latency, language specificity

- Critical path is not target codebase or ML library, **but the wrapper itself!** 😕

How about Converters?

- Often fall into the same trap as wrappers (i.e., fragility)
- Community driven (from the ML side) projects to develop a “deep learning DSL” can be of interest here, such as ONNX [FAIR, AWS, MSFT]

A serving layer removes integration friction, and allows you to always use your best models

Speculative Research Directions in HEP x ML

Trigger Deep Learning

- Work in the ML field on quantization for inference time speed-ups

- Existing literature training binary neural networks

- Can quantization + ONNX compilation allow for direct-to-hardware deployment?

    - Works on mobile / low power

Fast Simulation with Generative Modeling

- Existing work focuses on modeling interactions with detectors (discretized)
- Is it possible to learn to generate the graph structure quickly and perform discretization post-facto?
    - Affords geometric flexibility

Monte Carlo Tree Search for Tracking

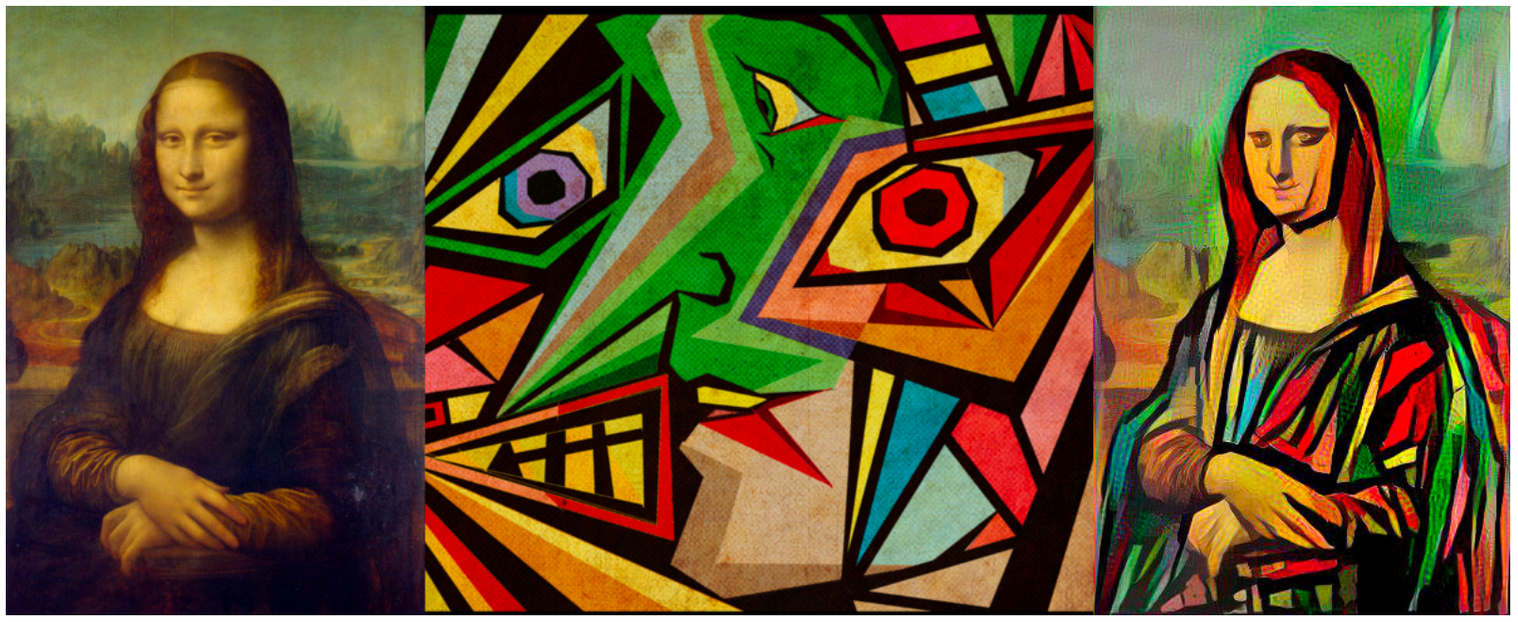
- MCTS has high profile success in Go (AlphaZero, etc.)

- Drawing inspiration from AlphaChem, can we learn to search the decision space for resolving tracks in a tractable and efficient way?

- In general, RL has been suggested as a solution to tracking, though still suffers from credit assignment and reward design

MC Style Transfer

- Style transfer has had numerous successes in Computer Vision
- Can we use similar principles to “correct” MC to look more data-like while still retaining truth info?



Concluding Remarks

- High level overview of current state of HEPML
- Workloads & engineering of the future for HEPML
- Select ideas for workflow optimizations that may be fruitful
- **ML will continue to play an increasingly important role in HEP**, and it is important to think about how to scale and get the most out of the R&D that will happen

Thanks!