MULTIPLICATION OF SIMULATED EVENTS USING MACHINE LEARNING TECHNIQUES[*]

Karol Sowa, Wojciech Krupa, Tomasz Szumlak
Agnieszka Obłąkowska-Mucha

AGH University of Science and Technology
Faculty of Physics and Applied Computer Science
al. Mickiewicza 30, 30-059 Kraków, Poland

Received 30 September 2022, accepted 2 January 2023,
published online 15 February 2023

Nowadays, simulated data are commonly used in modern high-energy physics experiments. They are essential not only in determining certain performances but also in training machine learning algorithms. However, in some cases, such as rare heavy meson decays, generating data requires enormous computational resources. To speed up this process significantly, we propose a new method — to replicate simulated data using existing samples. Preliminary results of the algorithm are presented.

DOI:10.5506/APhysPolBSupp.16.3-A17

1. Introduction

In recent years, multivariate analysis and machine learning have played an increasingly important role in HEP applications, mainly in the classification process. Constantly improving, sophisticated algorithms obtain unprecedented performance in terms of efficiency and purity. However, most of them require large datasets to achieve the best possible result. The most common solution to this problem is using Monte Carlo simulations to model the signal processes. Nevertheless, such an approach can be problematic when it comes to rare and topologically complex processes, namely heavy meson decays. In that case, creating artificial data requires enormous computational resources and takes a lot of time.

Hence, we present a different approach to this problem — data augmentation. This method is already widely used in various ML applications, such as computer-aided image analysis [1]. The main idea is to use the existing data as input to the generative neural network. Thus, in the end, we can

[*] Presented at the XIV International Conference on *Beauty, Charm and Hyperon Hadrons*, Kraków, Poland, 5–10 June, 2022.

get a set of new events, with slightly different parameters. This process can be noticeably faster than time- and resource-consuming Monte Carlo simulations.

Two main types of network architecture that can be used to generate new data are: generative adversarial networks (GANs [2]) and variational auto-encoders (VAEs [3]). In this paper, the application of the latter ones is covered.

2. Network architecture

2.1. Classical auto-encoders

An auto-encoder (AE) is a specific type of neural network which learns how to efficiently compress the input data $\boldsymbol{x}$, encode it in a hidden layer (or layers) $\boldsymbol{z}$, and then reconstruct it in the best possible way at the output $\boldsymbol{r}$. It consists of two main parts: an *encoder* — responsible for the transformation of input data into its latent representation (often referred to as *code*), and a *decoder* — a part that takes care of the reversed process. The idea itself is not new, as it was already present in the 80s and 90s (*e.g.* Rumelhart, Hinton, and Williams 1986 [4]). In the beginning, auto-encoders were used for dimensionality reduction, feature extraction, or denoising. Typical auto-encoder architecture is presented in Fig. 1.

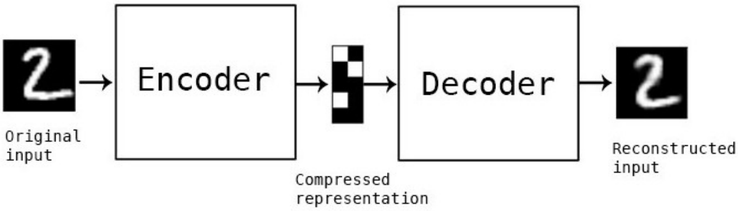


Fig. 1. Diagram of a typical auto-encoder used in image analysis [5].

In the most straightforward approach, one can treat the encoder and decoder as functions, $\boldsymbol{z} = f(\boldsymbol{x})$ and $\boldsymbol{r} = g(\boldsymbol{z})$, respectively. This interpretation, however, is not sufficient in most present applications. Instead, one takes a more general approach, using stochastic mapping: $q_{\text{encoder}}(\boldsymbol{z}|\boldsymbol{x})$ and $p_{\text{decoder}}(\boldsymbol{x}|\boldsymbol{z})$ [6]. Later, this generalisation will be crucial for understanding variational auto-encoders.

