Jet Flavor Classification with Set Transformer

Liu Taiyuan

Abstract

Code can be downloaded from `https://github.com/KoyoriSuki/JetFlavorClassification`

1. Introduction

The identification of jets produced by heavy flavor quarks is important for the physics program of the sPhenix experiment at the Relativistic Heavy Ion Collider. A main motivation for studying heavy flavor jets in heavy ion collisions is to understand the mechanism for parton-medium interactions and to further explore the issue of strong versus weak coupling[1]. The large and varying number of particles without a natural ordering in a jet leads to a difficult classification problem, on which some machine-learning techniques are applied[2]. This article introduces how a transformer[3], which is widely used in NLP, image generation, etc., such as GPT-3[4] and Stable Diffusion[5] can be used for jet flavor classification. The transformer is very suitable for analyzing information from a jet because they accept a variable-length sequence of vectors as the input and process them in a parallel way. Current results show its potential in the task of jet flavor classification.

This article is organized as follows. In Section 2, physics of jet are introduced briefly. In Section 3, the simulation and data generation are dicussed. In Section 4 and Section 5, machine-learning methods and details about construction and training of the transformer are provided. In Section 6, results and performance of the classifier based on transformer, as well as further work to be done are discussed.

2. An introduction to jets

2.1. Basic Physics

A jet is a narrow cone of hadrons and other particles produced by the hadronization of a quark or gluon. Colored partons emit soft and collinear hadrons around their directions of motion, which forms jets, so they are a manifestation of the underlying colored partons in hard scattering processes.

It is significant in many high energy physics experiment to identify the flavor of jets, i.e. whether a jet originates from a heavy-flavor quark (u,d,s) or light-flavor quark (c,b,t). The identification of c jets produced by pp collisions at a centre-of-mass energy of 13 TeV in CMS experiment can help with the study that Higgs boson decays to a pair of c quarks[6]. Jets and heavy flavor hadrons can also serve as important hard probes for studying QGP properties in STAR experiment on RHIC[7].

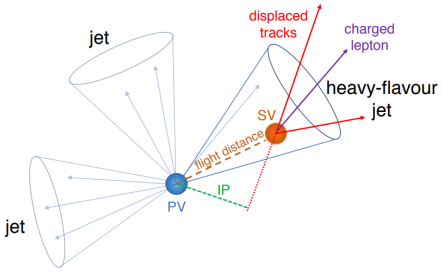


Figure 1: [6] Illustration of a heavy-flavour jet with a secondary vertex (SV) from the decay of a b or c hadron.

2.2. *Heavy-flavour jet discrimination*

There are features of heavy-flavour jets resulted from the radiation and hadronization process of b or c quarks which can be utilized for identification[6].

(1) The lifetime of b/c quarks are longer so a heavy quark tends to have a displacement after formation. This information can be demonstrated from reconstructed secondary vertices.

(2) Heavy-flavor quarks have larger mass and result in decay products with larger P_T.

(3) Charged leptons including muons and electrons are more likely to be present in the decay chain of b/c quarks.

2.3. *Jet clustering*

Jet clustering algorithms are necessary for analysing data and grouping final-state particles into jets. In this study the anti-kt jet clustering algorithm[8] is applied.

In this method, distances d_{ij} between entities (particles, pseudojets) i and j and d_{iB} between entity i and the beam (B) are introduced. The (inclusive) clustering proceeds by identifying the smallest of the distances. If it is a d_{ij}, then recombine entities i and j, while if it is d_{iB} calling i a jet and removing it from the list of entities. The distances are recalculated and the procedure repeated until no entities are left.

$$d_{ij} = min(1/k_{ti}^2, 1/k_{tj}^2)\frac{\Delta_{ij}^2}{R^2} \tag{1a}$$

$$d_{iB} = 1/k_{ti}^2 \tag{1b}$$

where $\Delta_{ij}^2 = (y_i - y_j)^2 + (\phi_i - \phi_j)^2$, and k_{ti}, y_i and ϕ_i are respectively the transverse momentum, rapidity and azimuth of particle i.

