Turbo-Sim: a generalised generative model with a physical latent space

Guillaume Quétant[ab*]
guillaume.quetant@unige.ch

Mariia Drozdova[ab]
mariia.drozdova@unige.ch

Vitaliy Kinakh[a]
vitaliy.kinakh@unige.ch

Tobias Golling[b]
tobias.golling@unige.ch

Slava Voloshynovkiy[a*]
svolos@unige.ch

[a]Department of Computer Science, University of Geneva, 1227 Carouge
[b]Department of Particle Physics, University of Geneva, 1205 Genève

Abstract

We present Turbo-Sim, a generalised autoencoder framework derived from principles of information theory that can be used as a generative model. By maximising the mutual information between the input and the output of both the encoder and the decoder, we are able to rediscover the loss terms usually found in adversarial autoencoders and generative adversarial networks, as well as various more sophisticated related models. Our generalised framework makes these models mathematically interpretable and allows for a diversity of new ones by setting the weight of each loss term separately. The framework is also independent of the intrinsic architecture of the encoder and the decoder thus leaving a wide choice for the building blocks of the whole network. We apply Turbo-Sim to a collider physics generation problem: the transformation of the properties of several particles from a theory space, right after the collision, to an observation space, right after the detection in an experiment.

1 Introduction

Deep learning models have proven to be promising in tasks consisting in the generation of new data following a desired, sometimes abstract, theoretical model or in the inference of certain parameters of this model given the observed data. Two predominant frameworks in such tasks are adversarial autoencoder (AAE) [1] and generative adversarial network (GAN) [2], which have been extensively studied, modified and redesigned in order to fit the problems they are applied to. In this paper we show a new framework, Turbo-Sim, that generalises the aforementioned models and gives a mathematical interpretation to the usual training strategies.

In both AAE and GAN there are two main variable spaces: a data space and a latent space. The key underlying idea of Turbo-Sim is to maximise the mutual information between the two spaces, assuming they are sampled from a joint, usually intractable, probability density. Once the density is carefully parametrized by deep networks, the mutual information can be decomposed and lower bounded in a number of ways, the maximisation of which gives rise to several loss terms, including those of AAEs and GANs, leaving the choice of the exact architecture of the building blocks open.[2] It

*G. Quétant and S. Voloshynovskiy are corresponding authors.

[2]A different approach discussed in [3, 4] gives rise to variational autoencoder (VAE) and links it to GAN.

Fourth Workshop on Machine Learning and the Physical Sciences (NeurIPS 2021).

also highlights the possibility of training the model in two directions called *direct* and *reverse*, which helps with the pseudo-invertibility of autoencoder-like models. This is expanded in Section 2.

Generative modelling is a crucial step in the process of scientific analysis, and in particular in particle physics. Indeed, the huge amount of data collected from experiments such as the Large Hadron Collider (LHC) needs to be compared with simulated data in order to make statistically meaningful conclusions. The fast growth of data taken by experiments is accompanied by a fast increase of the demand for the generation of simulated data, which is generally the bottleneck of the whole analysis pipeline. In this regard, machine learning aims to be an alternative to classical Monte Carlo generative models. Deep networks might for example be used in replacement of low-level feature simulations such as calorimeter hits [5, 6] or energy deposits in detector cells [7, 8], or of high-level feature simulations such as reconstructed observables from raw detector data [9]. In Section 3 we focus on the latter and make a direct comparison of our framework with the OTUS method [9].[3]