The classical auto-encoder has a shape of a bottleneck. It means that its latent space has a smaller dimension than the input. So, the encoding process will compress the data, while decoding will try to reconstruct it without losing much information. Such a procedure allows us to find the most important features in the dataset (feature extraction) and remove the unnecessary ones, *e.g.* the noise. One may notice that AE can be treated as enhanced PCA (as PCA transformation has to be linear).

2.2. Variational auto-encoders

Variational auto-encoders, contrary to classical ones, can generate new data. To do that, one samples the latent space, *i.e.* draws a data point from the distribution $p(\boldsymbol{z})$. Then, the network connects the drawn values of $\boldsymbol{z}$ to the features $\boldsymbol{x}$ via conditional distribution $p(\boldsymbol{x}|\boldsymbol{z})$ — a trained decoder. The latent space itself is decorrelated — the network should be able to learn all dependencies between variables and restore them during the decoding process. That is why all z_i-s can be treated as independent random variables.

It can be shown [3, 7] that the loss function for a variational auto-encoder is given by Eq. (1)

$$L_{\mathrm{VAE}} = -\mathbb{E}_{\boldsymbol{z}\sim q(\boldsymbol{z}|\boldsymbol{x})} \log p_{\mathrm{model}}(\boldsymbol{x}|\boldsymbol{z}) + D_{\mathrm{KL}}(q(\boldsymbol{z}|\boldsymbol{x})||p_{\mathrm{model}}(\boldsymbol{z}))\,. \tag{1}$$

This loss function is called ELBO [8] (Evidence Lower BOund) because one can recognise it as the variational log-likelihood lower bound with the reversed sign. The first term in Eq. (1) is the reconstruction log-likelihood that minimises the discrepancies between encoded and decoded data. The second term, expressed as the Kullback–Leibler divergence between posterior distribution $q(\boldsymbol{z}|\boldsymbol{x})$ and model prior $p(\boldsymbol{z})$, is responsible for latent space regularisation.

3. Results

Using the Python library PyTorch [9], we created the VAE model with a loss function in the form of Eq. (1). The tests were conducted using ROOT files containing all necessary variables in the $B_s^0 \to K^{*\mp} D_s^{*\pm}$ decay. The topology of this process is very complicated, so we started by examining the sub-decay: $D_s^{\pm} \to K^+K^-\pi^{\pm}$ and its final states' four-momenta were taken as the input to the auto-encoder.

The model successfully reconstructs the variables' distributions (Fig. 2), as well as the correlations between them (Fig. 3). However, a more detailed study was needed to check whether multiplied events are valid from the physics point of view. The invariant mass reconstruction from the 4-momentum coordinates seemed to be a reasonable test. Unfortunately, the resulting mass distribution does not follow the expected shape, as presented in Fig. 4 (a). The mass histogram does not peak around the $D_s^{\pm}$ table mass (1968.35 ± 0.07) MeV$/c^2$ [10]. The distribution is very spread out and a lot of negative values appear[1]. We tried to improve our model by adding a third, physically motivated term to our loss function

$$\lambda\,(m_{D_s,\mathrm{table}} - m_{KK\pi,\mathrm{rec}})^2\,, \qquad \lambda \in [0,1]\,, \tag{2}$$

[1] If the $E^2 - \boldsymbol{p}^2$ was negative for a given event, the mass would be calculated as $-\sqrt{|E^2 - \boldsymbol{p}^2|}$.