In hadron-hadron collisions Of the large number of initial state partons, only one “active parton” from each incident hadron participates in the hard scattering process, and thus Only a fraction of the hadrons in the final state are to be (loosely) associated with the hard scattering process, while the other part of final-state particles are described as ”beam jets” since they tend to have small momenta transverse to the beam axis, but possibly large momenta along the

beam axis [9]. In this study, the jet flavor classification is implemented only on the jets nearest to the active parton in each event. Details will be discussed in later sections.

3. Simulation of jet events

3.1. Event generation

Collisions, decays, showering and hadronization were simulated with PYTHIA v8.310[10]. The events are set to proton-proton collisions at centre-of-mass energies of 200 GeV, with all hard QCD physics process turned on. Jets are reconstructed from final-state particles generated by PYTHIA with Fastjet[11] toolkit implementing anti-kt jet clustering algorithm with a distance parameter of, in this study, $R = 0.7$, which means particles(tracks) are assigned to jets by requiring that they be within a cone of $\Delta R = \Delta y^2 + \Delta\phi^2 < 0.7$, as described in sec.(2.3). In addition, the transverse momentum of a jet is required to be larger than 5 GeV as a cutoff within tracker acceptance.

3.2. label of jet flavor

There are two methods to label a jet in the simulation. The first is to trace the decay chains of the constituent particles in a jet, as shown in fig.2. If a heavy-flavor quark is found, the jet will be labeled heavy flavor, otherwise the jet will be labeled light flavor[6].

```
jet 56: 0.974 -1.215 4.399
    constituent 0's pt: 0.189 chain: gamma<-pi0(index:721)<-u(index:693)<-u(index:579)<-p+(index:2)
    constituent 1's pt: 0.247 chain: pi+<-eta(index:680)<-b(index:665)<-b(index:428)<-b(index:250)<-b(index:109)<-b(index
:27)<-b(index:5)<-g(index:3)<-g(index:25)<-g(index:248)<-g(index:426)<-p+(index:1)
    constituent 2's pt: 0.129 chain: pi-<-omega(index:661)<-c(index:633)<-c(index:490)<-c(index:113)<-g(index:91)<-g(inde
x:42)<-g(index:40)<-g(index:89)<-g(index:193)<-g(index:219)<-g(index:485)<-p+(index:1)
    constituent 3's pt: 0.424 chain: pbar-<-Lambdabar0(index:1273)<-Sigmabar0(index:821)<-u(index:804)<-u(index:580)<-p+(
index:2)
```

Figure 2: List of the decay chain of constituent tracks in a jet by iteratively calling Parent() function

The other method is to match the active parton to the nearest jet[2]. The track of $index = 5$ is always the active parton that produces jets. By comparing the distance $\Delta R = \Delta y^2 + \Delta\phi^2$ between each jet and the parton, the jet with the samllest ΔR will be regarded as produced by the parton and labeled the same flavor as the parton. This method is used in this study for consistency in the group. Millions of labeled jets were generated in the simulation in total for later work.

4. Jet flavor classification methods

4.1. Introduction

The large and varying number of particles without a natural ordering in a jet leads to a difficult classification problem. Detectors directly provide us with low-level features, such as p_T, E_Cal, etc., while high-level features can

be derived from these observables, such as jet width, vertex mass, etc. Sophisticated Information of different tracks requires appropriate tools to analyze.

Shallow machine-learning techniques are firstly applied for this problem, which use expert-designed features to reduce the dimensionality before processing[12]. Such techniques perform well in the experiments, but are limited by the ability of shallow machine-learning methods to process high-dimensional datasets, causing potential loss of information when designing high-level features. Recently, deep-learning methods are applied in this problem to handle low-level high-dimensional data[2, 13], including feedforward neural network, long short-term memory (LSTM), etc., as they are able to extract features automatically from low-level information[14]. They have better performance with more information from low-level features.

It is a major challenge for machine-learning application on jet classification problem that the input sequences, i.e jets, have variable length. In addition, each element in a sequence is a vector representing the information of a track. LSTM applied on this problem addresses the problem of variable sequence length[2], but still faces the problem of nonparallel processing among track vectors, which leads to information loss if the sequence is long enough.