2 Turbo-Sim formalism

2.1 Base principle

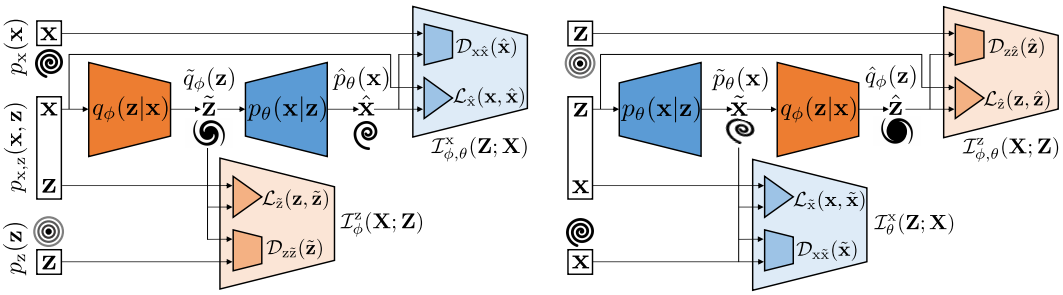

Figure 1: The *direct* (left) and *reverse* (right) directions of the Turbo-Sim framework.

Given a dataset composed of pairs $\{\mathbf{x}_i, \mathbf{z}_i\}_{i=1}^N$ sampled from the joint probability density $p_{\mathrm{x,z}}(\mathbf{x}, \mathbf{z})$ representing N realisations of the two random variables $\mathbf{X}$ and $\mathbf{Z}$, we show that one can maximise a parametrization of the true mutual information $I(\mathbf{X}; \mathbf{Z})$ between them in order to get the best approximation. The true mutual information is often intractable since the true joint density $p_{\mathrm{x,z}}(\mathbf{x}, \mathbf{z})$ and marginal densities $p_{\mathrm{x}}(\mathbf{x})$ and $p_{\mathrm{z}}(\mathbf{z})$ are usually unknown. In order to approximate the mutual information, the joint density is first rewritten as:

$$p_{\mathrm{x,z}}(\mathbf{x}, \mathbf{z}) = p_{\mathrm{z|x}}(\mathbf{z}|\mathbf{x})p_{\mathrm{x}}(\mathbf{x}) = p_{\mathrm{x|z}}(\mathbf{x}|\mathbf{z})p_{\mathrm{z}}(\mathbf{z}). \tag{1}$$

Both conditional densities $p_{\mathrm{z|x}}(\mathbf{z}|\mathbf{x})$ and $p_{\mathrm{x|z}}(\mathbf{x}|\mathbf{z})$ are then approximated by parametrized ones, $q_\phi(\mathbf{z}|\mathbf{x})$ and $p_\theta(\mathbf{x}|\mathbf{z})$ respectively. The marginal densities are denoted as:

$$\begin{aligned} \tilde{q}_\phi(\mathbf{z}) &= \mathbb{E}_{p_{\mathbf{x}}(\mathbf{x})}[q_\phi(\mathbf{z}|\mathbf{x})], & \hat{p}_\theta(\mathbf{x}) &= \mathbb{E}_{\tilde{q}_\phi(\mathbf{z})}[p_\theta(\mathbf{x}|\mathbf{z})], \\ \tilde{p}_\theta(\mathbf{x}) &= \mathbb{E}_{p_{\mathbf{z}}(\mathbf{z})}[p_\theta(\mathbf{x}|\mathbf{z})], & \hat{q}_\phi(\mathbf{z}) &= \mathbb{E}_{\tilde{p}_\theta(\mathbf{x})}[q_\phi(\mathbf{z}|\mathbf{x})]. \end{aligned}$$

This leads to multiple ways of decomposing the approximated mutual information.

2.2 Approximation for the *direct* direction

After some algebraic manipulations and by approximating $p_{\mathrm{z|x}}(\mathbf{z}|\mathbf{x}) \approx q_\phi(\mathbf{z}|\mathbf{x})$, one obtains the two following lower bounds to $I(\mathbf{X}; \mathbf{Z})$ (details can be found in Appendix A and B, as well as in [3, 4]):

[3] Code is available at `https://github.com/quetant/turbo-sim-supplementary`.

