Reinforced sorting networks for particle physics analyses

Martin Erdmann, Benjamin Fischer and Dennis Noll
RWTH Aachen University
E-mail: dennis.noll@rwth-aachen.de

Abstract. Deep learning architectures in particle physics are often strongly dependent on the order of their input variables. We present a two-stage deep learning architecture consisting of a network for sorting input objects and a subsequent network for data analysis. The sorting network (agent) is trained through reinforcement learning using feedback from the analysis network (environment). The optimal order depends on the environment and is learned by the agent in an unsupervised approach. Thus, the two-stage system can choose an optimal solution which is not known to the physicist in advance. We present the new approach and its application to the signal and background separation in top-quark pair associated Higgs boson production.

1. Introduction

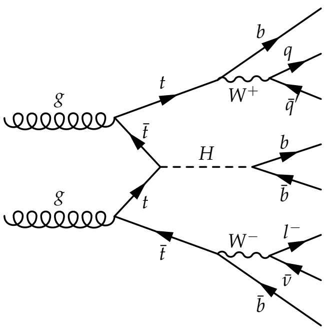
In the past year, autonomization through machine learning methods continued to advance in fundamental particle physics research [1–5]. Through their many network levels, deep neural networks autonomously transfer their input into sophisticated variables, which are optimized for answering a specific scientific question. In practice, however, common network architectures are limited by the sequence of linear operations with nonlinear activation functions and optimization by backpropagation. Challenges arise, for example, in questions where a suitable sorting of the input variables is required [6].

An example of a non-trivial sorting challenge appears at modern collider experiments, where microscopic processes appear in the sub-femtometer range and are not directly accessible. In decays of massive particles with a short lifetime, such as the top quark or the Higgs boson, only the hadrons, leptons and photons of the final state of the decays are visible in the detector. Thus the short-lived massive particles need to be reconstructed from these final-state particles. In the formation of suitable particle combinations, the ordering of the input particles is of decisive importance for the quality of the reconstruction as shown, e.g., in reference [7].

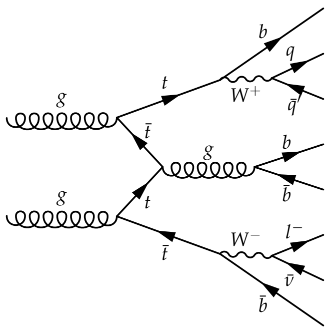
In this work we introduce a new method to optimize the sorting of input information via a deep neural network. This network combines the whole event information to pre-sort the input variables before the actual analysis. Because the best sorting is most generally not known, the training procedure is based on the principles of reinforcement learning. Another application of reinforcement learning in particle physics has been presented at the same conference [8].

As a case study we use top-quark pair associated production of Higgs bosons ($t\bar{t}H$) shown in fig. 1a. The final state of the Higgs and top-quark decays consist of six jets, a charged lepton and a neutrino, which are used as input to the analysis.

We do not provide a true sequence of the input to the sorting network, but instead use a second network to separate the signal process ($t\bar{t}H$) from the background process ($t\bar{t} + b\bar{b}$) where an example Feynman diagram is shown in fig. 1b. By assigning this classification task, both networks, the sorting network and the classification network are to be optimized. For the classification network we use the Lorentz-Boost-Network [7] as a pre-stage, which generates suitable variables, followed by a standard architecture of a fully-connected network for the classification of signal and background events.



(a)



(b)

Figure 1: Example Feynman diagrams of a) $t\bar{t}H$ and b) $t\bar{t} + b\bar{b}$ processes.

Ideally, the order of the final state particles is optimized by the sorting network in such a way that the classification network receives the jets from the decay of the Higgs boson or the two top quarks in a sequence suitable for the reconstruction of the decays. The two-stage architecture autonomously chooses an ordering accoring to the optimal outcome.

We will benchmark the results of classifying signal and background processes by comparing the accuracy depending on the initial sorting, i.e. the order of the transverse momenta, with the accuracy obtained by including the predictions of the sorting network. To get an upper benchmark, we show in addition the classification accuracy when training the sorting network with explicit generator information.

