Umami: A Python toolkit for jet flavour tagging

Jackson Barr[1], Joschka Birk[2], Maxence Draguet[3], Stefano Franchellucci[4], Alexander Froch[2], Philipp Gadow[5], Daniel Hay Guest[6], Manuel Guth[4], Nicole Michelle Hartman[7], Michael Kagan[8], Osama Karkout[9], Dmitrii Kobylianskii[10], Ivan Oleksiyuk[4], Nikita Ivvan Pond[1], Frederic Renner[11], Sebastien Rettie[5], Victor Hugo Ruelas Rivera[6], Tomke Schröer[4], Martino Tanasini[12], Samuel Van Stroud[1], and Janik Von Ahnen[11]

1 University College London, United Kingdom **2** Albert-Ludwigs-Universität Freiburg, Germany **3** University of Oxford, United Kingdom **4** Université de Genève, Switzerland **5** European Laboratory for Particle Physics CERN, Switzerland **6** Humboldt University Berlin, Germany **7** Technical University of Munich, Germany **8** SLAC National Accelerator Laboratory, United States of America **9** Nikhef National Institute for Subatomic Physics and University of Amsterdam, Netherlands **10** Department of Particle Physics and Astrophysics, Weizmann Institute of Science, Israel **11** Deutsches Elektronen-Synchrotron DESY, Germany **12** INFN Genova and Universita' di Genova, Italy

DOI: 10.xxxxxx/draft

Software

- Review
- Repository
- Archive

Editor: Matthew Feickert

Reviewers:

- @jpata
- @hqucms

Submitted: 22 August 2023
Published: unpublished

Summary

Flavour-tagging, the identification of jets originating from bottom and charm quarks, is a critically important technique in the data analysis of the ATLAS experiment (ATLAS Collaboration, 2008) at the Large Hadron Collider (Evans & Bryant, 2008). It is applied in precision measurements of the Standard Model, e.g. in characterisations of the Higgs boson properties, as well as in searches for yet unknown phenomena. The long lifetime, high mass, and large decay multiplicity of hadrons containing bottom and charm quarks provide distinct signatures in particle detectors which can be exploited by machine learning algorithms. Excellent knowledge of the detector and the physics processes at hand enables simulations to provide a high-quality training dataset representative of recorded ATLAS data. The `Umami` software toolkit provides a unified data pipeline, definition of the algorithms, training and performance evaluation with a high degree of automation.

Statement of need

`Umami` is a Python (Van Rossum & Drake, 2009) toolkit for training and evaluating machine learning algorithms used in high energy physics for jet flavour tagging. The creation and training of production-grade machine learning models is supported by the `TensorFlow` (Abadi et al., 2015) and keras (Chollet, 2015) packages. The training datasets feature highly imbalanced distributions among the target classes and input features of vastly different magnitude. Consequentially, the preprocessing of the training data requires resampling to provide balanced datasets and transformation of the input features by scaling and shifting.

`Umami` provides a class-based and user-friendly interface with `yaml` (*YAML Ain't Markup Language (YAML™) Version 1.2*, 2001) configuration files to steer the data preprocessing and the training of deep neural networks. It is deployed as a Python module which can be installed with `setuptools` (*Setuptools*, 2013) or used via Docker images (Merkel, 2014). `Umami` was designed to be used by researchers in the ATLAS collaboration and is open to be applied in a more general context.

Related work

The application of machine learning in high energy physics, particularly for the classification of jets, is a common and critically important technique (Cagnotta et al., 2022; D. Guest et al., 2018). In contrast to previous efforts in jet flavour tagging (ATLAS Collaboration, 2022; Bols et al., 2020), the current state-of-the-art algorithms (Qu et al., 2022a) rely on specialised toolkits, such as the `Weaver` framework (Qu & Gouskos, 2020). These toolkits enable the design of algorithms by taking care of input processing, steering the training on large datasets and providing performance metrics. `Umami` provides the required functionality to define, train and evaluate the algorithms used in ATLAS data analysis.