$$\mathcal{I}^{\mathrm{z}}_{\phi}(\mathbf{X};\mathbf{Z}) = \mathbb{E}_{p_{\mathrm{x},\mathrm{z}}(\mathbf{x},\mathbf{z})}\Big[\log \tfrac{q_\phi(\mathbf{z}|\mathbf{x})}{p_{\mathrm{z}}(\mathbf{z})}\tfrac{\tilde{q}_\phi(\mathbf{z})}{\tilde{q}_\phi(\mathbf{z})}\Big] \geq \underbrace{\mathbb{E}_{p_{\mathrm{x}}(\mathbf{x})}\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log q_\phi(\mathbf{z}|\mathbf{x})]}_{-\mathcal{L}_{\tilde{\mathrm{z}}}(\mathbf{z},\tilde{\mathbf{z}})} - \underbrace{D_{\mathrm{KL}}(p_{\mathrm{z}}(\mathbf{z})\|\tilde{q}_\phi(\mathbf{z}))}_{\mathcal{D}_{\mathrm{z}\tilde{\mathrm{z}}}(\tilde{\mathbf{z}})}, \quad (2)$$

$$\mathcal{I}^{\mathrm{x}}_{\phi,\theta}(\mathbf{Z};\mathbf{X}) = \mathbb{E}_{p_{\mathrm{x},\mathrm{z}}(\mathbf{x},\mathbf{z})}\Big[\log \tfrac{p_\theta(\mathbf{x}|\mathbf{z})}{p_{\mathrm{x}}(\mathbf{x})}\tfrac{\hat{p}_\theta(\mathbf{x})}{\hat{p}_\theta(\mathbf{x})}\Big] \geq \underbrace{\mathbb{E}_{p_{\mathrm{x}}(\mathbf{x})}\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}|\mathbf{z})]}_{-\mathcal{L}_{\hat{\mathrm{x}}}(\mathbf{x},\hat{\mathbf{x}})} - \underbrace{D_{\mathrm{KL}}(p_{\mathrm{x}}(\mathbf{x})\|\hat{p}_\theta(\mathbf{x}))}_{\mathcal{D}_{\mathrm{x}\hat{\mathrm{x}}}(\hat{\mathbf{x}})}. \quad (3)$$

One has thus to train the network in such a way as to maximise both $\mathcal{I}^{\mathrm{z}}_{\phi}(\mathbf{X};\mathbf{Z})$ and $\mathcal{I}^{\mathrm{x}}_{\phi,\theta}(\mathbf{Z};\mathbf{X})$ terms, with a possible weight factor α between them, in order to find the best parameters ϕ and θ of the encoder and the decoder respectively. This is achieved in the *direct* direction by minimising the following loss, leading to the left network shown in Fig. 1:

$$\bar{\mathcal{L}}^{\mathrm{Direct}}(\phi,\theta) = \mathcal{L}_{\tilde{\mathrm{z}}}(\mathbf{z},\tilde{\mathbf{z}}) + \mathcal{D}_{\mathrm{z}\tilde{\mathrm{z}}}(\tilde{\mathbf{z}}) + \alpha\,\mathcal{L}_{\hat{\mathrm{x}}}(\mathbf{x},\hat{\mathbf{x}}) + \alpha\,\mathcal{D}_{\mathrm{x}\hat{\mathrm{x}}}(\hat{\mathbf{x}}). \quad (4)$$

2.3 Approximation for the *reverse* direction

An analogous procedure with the approximation $p_{\mathrm{x}|\mathrm{z}}(\mathbf{x}|\mathbf{z}) \approx p_\theta(\mathbf{x}|\mathbf{z})$ gives the following two other lower bounds to $I(\mathbf{X};\mathbf{Z})$:

$$\mathcal{I}^{\mathrm{x}}_{\theta}(\mathbf{Z};\mathbf{X}) \geq \underbrace{\mathbb{E}_{p_z(\mathbf{z})}\mathbb{E}_{p_\theta(\mathbf{x}|\mathbf{z})}[\log p_\theta(\mathbf{x}|\mathbf{z})]}_{-\mathcal{L}_{\tilde{\mathrm{x}}}(\mathbf{x},\tilde{\mathbf{x}})} - \underbrace{D_{\mathrm{KL}}(p_{\mathrm{x}}(\mathbf{x})\|\tilde{p}_\theta(\mathbf{x}))}_{\mathcal{D}_{\mathrm{x}\tilde{\mathrm{x}}}(\tilde{\mathbf{x}})}, \quad (5)$$