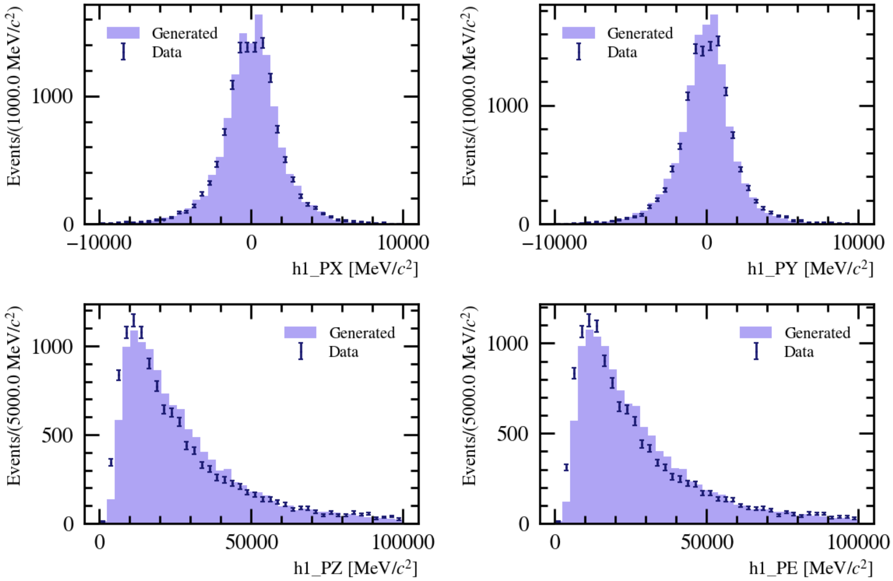
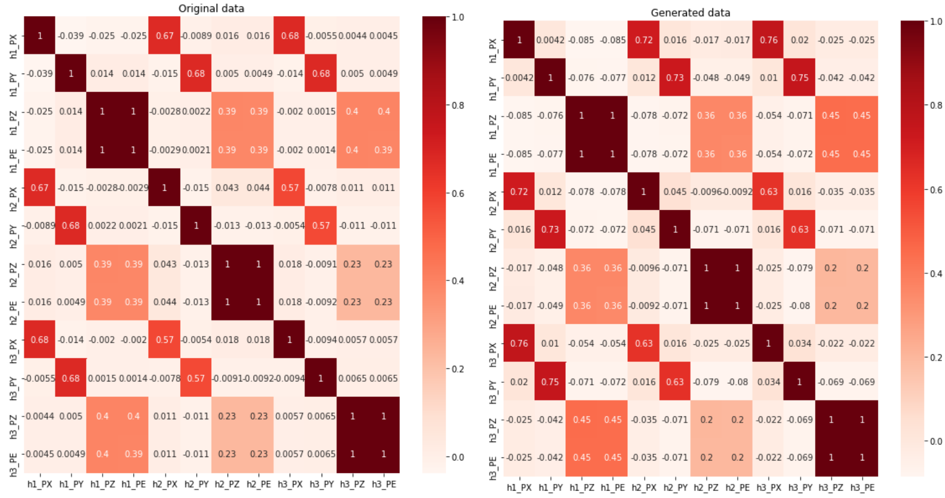


Fig. 2. 4-momentum distributions of one of the hadrons in the $D_s^{\pm} \to K^+K^-\pi^{\pm}$ final state — comparison of original samples and those generated using VAE.



Original data

	h1_PX	h1_PY	h1_PZ	h1_PE	h2_PX	h2_PY	h2_PZ	h2_PE	h3_PX	h3_PY	h3_PZ	h3_PE
h1_PX	1	-0.039	-0.025	-0.025	0.67	-0.0089	0.016	0.016	0.68	-0.0055	0.0044	0.0045
h1_PY	-0.039	1	0.014	0.014	-0.015	0.68	0.005	0.0049	-0.014	0.68	0.005	0.0049
h1_PZ	-0.025	0.014	1	1	-0.0028	0.0022	0.39	0.39	-0.002	0.0015	0.4	0.4
h1_PE	-0.025	0.014	1	1	-0.0029	0.0021	0.39	0.39	-0.002	0.0014	0.4	0.39
h2_PX	0.67	-0.015	-0.0028	-0.0029	1	-0.015	0.043	0.044	0.57	-0.0078	0.011	0.011
h2_PY	-0.0089	0.68	0.0022	0.0021	-0.015	1	-0.013	-0.013	-0.0054	0.57	-0.011	-0.011
h2_PZ	0.016	0.005	0.39	0.39	0.043	-0.013	1	1	0.018	-0.0091	0.23	0.23
h2_PE	0.016	0.0049	0.39	0.39	0.044	-0.013	1	1	0.018	-0.0092	0.23	0.23
h3_PX	0.68	-0.014	-0.002	-0.002	0.57	-0.0054	0.018	0.018	1	-0.0094	0.0057	0.0057
h3_PY	-0.0055	0.68	0.0015	0.0014	-0.0078	0.57	-0.0091	-0.0092	-0.0094	1	0.0065	0.0065
h3_PZ	0.0044	0.005	0.4	0.4	0.011	-0.011	0.23	0.23	0.0057	0.0065	1	1
h3_PE	0.0045	0.0049	0.4	0.39	0.011	-0.011	0.23	0.23	0.0057	0.0065	1	1