2. Simulated data sets

This analysis uses the same dataset as the previously published Lorentz Boost Network (LBN) publication [7]. Simulated $t\bar{t}H$ and $t\bar{t} + b\bar{b}$ events are used for training and evaluation of the classification task.

The events have been generated with the Pythia 8.2.26 program package [9] using beam conditions corresponding to LHC proton-proton collisions at $\sqrt{s} = 13\,\text{TeV}$. The production channel is chosen to be the dominant gluon-gluon process with the Higgs bosons preferentially decaying into bottom quark pairs. The detector response was simulated with the Delphes package [10] according to the CMS detector response and was covering all important effects such as pile-up, calometry and muon detection.

Simulated events contain the six jets from the hard process, one lepton and one neutrino. We simulated a total of 10^6 $t\bar{t}H$ and $t\bar{t} + b\bar{b}$ events and the ratio between training and validation data is chosen to be 80 : 20.

3. Forward pass: Sorting and Analysis

In this section, we will discuss the two-stage architecture in greater detail following fig. 2. In particular, we will focus on the sorting network and the analysis. Our explanations include the internal representation of the different permutations.

Algorithm 1 Reinforced Training

```
   Input: Events (e), Permutations ($\mathbb{P}$),
    Analysis ($A_0$), Sorting model ($S_0$)
1: for $i \leftarrow 0$ to epochs do
2:     $\hat{P} \leftarrow \underset{P_n \in \mathbb{P}}{\operatorname{argmax}} A_i(P_n(S_i(\mathbf{e})))$
3:     $T \leftarrow \hat{P}(S_i)$
4:     train $S_{i+1}$ to approximate $T$
5:     train $A_{i+1}$ with $S_{i+1}(\mathbf{e})$
```

3.1. Sorting

The inputs of the sorting mechanism are the p_T-sorted events ($\mathbf{e}$), consisting of the 6 jets in descending p_T-order, the lepton, the neutrino and global event information as described in [7].

First, a deep neural network produces an internal representation for the reordering of the jets. This internal representation is a feature vector which contains a position index for each jet. These indices can subsequently be used to reorder the event. The task of the sorting network is to specify suitable indices for the individual jets. The indices range between 0 and 1 and such the discrete problem of order is represented by a regression approach. After rearranging the 6 jets along the internal representation, they are assembled with the lepton and the neutrino to the sorted event ($S_i(\mathbf{e})$ with i being the current training epoch) and passed on to the analysis network.

The architecture of the sorting network is given by a feed-forward network with a depth of 6 layers, each layer consisting of 256 nodes with a selu activation [11]. The inputs are normalized and a dropout of 5% (drop probability), and an L_2-regularization with a scale of 10^{-7} are used in each layer.

3.2. Analysis

The LBN along with a deep neural network is used as analysis in this work following [7]. It is trained to combine specific input four-vectors to create particles and corresponding rest frames through linear combinations. Each particle is boosted into its dedicated rest frame via a Lorentz transformation and a generic set of features is extracted from the boosted particles. Subsequently a feed-forward deep neural network, which represents the classifier, takes the extracted high-level variables as inputs. The special sensitivity of the LBN to the order of inputs is due to the fixed linear combinations of the input particles before the Lorentz transformations. In [7] it has been shown that the LBN can achieve very good results at the $t\bar{t}H$ versus $t\bar{t} + b\bar{b}$ classification if the input particles are ordered with generator information. With a p_T sorting, the order of the jets of the different decays changes and in consequence, the performance of the LBN decreases.

The chosen hyperparameters are selected according to [7]. They have proven to be the optimal setting for this classification task, if the jets are sorted according to generator information.

Almost any analysis can be used in conjunction with the reinforced sorting mechanism. The only requirement is that for every single event, the analysis needs to have a well-defined objective. When training with an analysis based on a neural network, its loss can directly be used as feedback for the sorting mechanism.

4. Backward pass: Training

This section describes the training process of the two-stage architecture. First a pre-training is carried out, followed by an alternating training of the sorting and the analysis network. Particular attention will be paid to the permutation policy.