$$\mathcal{I}^{\mathrm{z}}_{\phi,\theta}(\mathbf{X};\mathbf{Z}) \geq \underbrace{\mathbb{E}_{p_z(\mathbf{z})}\mathbb{E}_{p_\theta(\mathbf{x}|\mathbf{z})}[\log q_\phi(\mathbf{z}|\mathbf{x})]}_{-\mathcal{L}_{\hat{\mathrm{z}}}(\mathbf{z},\hat{\mathbf{z}})} - \underbrace{D_{\mathrm{KL}}(p_{\mathrm{z}}(\mathbf{z})\|\hat{q}_\phi(\mathbf{z}))}_{\mathcal{D}_{\mathrm{z}\hat{\mathrm{z}}}(\hat{\mathbf{z}})}, \quad (6)$$

and the *reverse* direction loss, weighted by β, leading to the right network shown in Fig. 1:

$$\bar{\mathcal{L}}^{\mathrm{Reverse}}(\phi,\theta) = \mathcal{L}_{\tilde{\mathrm{x}}}(\mathbf{x},\tilde{\mathbf{x}}) + \mathcal{D}_{\mathrm{x}\tilde{\mathrm{x}}}(\tilde{\mathbf{x}}) + \beta\,\mathcal{L}_{\hat{\mathrm{z}}}(\mathbf{z},\hat{\mathbf{z}}) + \beta\,\mathcal{D}_{\mathrm{z}\hat{\mathrm{z}}}(\hat{\mathbf{z}}). \quad (7)$$

2.4 Generic implementation

The general formalism of the Turbo-Sim framework allows to explain several state-of-the-art autoencoder and generative models. The first crucial point is that the intrinsic encoder and decoder architectures are not specified. Indeed they could be any deterministic or stochastic mappers, $q_\phi(\mathbf{z}|\mathbf{x})$ and $p_\theta(\mathbf{x}|\mathbf{z})$ respectively. Dense network might be used for vector data, convolutional network for images, graph network for point cloud data, or any more sophisticated architecture for a given specific data structure. The second point is that by turning on or off certain weights (α, β, γ, ...) of the general loss:

$$\bar{\mathcal{L}}^{\mathrm{Turbo}}(\phi,\theta) = \bar{\mathcal{L}}^{\mathrm{Direct}}(\phi,\theta) + \gamma\,\bar{\mathcal{L}}^{\mathrm{Reverse}}(\phi,\theta), \quad (8)$$

one can recover a wide range of common training strategies. Here we allow a factor γ between the two losses to weight them independently. As a simple example, standard AAE [1] is described by $\mathcal{D}_{\mathrm{z},\tilde{\mathrm{z}}}(\tilde{\mathbf{z}})$ approximated by a discriminator network on the latent space and $\mathcal{L}_{\hat{\mathrm{x}}}(\mathbf{x},\hat{\mathbf{x}})$ given by the supervised ℓ_p-norm on the data space assuming an exponential distribution for $p_\theta(\mathbf{x}|\mathbf{z}) \propto \exp\big(-\lambda\|\mathbf{x}-\hat{\mathbf{x}}\|_p^p\big)$, where $\hat{\mathbf{x}}$ is a function of $\mathbf{z}$. Notice that thanks to the α, β and γ weights and because of the λ factors found in exponential distributions, all eight terms of $\bar{\mathcal{L}}^{\mathrm{Turbo}}(\phi,\theta)$ receive an independent factor and thus can be weighted independently.