Development Notes

The development of the package adheres to PEP8 standards (Rossum et al., 2001). They are enforced by a continuous integration pipeline in a GitLab project, using the `flake8` linter (*Flake8*, 2010) and the `black` command-line tool for code formatting (*Black*, 2018). The code quality is tested as part of the continuous integration pipeline with the `pytest` module (Krekel et al., 2004), using unit tests and integration tests. Documentation of the software is built automatically with the `mkdocs` (*MkDocs*, 2014) and `sphinx` (Brandl, 2008) modules and deployed to the website `https://umami.docs.cern.ch`. The `Umami` toolkit has been released as open-source software under the Apache v2 license.

Software description

The `Umami` toolkit provides an integrated workflow including input data preprocessing, algorithm training, and performance evaluation. Furthermore, it interfaces to `lwtnn` (D. H. Guest et al., 2022) to export the trained models to `json` files for `C++` deployment in the ATLAS software stack (ATLAS Collaboration, 2021).

Preprocessing

The algorithms are trained on simulated physics processes which provide jets originating from bottom and charm quarks, as well as the background processes which produce jets originating from other sources, such as light-flavour quarks, gluons, or hadronically decaying tau leptons. The input features to the algorithm provide discrimination between the processes. A more detailled discussion of the input features for jet flavour tagging is provided in Ref. (ATLAS Collaboration, 2022). The preprocessing in `Umami` addresses several challenges provided both by the nature of the training datasets and the input features.

The steps involved in the preprocessing workflow are illustrated in Figure 1.

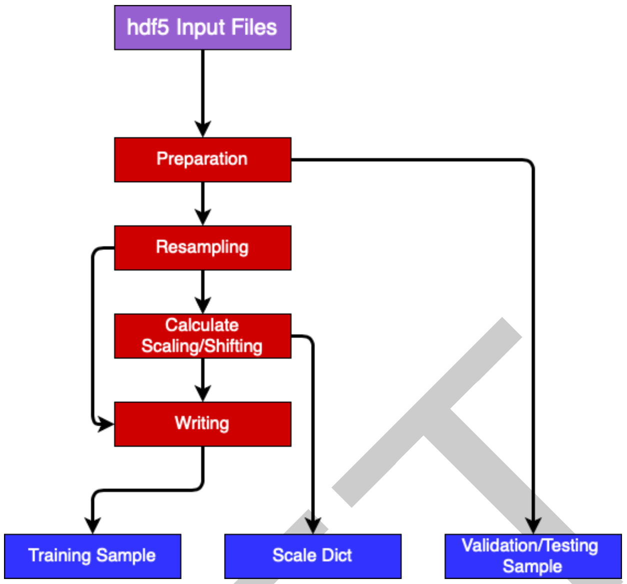

Figure 1: Illustration of the preprocessing workflow in `Umami`. Input files with simulated physics processes undergo several stages to provide a training sample, as well as a `json` file with scaling and shifting information ("Scale Dict") and validation/testing samples. These stages include a Preparation stage to define the target classes for the training, a Resampling stage to balance the classes, a Scaling/Shifting stage to transform the range of variables used for training, and a Writing stage to output the resulting samples. The validation/testing samples can also undergo the same resampling as the training sample (not shown).

Typically, the three classes "b-jets" (originating from bottom quarks), "c-jets" (originating from charm quarks), and "light-flavour jets" (originating from gluons and light-flavour quarks) are considered. Several datasets with different physics processes can be combined to a hybrid sample, which is populated over a large jet momentum range. The classes in the input dataset are highly imbalanced because the physics processes will predominantly produce light-flavoured jets instead of b-jets or c-jets. Equalising also the distributions for all classes for a certain set of features allows for the parameterisation of the algorithm performance in terms of these features, which can be desirable. Consequentially, `Umami` provides under- and oversampling methods as well as a weighting method to ensure similar kinematic distributions for the jets of different target classes.

In the first step "Preparation", datasets which are pure in the target classes are extracted from the simulated physics processes. Then, the training datasets are resampled in the "Resampling" step to provide balanced distributions between classes.