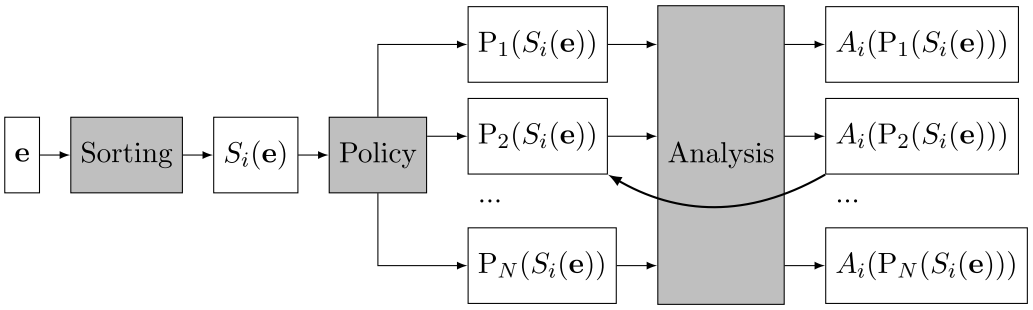

e
Sorting
$S_i(\mathbf{e})$
Policy
$\mathrm{P}_1(S_i(\mathbf{e}))$
$\mathrm{P}_2(S_i(\mathbf{e}))$
...
$\mathrm{P}_N(S_i(\mathbf{e}))$
Analysis
$A_i(\mathrm{P}_1(S_i(\mathbf{e})))$
$A_i(\mathrm{P}_2(S_i(\mathbf{e})))$
...
$A_i(\mathrm{P}_N(S_i(\mathbf{e})))$

Figure 2: Network architecture as used for training the sorting and the classification network. Tensors are marked in white, operators are marked in grey. The input ($\mathbf{e}$) consists of events with 6 p_T-sorted jets, the lepton and the neutrino. During evaluation, only the sorting agent and the analysis part are evaluated and $S_i(\mathbf{e})$ is directly fed into the analysis.

4.1. Pre-training
Before the actual reinforcement training, a pre-training of the sorting and the analysis network is performed. The sorting network is trained in a supervised approach to output the p_T-order, i.e. to reproduce the sorting according to the transverse momenta. The analysis network is trained with this p_T-order on the classification task. Both, sorting and analysis network are pre-trained for 10 epochs each. The pre-training avoids that a simple preliminary convergence of the networks must take place in the costly reinforcement learning process. In the next step, this pre-sorting state will be optimized autonomously by the reinforcement technique.

4.2. Training schedule
The two-staged architecture is trained in an alternating approach following **Algorithm 1**.

For the training of the sorting network the classification network is evaluated several times with a fixed number of permutations $\mathbb{P}$ of the current ordering ($S_i(\mathbf{e})$). The different permutations $P_n \in \mathbb{P}$ will be explained in section 4.3. The sorting network is trained to predict a sequence of jets that best separates signal and background processes with the classification network. The classification network is then trained with the optimized sequence ($S_{i+1}(\mathbf{e})$) for an improved separation of the signal and background processes.

Since in general the optimal sorting of the inputs for the analysis network is not known, the training targets of the sorting network are to be built depending on the analysis. To achieve this, the pre-sorted events ($S_i(\mathbf{e})$) are permuted according all permutations in $\mathbb{P}$. The permuted events are then evaluated by the analysis and the permuted events with the highest quality measure determine the targets of the sorting network. In this implementation, the quality of the permutations is directly measured by the analysis accuracy of the LBN A_i with i being the current epoch. After the evaluation of the different permutations, the sorting network is trained to output $P_n(S_i)$, where $A_i(P_n(S_i(\mathbf{e}))) > A_i(P_m(S_i(\mathbf{e}))) \ \forall \ m \neq n$. To achieve a higher training stability, the sorting network is trained not only on the best but the best three permutations weighted with a factor of $\exp(-a)$ with $a = 0$ for the permutation with the smallest objective, $a = 1$ for the permutation with the second smallest objective etc.