3 A generative model for particle physics

We apply our generalised Turbo-Sim framework to a collider physics transformation problem. In particular, we will focus on double top quarks production in proton–proton collision subsequently

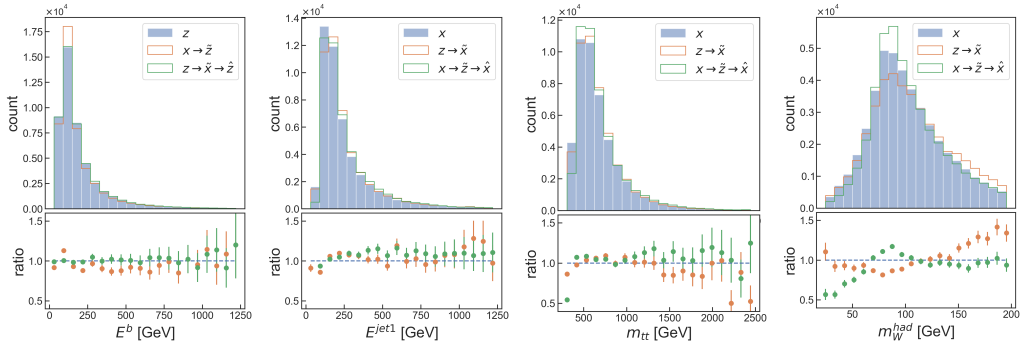

Figure 2: Distributions of a selection of observables for truth (blue), sampled (orange) and reconstructed (green) data.

decaying into b-quarks and W-bosons which in turn decay into a pair of light leptons and a pair of light quarks, $pp \to t\bar{t} \to e^- \bar{\nu}_e \, b\bar{b} \, u\bar{d}$ as described in [9].[4] The final state of this process is thus made of six particles whose four-momenta are taken to form the $\mathbf{Z}$ space. After parton shower, hadronisation and potential formation of jets in the detectors, the four-momenta assigned to the reconstructed objects are taken to form the $\mathbf{X}$ space. In this space, one loses information about the original particles, due to mainly two reasons. Firstly, the neutrino cannot be detected, hence it is indirectly defined by the missing transverse momentum, which must sum up to zero. Secondly, the four jets cannot be assigned to the four initial quarks since they all decay in statistically identical showers in the detector. They are thus treated as independent objects.

Transforming the random distributions of observables well defined by the rules of quantum field theory into somewhat modified distributions of the same observables due to the stochastic nature of the experiments is a common problem in particle physics generative modelling. The key point in this scenario is the word *transformation*. Indeed, standard simulation tools take the $\mathbf{Z}$ space as input and transform it by applying physically motivated stochastic processes in order to output the $\mathbf{X}$ space [10, 11, 12]. On the other hand, usual deep learning models are trained to mimic the same outputs by taking Gaussian, or any tractable distributions, as input. The paradigm shift introduced in [9] is to rather take this $\mathbf{Z}$ space as latent space for the deep network, and thus as input for the generative model. We follow the same line to train the full Turbo-Sim network shown in Fig. 1 and select the best model after a random hyperparameters search.[5] Details can be found in the code.

The distributions of a selection of observables resulting from our trained Turbo-Sim model are shown in Fig. 2, where sampled ($\tilde{\mathbf{z}}$ and $\tilde{\mathbf{x}}$) and reconstructed ($\hat{\mathbf{z}}$ and $\hat{\mathbf{x}}$) data are compared to the truth ($\mathbf{z}$ and $\mathbf{x}$). A quantitative comparison of samples with the OTUS method is shown in Tab. 1 by means of the Kolmogorov-Smirnov test, which defines a distance between two cumulative distributions. p_y^b, p_z^b, E^b are the momenta and energy of the b-quark, p_y^{jet1}, p_z^{jet1}, E^{jet1} are the momenta and energy of the leading jet. These variables, among others, are used during the training. From them we compute the masses of several underlying physical objects in order to verify how well the underlying physical correlations are learnt. m_{tt} is the top quark pair mass, m_W^{had} is the hadronic W-boson mass, m_t^{lep}, m_t^{had} are the leptonic and hadronic top quark masses. The reconstructed W-boson and top quark masses are computed from the four-momentum of the four jets, the electron and the reconstructed neutrino in the $\mathbf{X}$ space. All proper combinations of objects are compared to the true masses thanks to a chi-square test, the best one being kept. The neutrino longitudinal momentum is computed by solving $m_W^2 = (p^{e^-} + p^{\nu})^2$.