4.2. Set Transformer

Transformer[3] uses a novel mechanism called self-attention, which allows the model to learn the relationships and dependencies between different elements in the input sequences. It has been widely used for natural language processing, image generation, etc. Attention is a way of computing how much each element in a sequence is related to another element, and using this information to produce a new representation of the sequence.

$$Attention(Q, K, V) = (QK^T)V \tag{2}$$

where Q is the input query vectors, K, V are the key-value pairs used to map input sequence to output. The pairwise dot product QK^T measures the similarity between each pair of query and key vector. $Attention(Q, K, V)$ is a weighted sum of V where a value gets more weight if its corresponding key has larger dot product with the query, thus incorporating the information of the element correlation.

The input order of elements within a sequence is worth considering. In the application of LSTM[2], the elements with arbitrary ordering are sorted by decreasing d_0 significance, with a length limitation of 15. These requirements are mainly because LSTM is a sequential learning model whose input is nonparallel, leading to a difficulty capturing long-range dependencies[15]. In the case of this study, the track vectors input first may have less impact on the final output. Sorting by any single index (e.g. P_T) cannot ensure to maximize the preservation of information used for classification. Thus, parallel input with no prior ordering seems to be more appropriate for this classification task. For this purpose, a transformer framework named set transformer[16] is applied. Set transformer is designed to process set-structured data, which satisfy two critical requirements: firstly permutation invariant — the output of the model should not change under any permutation of the elements in the input set, and secondly the size of a set is variable.

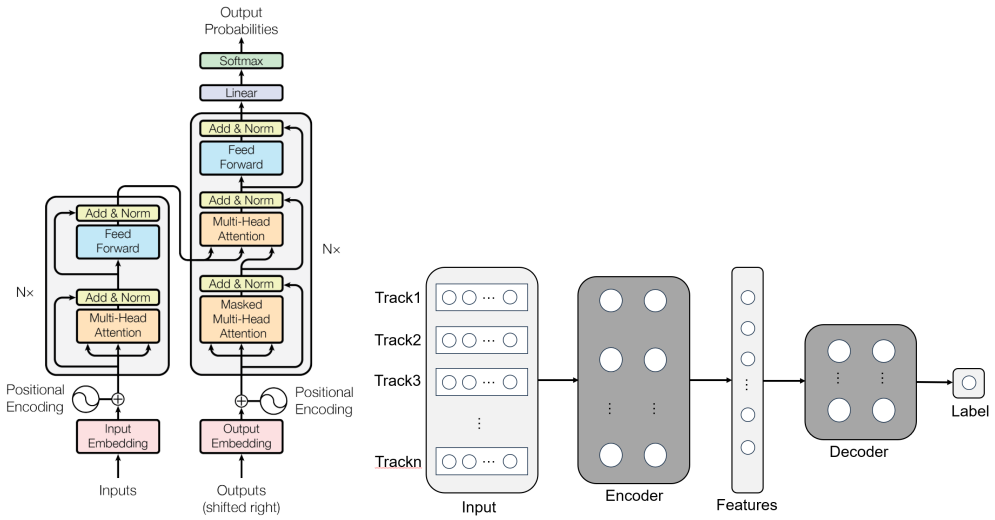


Figure 3: (Fig.3b is from ref.[3]) Left: model architecture of sequence-to-sequence transformer. Right: architecture of transformer used in jet flavor classification.

There are mainly two parts in the set transformer: an encoder which independently acts on each element of a set, and a decoder which aggregates these encoded features and produces our desired output[16], as shown in Fig.(3b). Layers involved in the model are:

MAB[3]: Multihead Attention Block. An adaptation of the encoder block of the Transformer without positional encoding and dropout.

SAB: Set Attention Block. Self-attention is performed in this layer between the elements in the set and the output contains information about interactions among input elements. It can be stack to encode higher order interactions.

ISAB: Induced Set Attention Block. Based on SAB, inducing points I are used to reduce quadratic time complexity.

PMA: Pooling by Multihead Attention. It aggregates features by applying multihead attention on a learnable set of k seed vectors.

All MABs (inside SAB, ISAB and PMA) use fully-connected layers with ReLU activations. A set transformer can be constructed using the ingredients introduced above, with the property of permutation-invariance and the ability of approximating any permutation invariant functions[16], including the jet classification problem.