The range of values of the input features on which the algorithm is trained can differ considerably. Consequentially, `Umami` transforms the range of the variables used in training and creates a `json` file with scaling and shifting parameters. The input features are scaled and shifted in the "Scaling/Shifting" step. The resulting training data has balanced target classes and transformed input features. Finally, the training sample is written to disk, together with the `json` file and datasets for validation and performance evaluation. The resulting datasets can be stored either as an `hdf5` file (The HDF Group, 1997) or in the binary `TFRecords` format to improve reading speed provided by `TensorFlow`. The validation and testing samples can undergo the same resampling procedure as the training data if desired by the user.

Using `Umami` is not limited to jet flavour tagging but provides support for a broad range of applications. The preprocessing capabilities are demonstrated with simulated physics processes from the JetClass dataset (Qu et al., 2022b) to distinguish jets originating from Higgs boson decays from jets originating from top quark decays. This represents a similar but slightly different use of machine learning algorithms for jet classification. The software is flexible enough to address this task with only minimal modifications in configuration files. A comprehensive discussion of flavour tagging algorithm training is provided in Ref. (ATLAS Collaboration, 2022).

Figure 2 shows the pseudorapidity η and its absolute value $|\eta|$ of the jets from Higgs boson decays to b-quarks, Higgs boson decays to c-quarks, and to top quarks before and after the re-sampling step in the preprocessing. The total number of events in each class is equalised and the shape differences between classes are removed by the resampling.

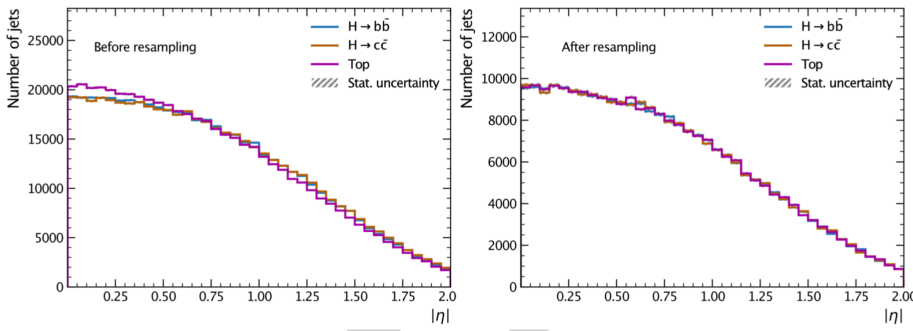

Figure 2: Distributions of the pseudorapidity η of jets from Higgs boson decays to b-quarks ($H \to b\bar{b}$-jets), Higgs boson decays to c-quarks ($H \to c\bar{c}$-jets), and to top quarks (Top) before and after resampling.

Similarly, Figure 3 shows the invariant mass before and after pre-processing, including the transformation of the dimensional quantity to a smaller range centered around zero. She scaling and shifting results in input features which are centred around zero and have the distribution in similar order of magnitude as other features.

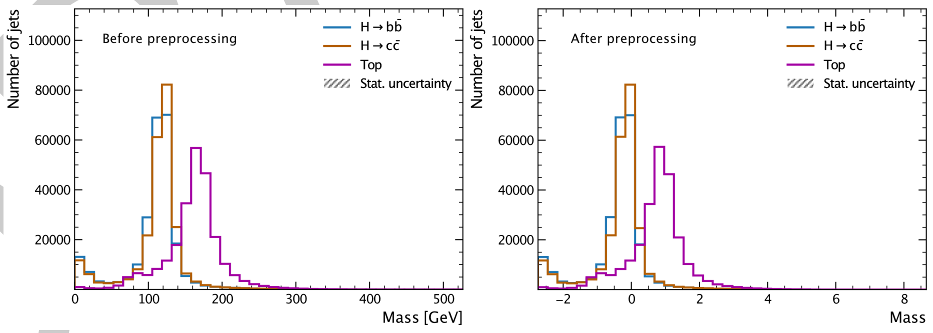

Figure 3: Distributions of the invariant mass jets from Higgs boson decays to b-quarks ($H \to b\bar{b}$-jets), Higgs boson decays to c-quarks ($H \to c\bar{c}$-jets), and to top quarks (Top) before and after pre-processing.