We observe that, although the OTUS method looks great at sampling the $\mathbf{Z}$ space, the Turbo-Sim method is better at sampling the $\mathbf{X}$ space, which is the main goal of a generative model. In addition, Turbo-Sim outperforms OTUS when it comes to the reconstruction of decayed objects, implying that the underlying physics has been better learnt.

[4] The dataset was shared with us and is now publicly available at `https://github.com/yiboyang/otus`.

[5] The random hyperparameters search was performed on an internal cluster with either nVidia Titan X, P100, RTX 2080 Ti, RTX 3090 or A100 GPUs, for a maximum of 12 hours.

Table 1: Kolmogorov-Smirnov distance to truth [$\times 10^{-2}$] for several observables including those of Fig. 2. A lower value means a higher accuracy.

	$\mathbf{Z}$ space			$\mathbf{X}$ space			Reconstructed physics			
Model	p_y^b	p_z^b	E^b	p_y^{jet1}	p_z^{jet1}	E^{jet1}	m_{tt}	m_W^{had}	m_t^{lep}	m_t^{had}
Turbo-Sim	5.28	7.28	3.96	**2.89**	10.3	**4.43**	**2.97**	**7.72**	**5.20**	**8.52**
OTUS	**1.59**	**1.23**	**2.76**	3.78	**2.39**	5.75	15.8	11.7	14.1	24.9

4 Discussion

In this work we presented a new autoencoder interpretation based on the maximisation of the mutual information between the latent space and the data space. We used this formalism to build a generative model for particle physics called Turbo-Sim and showed that it is able to compete with state-of-the-art methods such as OTUS, even outperforming it in critical tasks. It should be emphasised that these results are reached with very basic network building blocks: both the encoder and the decoder are only made of fully connected layers, as well as the Wasserstein gradient penalty discriminators. Future implementations with more expressive networks are expected to enhance the performance. One key improvement would be in the way stochasticity is handled. Instead of simply adding some noise to the input of the networks, one might use a layer-wise noise implementation following the EigenGAN architecture [13]. This way, one allows the model to learn how to apply different noise inputs to different hidden features of the data, thus controlling them separately. Moreover, in contrast with the OTUS method, we did not include any physically motivated constraints such as momentum alignment or mass conservation alongside our Turbo-Sim loss terms, while yielding comparable performance.

Another interesting application of such models is to use them for unfolding tasks. Indeed, an important aspect of particle physics analysis is also to get back from the observed data to the actual physics. In our language, this means going from the $\mathbf{X}$ space to the $\mathbf{Z}$ space. Thanks to the paradigm of using a physically meaningful latent space, i.e. the theoretical distributions of energy and momenta, it turns out that our Turbo-Sim model is also trained to achieve this task, but its performance is still below that of OTUS. However, because of the symmetrical nature of the network that manifests itself with the *direct* and *reverse* direction of training, it should be able to also learn this transformation properly. Nevertheless, a limitation could come from the data itself. Due to the highly stochastic nature of detector observations, the $\mathbf{X}$ space shows more smoothed out features than the $\mathbf{Z}$ space, which makes the inference direction harder. Future EigenGAN-like implementations might help handling this stochasticity, hence we are confident that a more expressive model will be able to perform equally well in both generation and unfolding tasks.