When the analysis network is trained further, the preferred permutations in the training of the sorting network change. When the sorting network is trained further, the inputs of the analysis network change. The performance of the two networks therefore directly depends on each other. The flexibility of this end-to-end approach allows the overall system to learn permutations that lie beyond naive physical intuition.

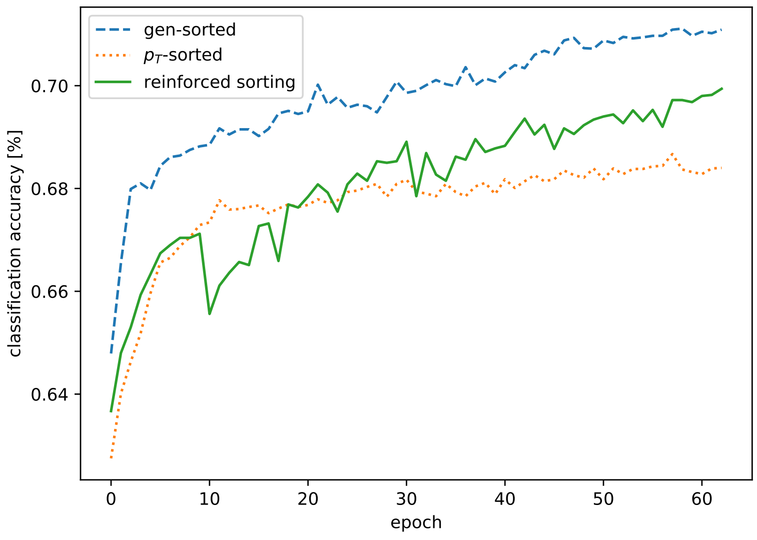


Figure 3: The training process of the reinforced sorting in terms of the classification accuracy of the analysis network. Two trainings form the lower (p_T-sorted) and upper (sorting trained with generator information) benchmark for the training process. The first 10 epochs on the reinforced sorting are pre-training.

4.3. Policy

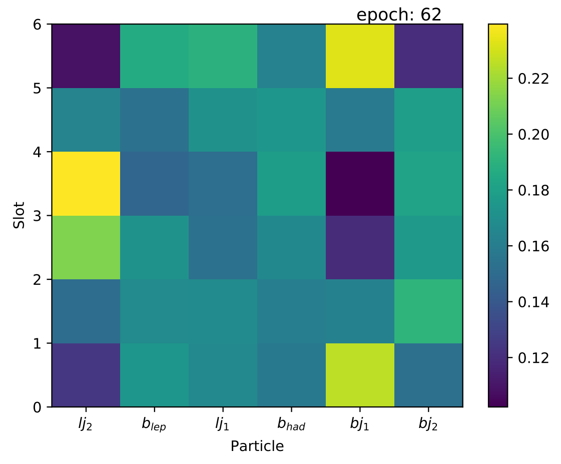
The policy decides which new permutations are to be considered in each training step. Thus it forms the core of the reinforced sorting mechanism. The policy ensures that in each training step as much space as possible of previously unseen permutations is considered. At the same time, however, it ensures that the number of evaluations remains so small that they remain quickly computable. In this implementation we consider all two-particle swaps of the sorted events $S_i(\mathbf{e})$, i.e. all exchanges ξ_{kl} of element k with element l for all $k > l$ and $k \leq n$. Each sorted event results in a total of $K(K-1)/2$ new permutations, where K is given by the number of permutable particles. The number of new permutations per training step thus increases according to $\mathcal{O}(K^2)$ and remains computable for large K as well. The identity (ξ_{kk}) is always also taken into account to allow the mechanism to remain stable at good permutations.