Training

Different architectures of neural networks, including Deep Multi-Layer-Perceptrons (LeCun et al., 2015) and Deep Sets (Zaheer et al., 2017), are supported in `Umami` for definition with configuration files. The training is performed with `keras` using the `TensorFlow` back-end and the Adam optimiser (Kingma & Ba, 2015), supporting the use of GPU resources to shorten the required time to train the networks by an order of magnitude.

The steps involved in the training workflow are illustrated in Figure 4. After the "Training" step, the optimal model configuration is chosen in the "Validation" step by evaluating the trained model with the `json` file providing the scaling and shifting parameters on the validation sample which was prepared in the preprocessing.

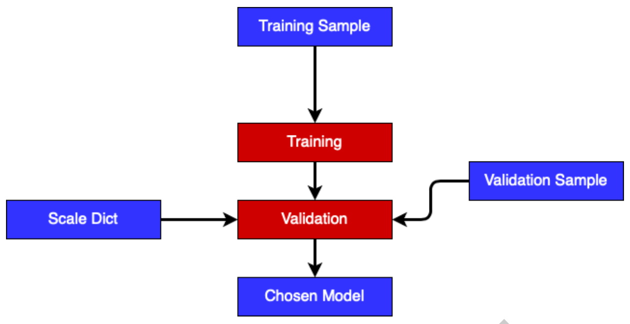

Figure 4: Illustration of the training workflow in `Umami`. The training sample is processed to determine the optimal weights of the network with a given loss function. Using the validation sample and applying to it the scaling and shifting parameters from the `json` file ("Scale Dict") obtained from the preprocessing, the performance of the training is validated to chose a certain model.

The resulting model from each epoch (in which the whole dataset was processed by the algorithm) is saved during the training. These models are evaluated on a validation dataset to identify the network weights corresponding to the epoch with the best performance. Typical performance metrics include the validation loss and the efficiency in identifying the correct jet labels. These can be plotted as a function of the training epoch to guide the selection.

As an example, a deep neural network is trained on the previously discussed JetClass dataset to separate jets originating from Higgs boson decays from jets originating from top quark decays. Figure 5 shows the loss function which is minimised while training the model on the training sample (purple curve) as a function of the epoch together with the loss function evaluated on the validation sample (green curve). Similarly, Figure 6 shows the accuracy.

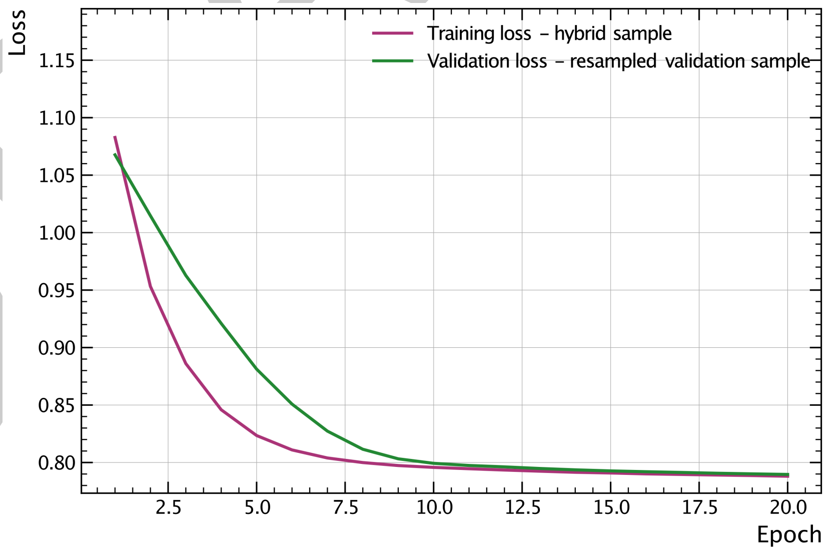

Figure 5: The loss function which is minimised while training a deep neural network for separating jets originating from Higgs boson decays from jets originating from top quark decays in the JetClass dataset, shown both for the training data (purple curve) and the validation data (green curve).

