DeepJet: A Machine Learning Environment for High-energy Physics

Swapneel Mehta [1], **Mauro Verzetti** [2], **Jan Kieseler** [2], **Markus Stoye** [3]
[1] Dept. of Computer Engineering, D. J. Sanghvi College of Engineering
[2] European Organisation for Nuclear Research
[3] Dept. of Physics and Data Science Institute, Imperial College London
swapneel.mehta, mauro.verzetti, jan.kieseler, markus.stoye @ cern.ch

Abstract

The DeepJet Framework is used for applying cutting-edge practices in deep learning to supervised learning problems in high-energy physics. Originally envisaged to support jet-flavour 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 paper illustrates the motivation behind the development of DeepJet, its features, architecture, and workflow.

1 Introduction

Jets are collimated streams of radiation that often contribute significantly to the data captured in a high-energy collision. Jet tagging has been studied and has resulted in the development of numerous algorithms with resulting efficiency as shown above. The central idea in the DeepCSV (Combined Secondary Vertex) [2] jet tagger that we have released 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 DeepFlavour [3] tagger in part initiated the project of extending the script(s) into a full-fledged tool for physics analysis.

2 Motivation and Challenges

As is often the case with most software development projects, moving from a development environment to a production-ready setup especially for large-scale machine learning models is a non-trivial task. It involves constraints with the compute requirement, memory, threads, and processes apart from compatibility and dependency-management [4][5]. A few breakpoints include:

- Attempting to run DeepJet on the CMS Software Environment [6] involves interaction with custom C++ interfaces, Python code for TensorFlow [7] with a backend in C++ creating huge memory overheads.
- 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 performance of code in production especially when attempting to involve third-parties for deployment of code without a software development workflow in place:

- Code often underutilises parallelization and low-memory features available as part of recent updates to popular frameworks.

32nd Conference on Neural Information Processing Systems (NIPS 2018), Montréal, Canada.

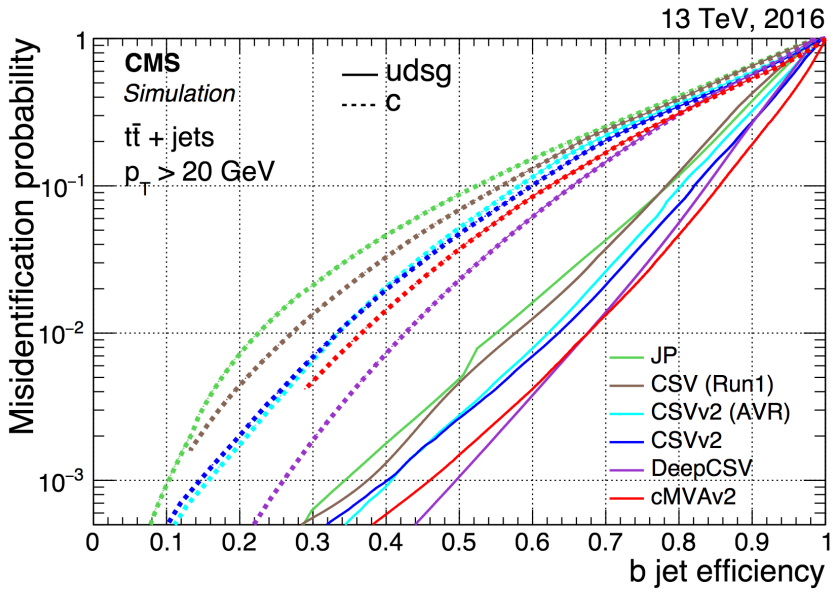

Figure 1: The performance of DeepCSV was comparable to other algorithms for jet classification.

- It is unnecessary to reinvent the wheel as opposed to using a combination of existing relevant software to provide the necessary functionality.