5. Results

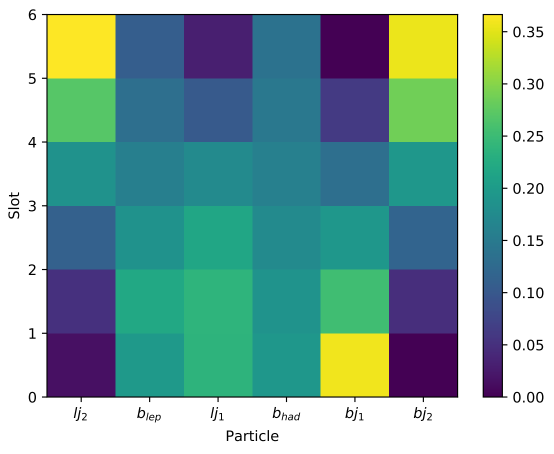
For the evaluation, only the sorting S_i and the analysis A_i are used (fig. 2). The training process is shown in fig. 3 in terms of the classification accuracy of the analysis network. Two trainings form the upper and lower benchmark for the training process. As lower bound, the training with p_T-sorted events is shown. For the upper benchmark, we first train the sorting network until convergence to output a fixed order based on generator information and subsequently train the LBN.

It can be seen that the performance of the classifier increases in conjunction with the sorting network. The classification accuracy of the two-stage architecture starts at the lower benchmark (p_T-sorted) and during training advances towards the upper benchmark (generator-sorted).

The internal representation of the jets can be seen in fig. 4a. By comparing the internal representation of the sorting network and the p_T sorting shown in fig. 4b it can be seen that the sorting network learns an intrinsic structure which is different to the p_T sorting. It shows that the most important jets for the classification task (b_{j1} and b_{j2}) are arranged closer.



(a) network sorting



(b) p_T sorting

Figure 4: Structure of the p_T sorting and the sorting network at the end of the training. The x-axis comprises the true particle information, the y-axis comprises the slot of the sorting network prediction and p_T sorting.

6. Conclusion

We presented a two-stage deep learning architecture consisting of a network for sorting input objects and a subsequent network for data analysis. The sorting network combines the whole event information and explicitly pre-sorts the inputs of the analysis. The system of sorting and analysis network is then trained with an end-to-end reinforcement learning approach. Using the example of top-quark pair associated Higgs boson production, we show an improvement of the signal and background separation in comparison to conventional sorting of jets with respect to their transverse momenta.

Acknowledgments

We wish to thank Peter Fackeldey, Jonas Glombitza, Yannik Rath, Marcel Rieger, David Schmidt and Marcus Wirtz for valuable discussions. This work is supported by the Ministry of Innovation, Science and Research of the State of North Rhine-Westphalia, and by the Federal Ministry of Education and Research (BMBF).

References

[1] Erdmann M, Glombitza J and Walz D 2018 *Astropart. Phys.* **97** 46–53 (*Preprint* 1708.00647)
[2] Delaquis S *et al.* (EXO) 2018 *JINST* **13** P08023 (*Preprint* 1804.09641)
[3] Adams C *et al.* (MicroBooNE) 2019 *Phys. Rev.* **D99** 092001 (*Preprint* 1808.07269)
[4] Louppe G, Cho K, Becot C and Cranmer K 2019 *JHEP* **01** 057 (*Preprint* 1702.00748)
[5] Butter A, Kasieczka G, Plehn T and Russell M 2018 *SciPost Phys.* **5** 028 (*Preprint* 1707.08966)
[6] Erdmann M, Fischer B and Rieger M 2017 *JINST* **12** P08020 (*Preprint* 1706.01117)
[7] Erdmann M, Geiser E, Rath Y and Rieger M 2019 *Journal of Instrumentation* **14** P06006P06006 ISSN 1748-0221 URL http://dx.doi.org/10.1088/1748-0221/14/06/P06006
[8] Carrazza S and Dreyer F A 2019 Jet grooming through reinforcement learning (*Preprint* 1903.09644)
[9] Sjöstrand T *et al.* 2015 *Comput. Phys. Commun.* **191** 159–177 (*Preprint* 1410.3012)
[10] de Favereau J *et al.* (DELPHES 3) 2014 *JHEP* **02** 057 (*Preprint* 1307.6346)
[11] Klambauer G, Unterthiner T, Mayr A and Hochreiter S 2017 *arXiv e-prints* arXiv:1706.02515 (*Preprint* 1706.02515)