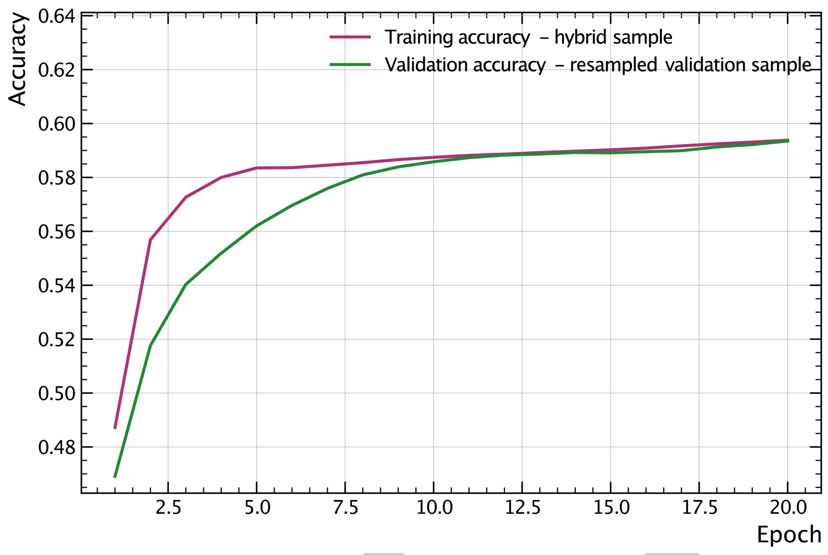

Figure 6: The accuracy for classifying jets originating from Higgs boson decays while training a deep neural network for separating jets originating from Higgs boson decays from jets originating from top quark decays in the JetClass dataset, shown both for the training data (purple curve) and the validation data (green curve).

Typically, in the early epochs the accuracy on the training data is higher than on the validation data which are not used in training. Convergence of the two curves for later epochs demonstrates that the model does generalise well and does not pick up peculiarities of the training data ("overtraining").

Performance evaluation

The performance of the chosen model can be evaluated in publication-grade plots, which are steered with configuration files. The plots are created using the `matplotlib` (Hunter, 2007) and puma (Birk et al., 2023) Python modules. Typical performance plots include

- receiver-operator-characteristics (ROC),
- efficiency of the signal class and rejection of background classes as functions of certain variables,
- confusion matrices (indicating percentage of correct or wrong classification),
- interpretability plots based on SHAPley (Lundberg & Lee, 2017) to evaluate the impact of input features to the discrimination between the classes, as well as saliency plots indicating impact of input features to final discriminant

Furthermore, all input features can be plotted with a single command, based on a `yaml` configuration file. The steps involved in the evaluation stage are illustrated in Figure 7. The inference is carried out by running the chosen model on test samples. The evaluation results are rendered in a suite of performance plots.

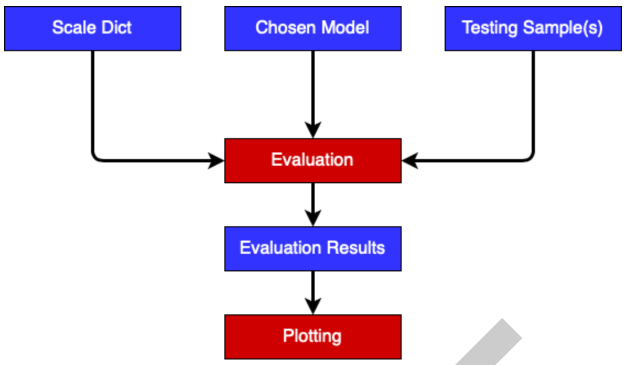

Figure 7: Illustration of the evaluation workflow in `Umami`. The chosen model after training is evaluated on the testing sample with the scaling and shifting parameters applied to it from the `json` file ("Scale Dict") obtained from the preprocessing. The results of the evaluation can be plotted.

The plotting capabilities of the `Umami` toolkit are used to evaluate the performance of flavour tagging algorithms used in ATLAS and in the corresponding publications. Figure 7 (ATLAS Collaboration, 2022) shows the light-flavour jet and c-jet rejection factors as a function of the b-jet efficiency for three different ATLAS flavour tagging algorithms MV2c10, DL1, and DL1r.