3 Features

At its core, DeepJet is built to include a set of wrappers for TensorFlow and Keras [8] that is integrated with a set of custom extensions to satisfy the constraints and computing caveats associated with the execution of physics analysis within the CMS Software Environment

It sports a range of features aimed at ease-of-use and reproducible machine learning research:

1. Simple out-of-memory training with a multi-threaded approach to maximally exploit the hardware acceleration.
2. Streamlined I/O to help with bookkeeping of the development.
3. Complete set of ready-to-use templates for a simplified learning curve.
4. Component-based architecture that allows users to add custom code or build their own models from scratch.

The DeepJetCore package includes custom extensions written in C++ for multi-threaded I/O and functionality associated with handling data conversion and storage of ROOT Ntuples [9] that are commonly used for storing event data in high-energy physics experiments performed within the Large Hadron Collider. It has undergone a prolonged series of revisions in order to function within the CMS Software Repository (CMSSW) [Pull Request #19893], is now available as a Python package, and within a Docker image to simplify the deployment across multiple systems.

`https://www.github.com/DL4Jets/DeepJetCore`

4 Architecture

The DeepJet architecture segregates the DeepJetCore (core scripts) package from the user-defined modules and data structures contained in the DeepJet package as illustrated in the diagram below.

The availability of DeepJet as a separate ‘example’ package makes the definition of models and training parameters lucid and flexible for customisation, should the user(s) feel the need to do so.