5. Set transformer for jet flavor classification

5.1. Model structure

The Encoder consists of a stack of ISABs, and the decoder is a PMA followed by a fully-connected(linear) layer, with a dropout layer[17](in which nodes are randomly inactivated) inserted with $p = 0.5$ for regularization. The output of the decoder is processed with a sigmoid function to map the output to a value between 0 to 1.

$$Encoder = ISAB(ISAB(X)) \tag{3a}$$

$$Decoder = Linear(Dropout(PMA(X))) \tag{3b}$$

The whole structure of this transformer is

$$Y = Sigmoid(Decoder(Encoder(X))) \tag{4}$$

The dimension of input elements is 5 or 9 (based on the track information used). The output is a 1-dim number between 0 and 1, with a threshold of 0.5, which means any output above 0.5 will be regarded as label 1 and below 0.5 as label 0. The dimension of linear layers in the middle of the structure is set to 128.

Adaptive gradient descent is performed with Adam optimizer[18] to optimize the parameters and weights inside the transformer, with a learning rate of $5e-4$, $\beta_1 = 0.9$, $\beta_2 = 0.99$ and an L2 regularization coefficient (weight decay) of $1e-3$. The binary cross entropy (BCE) loss function is applied to measure the difference between the true labels and the probabilities predicted by the model. A manual optimization is performed over different combination of hyper-parameters to find the best model.

5.2. Data processing

As described in sec.3, the data used for training are jet-flavor pairs generated by PYTHIA8 simulation. Each jet consists of a sequence of tracks whose information is denoted by a vector, and the heavy or light flavor is denoted by 1 or 0 respectively. The dimension of the track vectors is based on how much information is used. Two kinds of combination of features are tested in this study:

Four-momentum only: The track vector is in the shape of $(p_x, p_y, p_z, E, charge)$.

Four-momentum with Vertex: The track vector is in the shape of $(p_x, p_y, p_z, E, charge, x, y, z, t)$, where the x, y, z and t come from the vertex where this track is produced.

All simulation data are split into 3 independent parts. The first part is named training set used for gradient descent to optimize model parameters. The second part is named cross-validation set used to optimize hyper-parameters, including learning rate, dim of layers, batch size, etc. The final part is named testing set used to measure the performance of the model.

5.3. Model training

The model is trained on a data set of 350109 light-flavor jets and 25606 heavy-flavor jets, and 1 million more for cross-validation. Due to the unbalance of two types of jets in the training set, weight of label 1 (heavy flavor) in the loss function is set to 7 to improve efficiency of this model. Both the model using four-momentum only and four-momentum with vertex are trained for 30 epochs.

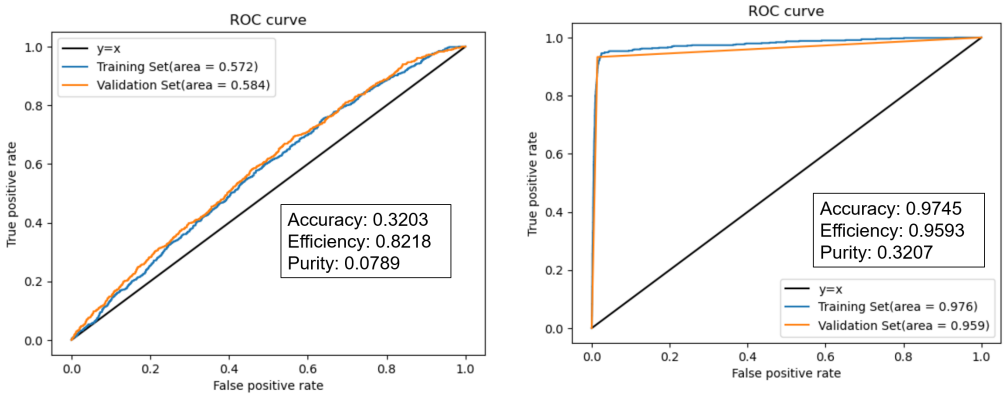
The models were constructed with PyTorch[19] v2.0.1 and computation was performed on NVIDIA GeForce RTX 3060 Laptop GPU and Intel Core i9-12900H.