(a)

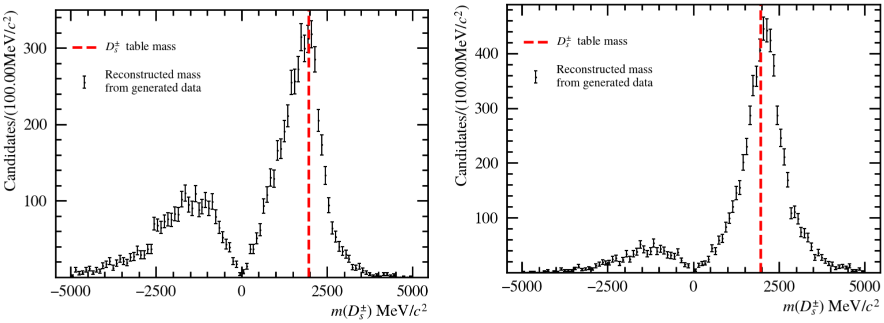
Generated data

	h1_PX	h1_PY	h1_PZ	h1_PE	h2_PX	h2_PY	h2_PZ	h2_PE	h3_PX	h3_PY	h3_PZ	h3_PE
h1_PX	1	0.0042	-0.085	-0.085	0.72	0.016	-0.017	-0.017	0.76	0.02	-0.025	-0.025
h1_PY	0.0042	1	-0.076	-0.077	0.012	0.73	-0.048	-0.049	0.01	0.75	-0.042	-0.042
h1_PZ	-0.085	-0.076	1	1	-0.078	-0.072	0.36	0.36	-0.054	-0.071	0.45	0.45
h1_PE	-0.085	-0.077	1	1	-0.078	-0.072	0.36	0.36	-0.054	-0.072	0.45	0.45
h2_PX	0.72	0.012	-0.078	-0.078	1	0.045	-0.0096	-0.0092	0.63	0.016	-0.035	-0.035
h2_PY	0.016	0.73	-0.072	-0.072	0.045	1	-0.071	-0.071	0.016	0.63	-0.071	-0.071
h2_PZ	-0.017	-0.048	0.36	0.36	-0.0096	-0.071	1	1	-0.025	-0.079	0.2	0.2
h2_PE	-0.017	-0.049	0.36	0.36	-0.0092	-0.071	1	1	-0.025	-0.08	0.2	0.2
h3_PX	0.76	0.01	-0.054	-0.054	0.63	0.016	-0.025	-0.025	1	0.034	-0.022	-0.022
h3_PY	0.02	0.75	-0.071	-0.072	0.016	0.63	-0.079	-0.08	0.034	1	-0.069	-0.069
h3_PZ	-0.025	-0.042	0.45	0.45	-0.035	-0.071	0.2	0.2	-0.022	-0.069	1	1
h3_PE	-0.025	-0.042	0.45	0.45	-0.035	-0.071	0.2	0.2	-0.022	-0.069	1	1

(b)