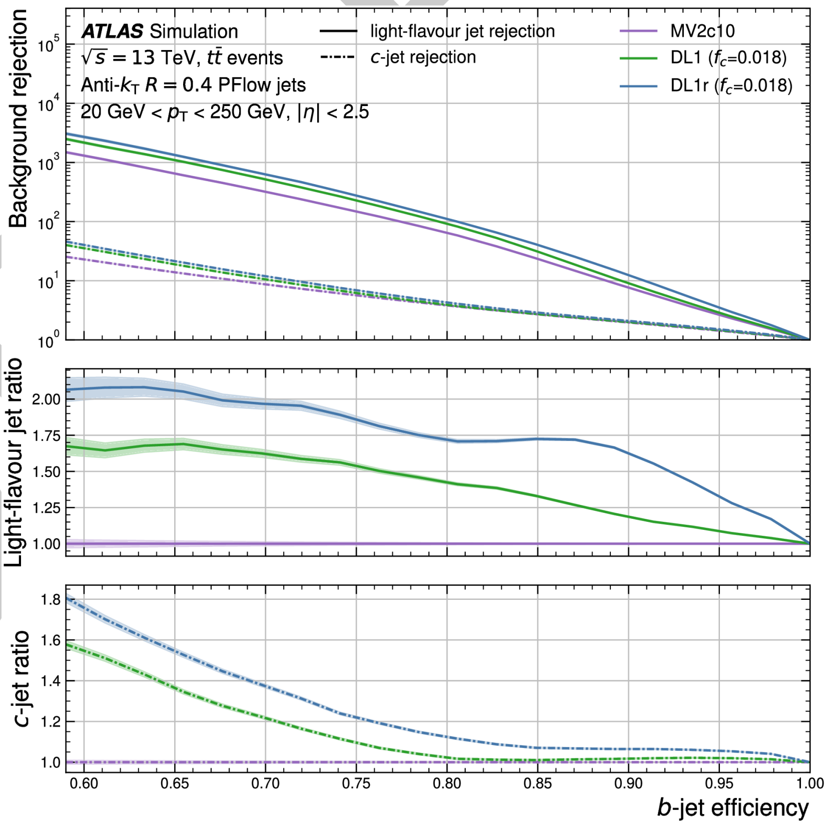

Figure 8: The light-flavour jet and c-jet rejection factors as a function of the b-jet efficiency for the high-level b-tagging algorithms MV2c10, DL1, and DL1r. The lower two panels show the ratio of the light-flavour jet rejection and the (c)-jet rejection of the algorithms to MV2c10. The statistical uncertainties of the rejection factors are calculated using binomial uncertainties and are indicated as coloured bands. Reproduced from (ATLAS Collaboration, 2022).

Conclusions and future work

We present `Umami`, a Python toolkit designed for training machine learning algorithms for jet flavour tagging. Its strong point is that it unifies the steps for preprocessing of the training samples, the training and validation of the resulting models in a mostly automated and user-friendly way. The software is widely used within the ATLAS collaboration to design neural networks which classify jets originating from bottom quarks, charm quarks or other sources. While the software is customised for this application, it is not limited to it. It is straightforward to modify the expected input features and target classes, such that the general preprocessing and training capabilities can be used in wider contexts. The identification of charged particle tracks or classification of hadronically decaying tau leptons present relevant and adequate possible use-cases.

Acknowledgements

This work was done as part of the offline software research and development programme of the ATLAS Collaboration, and we thank the collaboration for its support and cooperation.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., … Zheng, X. (2015). *TensorFlow: Large-scale machine learning on heterogeneous systems*. https://www.tensorflow.org/

ATLAS Collaboration. (2008). The ATLAS Experiment at the CERN Large Hadron Collider. *JINST*, *3*, S08003. https://doi.org/10.1088/1748-0221/3/08/S08003

ATLAS Collaboration. (2021). *The ATLAS Collaboration Software and Firmware*. ATL-SOFT-PUB-2021-001. https://cds.cern.ch/record/2767187

ATLAS Collaboration. (2022). *ATLAS flavour-tagging algorithms for the LHC Run 2 pp collision dataset*. https://arxiv.org/abs/2211.16345

Birk, J., Froch, A., VS, S., Guth, M., Gadow, P., Schröer, T., Kobylianskii, D., Rettie, S., & Strebler, T. (2023). *Umami-hep/puma: v0.2.4*. Zenodo. https://doi.org/10.5281/ZENODO.7806395

Black. (2018). https://github.com/psf/black.