6. Results from the set transformer

Several metrics can be used to measure the performance of the trained set transformer classfier. The efficiency (recall) represents the ratio of truly predicted heavy flavor to the total number of jets, and the purity (precision) refers to the ratio of truly predicted heavy-flavor to all predicted heavy flavor. These two metrics are straightforward but affected by the ratio of heavy flavor versus light flavor in the data set.

Another effective metric is ROC and AUC. ROC stands for Receiver Operating Characteristic curve who plots the True Positive Rate (TPR) against the False Positive Rate (FPR). And AUC stands for Area Under the (ROC) Curve, which is a measure of how well a binary classifier can distinguish between positive and negative classes regardless of the class balance. AUC equals 0.5 if the classifier performs random classification and equals 1.0 if performs perfect classification.

6.1. Performance

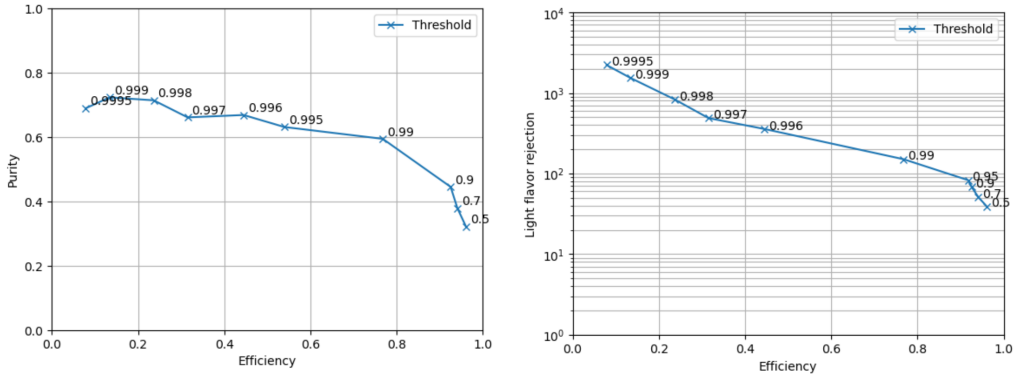


(a) (b)

Figure 4: Left: four-momentum only. Right: four-momentum with vertex information.

Fig.4 demonstrates the performance of the classifier on the validation set. The AUC of the classifier using four-momentum only is about 0.584 which is slightly better than a random classifier, illustrating the connection, though

not decisive, between jet momentum and jet flavor, as described in sec.(2.2). However, with more information from the vertex of production, the AUC increases to 0.959 with relatively high purity and efficiency.



(a) (b)

Figure 5: Performance modifying threshold of the model using four-momentum with vertex information. The value of threshold of each data point is labeled in the figure. Left: relationship between efficiency and purity. Right: relationship between efficiency and rejection, where rejection is the ratio of all light-flavor jets to light-flavor jets predicted as heavy flavor.

In addition, the threshold (0.5 by default) of the classifier can be modified to find a balance between efficiency and purity. Higher threshold can increase purity but decrease efficiency because a jet is harder to be classified as heavy flavor. As shown in fig.5, however, the increasing of purity slows down at the value of 0.7, which may suggest some problems such as incorrect label, insufficient training, etc. High rejection shows the transformer has a good ability to reject signals of light-flavor jets.

6.2. Discussion

Positively speaking, The AUC value of the transformer model is a little bit higher than that of the LSTM model[2]. But this is partly because track information used in the transformer (four-momentum with vertex) is directly derived from the simulation regardless of the bias from the detectors and algorithms in the real experiment. Simulations of sPhenix detectors[20] should be carried out to acquire data closer to the truth. On the other hand, the hyper-parameter optimization is not fully performed, which can be done later after the more realistic data sets are generated.

The task of jet flavor classification should be data-driven no matter which machine-learning framework is used. Accurate data sets are always the crucial point to get a well-performed machine learning model. Besides, the interpretation of the model, e.g. which feature is important for the classification, should be analyzed to reach consistency with physics.

In conclusion, the set transformer is very suitable for jet flavor classification as it handles well the data structure of the jets and perform parallel processing among the elements to avoid a loss of long dependencies existing in LSTM.