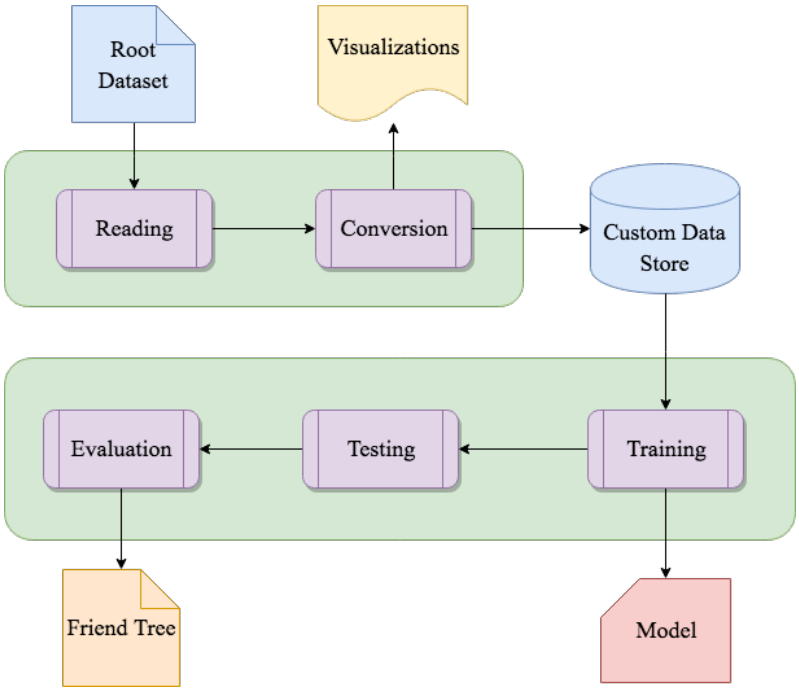

Figure 2: An overview of the framework depicting the end-to-end workflow

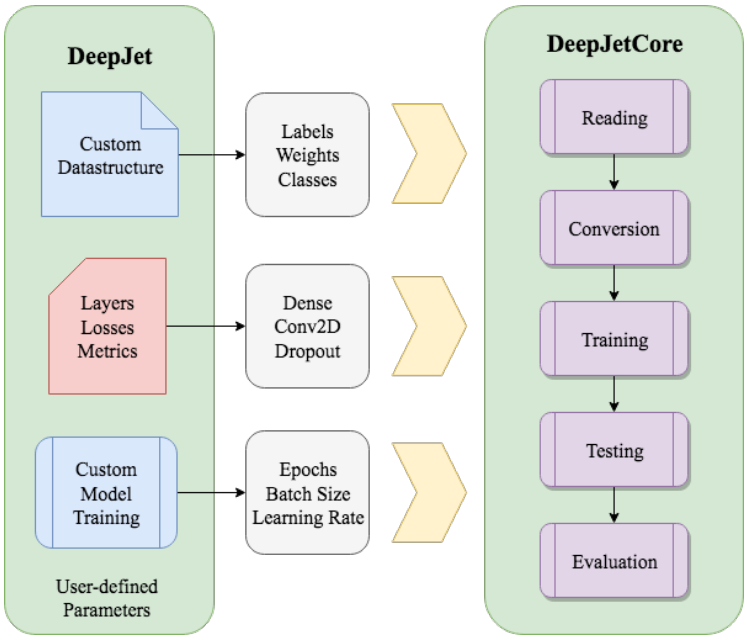

Figure 3: The definitions within the DeepJet package for custom, user-defined models

For this purpose, there are pre-defined templates and data structures available to run the training for the DeepCSV classifier bundled together in the initial install. Custom user-modules are located and loaded using the system path when executing the DeepJetCore training.

5 Example Usage

The following set of commands illustrates hos the process of training a machine learning model is simplified through the use of the DeepJet Framework. It assumes that the framework has been installed on a system as per the instructions available on the software repository for DeepJetCore.

```
# the prerequisite for running these commands is to define
a custom data structure or use preexisting ones to pick
labels/branches from the root files to be used for
training your model(s)

# the first command converts the input dataset comprising
of a set of ROOT Ntuples into a custom format for training
1. convertFromRoot -c <custom_datastructure> <data>

# the second command allows users to use custom model
parameters or use predefined ones from the included or
shared templates
2. train_template <data> <custom_template>

# the same conversion command converts the test dataset
similar to the conversion of the training ROOT dataset

# the final command run is for inference using the model
trained on the input data which creates an output file in
the same format as the input ROOT dataset
3. predict <model> <data>
```

6 Conclusion

The DeepJet framework was designed primarily as an internal tool to streamline the process conventionally employed by physicists to employ machine learning for multi-jet classification. As features were added to deal with the increasing computational complexity espoused by distributed systems and large-scale datasets, it gravitated towards a broader set of use-cases. Currently we are not only catering to the needs of existing teams of researchers within CERN but also working on a publicly accessible demonstration of using DeepJet to train classifiers for 'non-physics' data such as the popularly used Iris dataset [Fisher, 1936].

Acknowledgments

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement number 772369).

References

[1] M. Stoye, et. al., "DeepJet: Generic physics object based jet multiclass classification for LHC experiments", *NIPS Workshop on Deep Learning for Physical Sciences* (2017).

[2] CMS Collaboration, "Identification of heavy-flavour jets with the CMS detector in pp collisions at 13 TeV", *JINST 13* (2018), arXiv:1712.07158v2 [physics.ins-det].

[3] CMS Collaboration, "CMS Phase 1 heavy flavour identification performance and developments," *Detector Performance Figures: CMS-DP-17-013*, `http://cds.cern.ch/record/2263802` (2017).

[4] M. Verzetti, et. al., "DeepJet: a deep-learned multiclass jet-tagger for slim and fat jets", 2nd *CMS IML Workshop* (2018).

[5] M. Rieger, et. al., "Tensorflow in CMSSW", *CERN IML Working Group Meeting* (2018).

[6] Innocente, Vincenzo, L. Silvestris, and D. Stickland. "CMS Software Architecture: Software framework, services and persistency in high level trigger, reconstruction and analysis." *Computer Physics Communications* 140.1-2 (2001): 31-44.

[7] Abadi, Martín, et al. "Tensorflow: a system for large-scale machine learning." *OSDI*. Vol. 16. 2016.

[8] Chollet, François. "Keras." (2015).

[9] Brun, Rene, and Fons Rademakers. "ROOT—an object oriented data analysis framework." *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 389.1-2 (1997): 81-86.