DeepJet: A Machine Learning Environment for High-energy Physics

Swapneel Mehta; Mauro Verzetti; Jan Kieseler; Markus Stoye
on behalf of the CMS Collaboration at CERN

Abstract

The DeepJet Framework is used to extrapolate and find applications for cutting-edge practices in deep learning to problems involving supervised learning for high-energy physics. Originally envisaged to support jet-flavor tagging and classification, it has grown to encompass a range of use-cases as it underwent a transformation into a multi-purpose tool for physics analysis at the Compact Muon Solenoid (CMS) Experiment. This poster illustrates the workflow, features, and use-cases for this framework as we extend it to a generalised supervised learning framework for physics at the European Organisation for Nuclear Research (CERN).

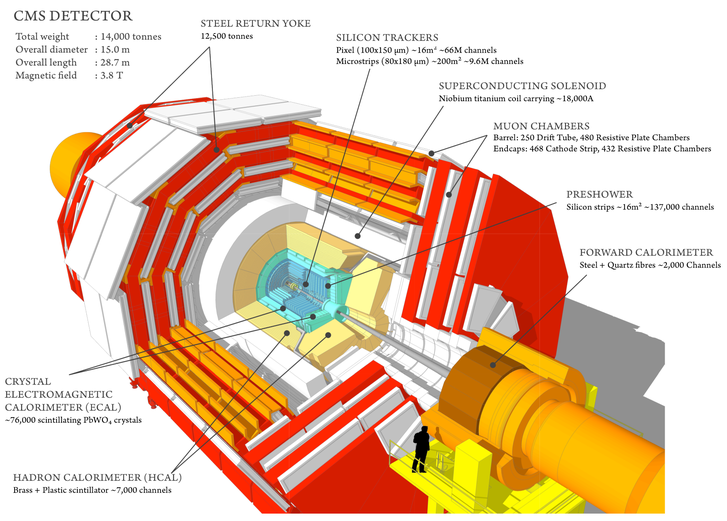


Figure 1. The Compact Muon Solenoid (CMS) Detector

Development

Jets are collimated streams of radiation that often contributes significantly to the background in a high-energy collision. Jet tagging has been studied is to and has resulted in the development of numerous algorithms as presented by the CMS Experiment [1].

The central idea in the DeepCSV tagger is the incorporation of flavour information in addition to kinematic information that gives it equal or improved performance over the other jet taggers (b and c) making it an efficient multi jet classifier.

The development of the DeepCSV (Combined Secondary Vertex) flavor tagger in part initiated the project of extending the script(s) into a full-fledged tool for Physics Analysis.

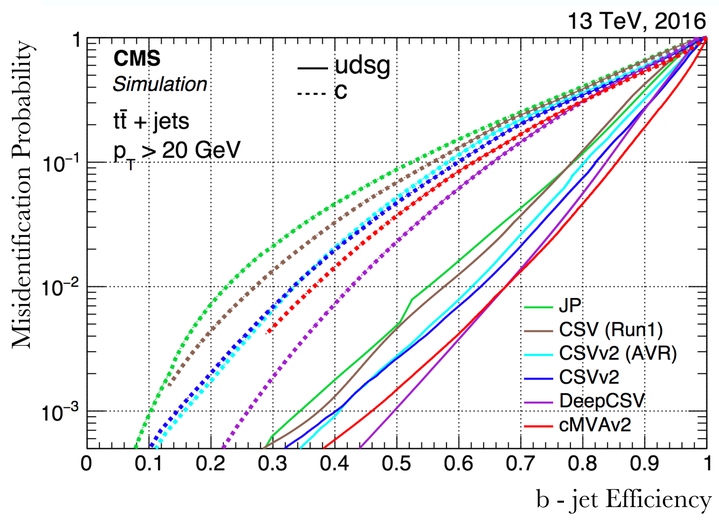


Figure 2. Comparing the Performance of Jet Taggers used by the CMS Experiment over the years

Challenges

Moving from a development environment to a production-ready setup is, quite unsurprisingly, a challenging task. It involves constraints with the compute requirement, memory, threads, and processes apart from compatibility and dependency-management. A few breakpoints include:

1. Attempting to run DeepJet on CMSSW involves interaction with certain C++ interfaces, Python code for TensorFlow with a backend in C++ creating huge memory overheads.
2. Training time is seldom outlived by the lifetime of Kerberos tokens which requires repetitive, automated renewal to avoid failed jobs.

Human elements often impact the process especially when attempting to involve third-parties for deployment:

1. Code may be underutilising parallelization and low memory features available within frameworks.
2. It is often unnecessary to reinvent the wheel with regards to existing software for handling issues.

Development	Deployment
TensorFlow + Keras	Custom Framework
Python-based	C++ based
Minimal interaction with ROOT	ROOT-centric I/O
Fewer Processes	Multiple Processes
Expendable Jobs	Job Failure Expensive
Few Memory Constraints	Memory Constraints

Table 1. Comparing 'Development' and 'Deployment'

Technical Details

At its core, DeepJet is a set of wrappers built for TensorFlow and Keras that is integrated with a set of custom libraries that manage to bypass the constraints that often pose significant computing challenges to the execution of physics analysis.