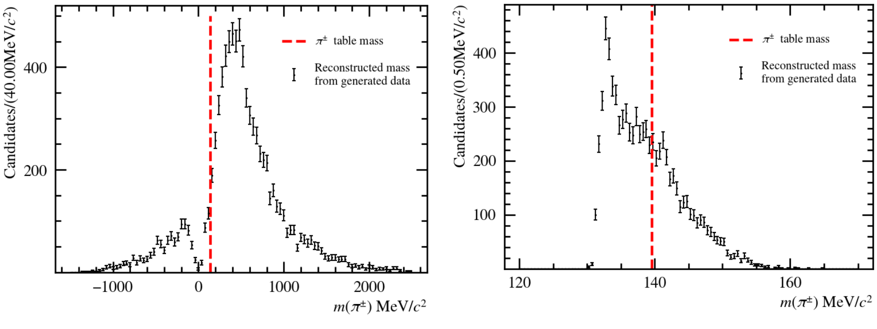
Fig. 3. Correlation matrices for: (a) original samples and (b) those generated using VAE. 4-momenta of $D_s^{\pm} \to K^+K^-\pi^{\pm}$ final states were used as input features.

where $m_{D_s,\text{table}}$ is a table mass of $D_s^\pm$ [10] and $m_{KK\pi,\text{rec}}$ is a mass calculated from reconstructed 4-momentum coordinates of the final-state particles. It improved our results significantly, see Fig. 4 (b), but negative values were still observed. Reconstructing the mass of a 3-particle system seemed too difficult for our network at the time. Hence, we tried a less challenging task: reconstructing the mass of a single charged pion. Surprisingly, even with the new loss function, the model failed to do it properly, see Fig. 5 (a). The breakthrough has come when we used squared 4-momentum coordinates as the input. It allowed us to obtain very promising results. The mass peak was centered around the pion table mass (139.57039 ± 0.00018) [10], the spread of the distribution was two orders of magnitude smaller, and no negative values were observed (Fig. 5 (b)).



(a) (b)

Fig. 4. Reconstructed mass of D_s candidates before (a) and after (b) adding a physically motivated term to loss function.



(a) (b)

Fig. 5. Reconstructed mass of $\pi^\pm$ candidates for: (a) 4-momentum coordinates as input, (b) squared 4-momentum coordinates as the input.

4. Conclusions and future plans

The results presented in this paper are preliminary, and further work is underway to refine the method. The reconstruction of 4-momentum distributions is at a satisfactory level. However, significant discrepancies remain when it comes to particle mass reconstruction. Further research will be done, and our model will be developed to achieve desired results for individual particles and, subsequently, for the whole decay chain.

This research was supported in part by National Research Centre, Poland (NCN), grants No. UMO-2018/31/N/ST2/01471, UMO2020/36/T/ST2/00168 and in part by PLGrid Infrastructure. We would also like to express our gratitude to Jakub Michczyński from the Faculty of Physics and Applied Computer Science, AGH UST for his help and knowledge sharing throughout the project.

REFERENCES

[1] C. Shorten, T.M. Khoshgoftaar, *J. Big Data* **6**, 60 (2019).

[2] I.J. Goodfellow *et al.*, arXiv:1406.2661 [stat.ML].

[3] D.P. Kingma, M. Welling, «Auto-encoding Variational Bayes» arXiv:1312.6114 [stat.ML].

[4] D.E. Rumelhart, G.E. Hinton, R.J. Williams, «Learning Internal Representations by Error Propagation» in: «Parallel Distributed Processing. Vol 1: Foundations», *MIT Press, Cambridge*, MA 1986.

[5] Will Badr, «Auto-encoder: What is it? and what is it used for?» https://towardsdatascience.com/auto-encoder-what-is-it-and-what-is-it-used-for-part-1-3e5c6f017726

[6] I. Goodfellow, Y. Bengio, A. Courville, «Deep Learning», *MIT Press*, 2016, http://www.deeplearningbook.org

[7] S. Levine, «Deep reinforcement learning», online lecture, https://youtu.be/UTMpM4orS30, 2020.

[8] S. Odaibo, arXiv:1907.08956 [cs.LG].

[9] PyTorch github, https://github.com/pytorch/pytorch

[10] Particle Data Group (R.L. Workman *et al.*), *Prog. Theor. Exp. Phys.* **2022**, 083C01 (2022).