Bols, E., Kieseler, J., Verzetti, M., Stoye, M., & Stakia, A. (2020). Jet flavour classification using DeepJet. *Journal of Instrumentation*, *15*(12), P12012–P12012. https://doi.org/10.1088/1748-0221/15/12/p12012

Brandl, G. (2008). *Sphinx*. https://www.sphinx-doc.org

Cagnotta, A., Carnevali, F., & Iorio, A. D. (2022). Machine learning applications for jet tagging in the CMS experiment. *Applied Sciences*, *12*(20), 10574. https://doi.org/10.3390/app122010574

Chollet, F. (2015). *Keras*. https://keras.io.

Evans, L., & Bryant, P. (2008). LHC Machine. *JINST*, *3*, S08001. https://doi.org/10.1088/1748-0221/3/08/S08001

Flake8. (2010). https://github.com/PyCQA/flake8.

Guest, D. H., Smith, J. W., Paganini, M., Kagan, M., Lanfermann, M., Krasznahorkay, A., Marley, D. E., Ghosh, A., Huth, B., & Feickert, M. (2022). *Lwtnn/lwtnn: Version 2.13*. Zenodo. https://doi.org/10.5281/ZENODO.6467676

Guest, D., Cranmer, K., & Whiteson, D. (2018). Deep learning and its application to LHC physics. *Annual Review of Nuclear and Particle Science*, *68*(1), 161–181. https://doi.org/10.1146/annurev-nucl-101917-021019

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, *9*(3), 90–95. https://doi.org/10.1109/MCSE.2007.55

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In Y. Bengio & Y. LeCun (Eds.), *3rd international conference on learning representations, ICLR 2015, san diego, CA, USA, may 7-9, 2015, conference track proceedings*. http://arxiv.org/abs/1412.6980

Krekel, H., Oliveira, B., Pfannschmidt, R., Bruynooghe, F., Laugher, B., & Bruhin, F. (2004). *Pytest 7.3*. https://github.com/pytest-dev/pytest

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. https://doi.org/10.1038/nature14539

Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in neural information processing systems 30* (pp. 4765–4774). Curran Associates, Inc. http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf

Merkel, D. (2014). Docker: Lightweight linux containers for consistent development and deployment. *Linux Journal*, *2014*(239), 2.

MkDocs. (2014). https://github.com/mkdocs/mkdocs.

Qu, H., & Gouskos, L. (2020). Jet tagging via particle clouds. *Physical Review D*, *101*(5). https://doi.org/10.1103/physrevd.101.056019

Qu, H., Li, C., & Qian, S. (2022a). Particle transformer for jet tagging. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, & S. Sabato (Eds.), *Proceedings of the 39th international conference on machine learning* (Vol. 162, pp. 18281–18292). PMLR. https://proceedings.mlr.press/v162/qu22b.html

Qu, H., Li, C., & Qian, S. (2022b). *JetClass: A large-scale dataset for deep learning in jet physics*. Zenodo. https://doi.org/10.5281/ZENODO.6619768

Rossum, G. van, Warsaw, B., & Coghlan, N. (2001). *Style guide for Python code* (PEP No. 8). https://www.python.org/dev/peps/pep-0008/

Setuptools. (2013). https://github.com/pypa/setuptools.

The HDF Group. (1997). *Hierarchical Data Format, version 5*.

Van Rossum, G., & Drake, F. L. (2009). *Python 3 reference manual*. CreateSpace. ISBN: 1441412697

YAML ain't markup language (YAML™) version 1.2. (2001). https://yaml.org/spec/1.2.2/.

Zaheer, M., Kottur, S., Ravanbhakhsh, S., Póczos, B., Salakhutdinov, R., & Smola, A. J. (2017). Deep sets. *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 3394–3404. ISBN: 9781510860964