Our framework sports a range of features: simple out-of-memory training a with multi-threaded approach to maximally exploit the hardware acceleration, simple and streamlined I/O to help bookkeeping of the developments. It offers a complete set of ready-to-use templates for a simplified learning curve, also serving as guidelines for users to build their own components within DeepJet.

The DeepJet environment is compatible with the CMS Software Repository. It has undergone a prolonged series of revisions and updates in order to be able to function within the CMSSW [PR #19893].

It is installable as a Python package and available within a Docker image to simplify the deployment across multiple systems.

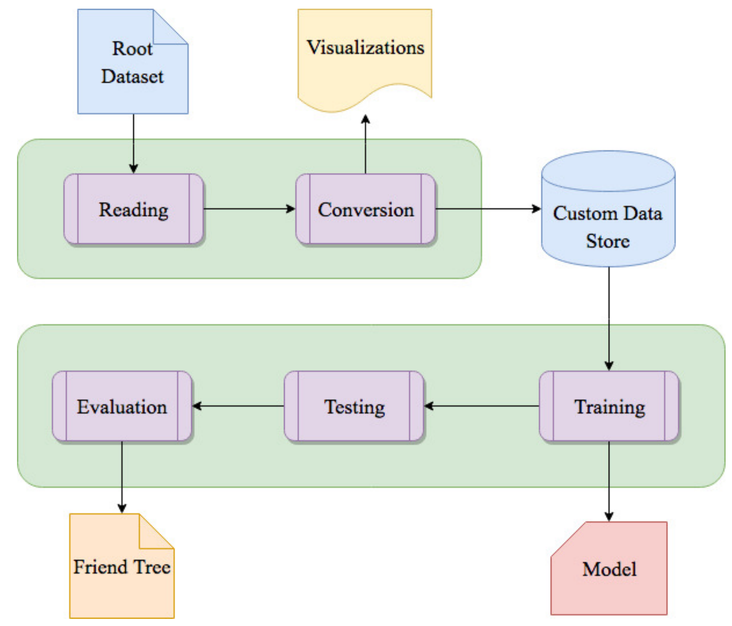


Chart 1. The Workflow for DeepJet

Tutorial

The DeepJet Framework aims to provide a set of templates as a clear set of guidelines for creating, distributing, and customising existing models. Our workflow segregates the core package from the user-defined modules and data structures as illustrated.

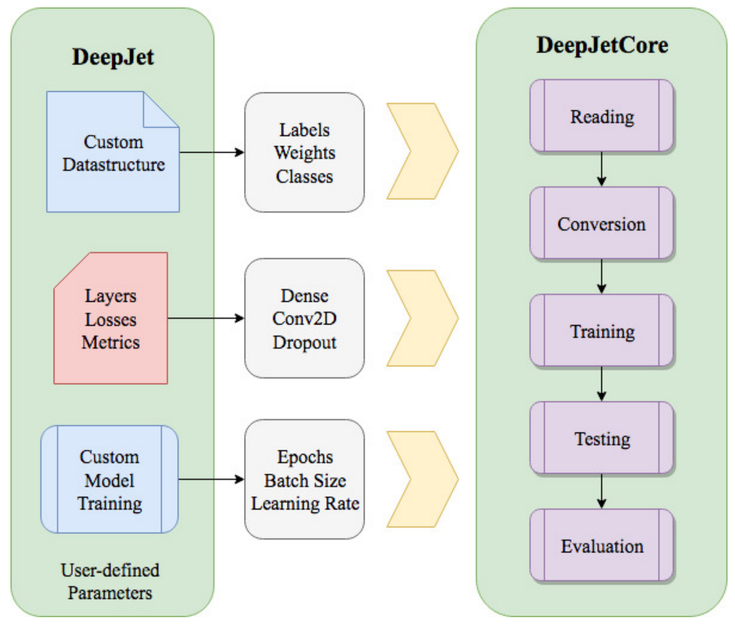
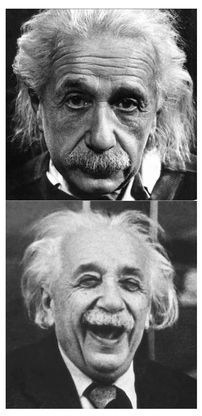


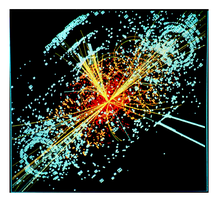
Chart 2. A Functional Overview of the Framework

Conclusions

The transition from development to deployment is a non-trivial process especially in the context of large-scale deep learning models. It is imperative to afford at least an equivalent amount of consideration to this stage of the software development life cycle as we do the planning and development phases.

DeepJet is a case-study for such a scenario whereby a set of scripts evolved into a multipurpose machine learning framework that simplifies the workflow for hundreds of analyses at the CMS Experiment.





ROOT Data Analysis Framework

Keras

DeepJet Framework

Contact

Swapneel Mehta,
European Organisation for Nuclear Research
swapneel.mehta@cern.ch

References