It shows excellent classification ability with track and vertex information, but still to be verified by using information from detector simulations. The set transformer has the potential to be applied more widely in data analysis.

7. Unfinished work

(1) Detector simulation. Taking efficiency and momentum/position resolution into account.

(2) Proton-proton collisions can be mixed with Au-Au collision background generated with AMPT simulation which can make it harder to identify the jets.

(3) Physics interpretation of the model.

(4) Better training.

References

[1] C. Aidala, et al., sPHENIX: An Upgrade Concept from the PHENIX Collaboration (7 2012). arXiv:1207.6378.

[2] D. Guest, J. Collado, P. Baldi, S.-C. Hsu, G. Urban, D. Whiteson, Jet Flavor Classification in High-Energy Physics with Deep Neural Networks, Phys. Rev. D 94 (11) (2016) 112002. arXiv:1607.08633, doi:10.1103/PhysRevD.94.112002.

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin, Attention is all you need, CoRR abs/1706.03762 (2017). arXiv:1706.03762.
URL http://arxiv.org/abs/1706.03762

[4] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, D. Amodei, Language models are few-shot learners (2020). arXiv:2005.14165.

[5] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, B. Ommer, High-resolution image synthesis with latent diffusion models (2022). arXiv:2112.10752.

[6] A. M. Sirunyan, et al., Identification of heavy-flavour jets with the CMS detector in pp collisions at 13 TeV, JINST 13 (05) (2018) P05011. arXiv:1712.07158, doi:10.1088/1748-0221/13/05/P05011.

[7] N. R. Sahoo, STAR Experimental Highlights at Hard Probes 2023, in: 11th International Conference on Hard and Electromagnetic Probes of High-Energy Nuclear Collisions: Hard Probes 2023, 2023. arXiv:2308.04801.

[8] M. Cacciari, G. P. Salam, G. Soyez, The anti-k_t jet clustering algorithm, JHEP 04 (2008) 063. arXiv:0802.1189, doi:10.1088/1126-6708/2008/04/063.

[9] S. D. Ellis, D. E. Soper, Successive combination jet algorithm for hadron collisions, Phys. Rev. D 48 (1993) 3160–3166. arXiv:hep-ph/9305266, doi:10.1103/PhysRevD.48.3160.

[10] C. Bierlich, et al., A comprehensive guide to the physics and usage of PYTHIA 8.3 (3 2022). arXiv:2203.11601, doi:10.21468/SciPostPhysCodeb.8.

[11] M. Cacciari, G. P. Salam, G. Soyez, FastJet User Manual, Eur. Phys. J. C 72 (2012) 1896. arXiv:1111.6097, doi:10.1140/epjc/s10052-012-1896-2.

[12] G. Aad, et al., Performance of b-Jet Identification in the ATLAS Experiment, JINST 11 (04) (2016) P04008. arXiv:1512.01094, doi:10.1088/1748-0221/11/04/P04008.

[13] J. Collado, J. N. Howard, T. Faucett, T. Tong, P. Baldi, D. Whiteson, Learning to identify electrons, Physical Review D 103 (11) (jun 2021). doi:10.1103/physrevd.103.116028.
URL https://doi.org/10.1103%2Fphysrevd.103.116028

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

[15] A. Karpathy, J. Johnson, L. Fei-Fei, Visualizing and understanding recurrent networks, arXiv preprint arXiv:1506.02078 (2015).

[16] J. Lee, Y. Lee, J. Kim, A. R. Kosiorek, S. Choi, Y. W. Teh, Set transformer: A framework for attention-based permutation-invariant neural networks (2019). arXiv:1810.00825.

[17] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting, Journal of Machine Learning Research 15 (56) (2014) 1929–1958.
URL http://jmlr.org/papers/v15/srivastava14a.html

[18] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization (2017). arXiv:1412.6980.

[19] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, Pytorch: An imperative style, high-performance deep learning library, in: Advances in Neural Information Processing Systems 32, Curran Associates, Inc., 2019, pp. 8024–8035.
URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[20] A. Adare, et al., An Upgrade Proposal from the PHENIX Collaboration (1 2015). arXiv:1501.06197.