Acknowledgments and Disclosure of Funding

We would like to thank Sebastian Pina-Otey and Johnny Raine for fruitful discussions, as well as for their enlightening comments and feedback. This research was partially funded by the SNF Sinergia project (CRSII5_193716): Robust Deep Density Models for High-Energy Particle Physics and Solar Flare Analysis (RODEM).

References

[1] Alireza Makhzani et al. *Adversarial autoencoders*. 2015. arXiv: 1511.05644 [cs.LG].

[2] Ian J. Goodfellow et al. *Generative Adversarial Networks*. 2014. arXiv: 1406.2661 [stat.ML].

[3] Slava Voloshynovskiy et al. *Information bottleneck through variational glasses*. 2019. arXiv: 1912.00830 [cs.CV].

[4] Slava Voloshynovskiy et al. "Variational Information Bottleneck for Semi-Supervised Classification". In: *Entropy* 22.9 (2020). DOI: 10.3390/e22090943.

[5] ATLAS Collaboration. *Fast simulation of the ATLAS calorimeter system with Generative Adversarial Networks*. Tech. rep. Geneva: CERN, 2020. URL: https://cds.cern.ch/record/2746032.

[6] ATLAS Collaboration. *AtlFast3: the next generation of fast simulation in ATLAS*. 2021. arXiv: 2109.02551 [hep-ex].

[7] ATLAS Collaboration. *Deep generative models for fast shower simulation in ATLAS*. Tech. rep. Geneva: CERN, 2018. URL: http://cds.cern.ch/record/2630433.

[8] Erik Buhmann et al. "Getting High: High Fidelity Simulation of High Granularity Calorimeters with High Speed". In: *Computing and Software for Big Science* 5.13 (2021). DOI: 10.1007/s41781-021-00056-0.

[9] Jessica N. Howard et al. *Foundations of a Fast, Data-Driven, Machine-Learned Simulator*. 2021. arXiv: 2101.08944 [hep-ph].

[10] J. Alwall et al. "The automated computation of tree-level and next-to-leading order differential cross sections, and their matching to parton shower simulations". In: *Journal of High Energy Physics* 2014.7 (2014). DOI: 10.1007/jhep07(2014)079.

[11] S. Agostinelli et al. "Geant4—a simulation toolkit". In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 506.3 (2003), pp. 250–303. DOI: https://doi.org/10.1016/S0168-9002(03)01368-8.

[12] J. de Favereau et al. "DELPHES 3: a modular framework for fast simulation of a generic collider experiment". In: *Journal of High Energy Physics* 2014.2 (2014). DOI: 10.1007/jhep02(2014)057.

[13] Zhenliang He, Meina Kan, and Shiguang Shan. *EigenGAN: Layer-Wise Eigen-Learning for GANs*. 2021. arXiv: 2104.12476 [cs.CV].

A Lower bound to the true mutual information

In this section, we prove that the parametrization of the true mutual information between the $\mathbf{X}$ and the $\mathbf{Z}$ spaces $I(\mathbf{X};\mathbf{Z}) \simeq \mathcal{I}^{\mathbf{z}}_{\phi}(\mathbf{X};\mathbf{Z})$ is not only an approximation, but also a lower bound i.e. $I(\mathbf{X};\mathbf{Z}) \geq \mathcal{I}^{\mathbf{z}}_{\phi}(\mathbf{X};\mathbf{Z})$:

$$
\begin{aligned}
I(\mathbf{X};\mathbf{Z}) &= \mathbb{E}_{p_{\mathbf{x},\mathbf{z}}(\mathbf{x},\mathbf{z})}\left[\log \frac{p_{\mathbf{x},\mathbf{z}}(\mathbf{x},\mathbf{z})}{p_{\mathbf{x}}(\mathbf{x})p_{\mathbf{z}}(\mathbf{z})}\right] \\
&= \mathbb{E}_{p_{\mathbf{x},\mathbf{z}}(\mathbf{x},\mathbf{z})}\left[\log \frac{p_{\mathbf{z}|\mathbf{x}}(\mathbf{z}|\mathbf{x})}{p_{\mathbf{z}}(\mathbf{z})}\right] \\
&= \mathbb{E}_{p_{\mathbf{x},\mathbf{z}}(\mathbf{x},\mathbf{z})}\left[\log \frac{p_{\mathbf{z}|\mathbf{x}}(\mathbf{z}|\mathbf{x})}{p_{\mathbf{z}}(\mathbf{z})}\frac{q_{\phi}(\mathbf{z}|\mathbf{x})}{q_{\phi}(\mathbf{z}|\mathbf{x})}\right] \\
&= \mathbb{E}_{p_{\mathbf{x},\mathbf{z}}(\mathbf{x},\mathbf{z})}\left[\log \frac{q_{\phi}(\mathbf{z}|\mathbf{x})}{p_{\mathbf{z}}(\mathbf{z})}\right] + \mathbb{E}_{p_{\mathbf{x},\mathbf{z}}(\mathbf{x},\mathbf{z})}\left[\log \frac{p_{\mathbf{z}|\mathbf{x}}(\mathbf{z}|\mathbf{x})}{q_{\phi}(\mathbf{z}|\mathbf{x})}\right] \\
&= \mathbb{E}_{p_{\mathbf{x},\mathbf{z}}(\mathbf{x},\mathbf{z})}\left[\log \frac{q_{\phi}(\mathbf{z}|\mathbf{x})}{p_{\mathbf{z}}(\mathbf{z})}\right] + \mathbb{E}_{p_{\mathbf{x}}(\mathbf{x})}\left[D_{\mathrm{KL}}(p_{\mathbf{z}|\mathbf{x}}(\mathbf{z}|\mathbf{x})\|q_{\phi}(\mathbf{z}|\mathbf{x}))\right] \\
&\geq \mathbb{E}_{p_{\mathbf{x},\mathbf{z}}(\mathbf{x},\mathbf{z})}\left[\log \frac{q_{\phi}(\mathbf{z}|\mathbf{x})}{p_{\mathbf{z}}(\mathbf{z})}\right] \\
&= \mathbb{E}_{p_{\mathbf{x},\mathbf{z}}(\mathbf{x},\mathbf{z})}\left[\log \frac{q_{\phi}(\mathbf{z}|\mathbf{x})}{p_{\mathbf{z}}(\mathbf{z})}\frac{\tilde{q}_{\phi}(\mathbf{z})}{\tilde{q}_{\phi}(\mathbf{z})}\right] = \mathcal{I}^{\mathbf{z}}_{\phi}(\mathbf{X};\mathbf{Z}),
\end{aligned}
\tag{9}
$$

where the inequality holds because $D_{\mathrm{KL}}(p_{\mathbf{z}|\mathbf{x}}(\mathbf{z}|\mathbf{x})\|q_{\phi}(\mathbf{z}|\mathbf{x})) \geq 0$. Maximising the approximated mutual information ensures that the parametrized density $q_{\phi}(\mathbf{z}|\mathbf{x})$ is as close as possible to the true density $p_{\mathbf{z}|\mathbf{x}}(\mathbf{z}|\mathbf{x})$. The proves for $\mathcal{I}^{\mathrm{x}}_{\phi,\theta}(\mathbf{Z};\mathbf{X})$, $\mathcal{I}^{\mathrm{x}}_{\theta}(\mathbf{Z};\mathbf{X})$ and $\mathcal{I}^{\mathrm{z}}_{\phi,\theta}(\mathbf{X};\mathbf{Z})$ follow the exact same logic.

B Lower bounds to the approximate mutual information

In this section, we derive a lower bound to the parametrized approximation of the mutual information $\mathcal{I}^{\mathrm{z}}_{\phi}(\mathbf{X};\mathbf{Z})$ by decomposing it for the *direct* direction as:

$$
\begin{aligned}
\mathcal{I}^{\mathrm{z}}_{\phi}(\mathbf{X};\mathbf{Z}) &= \mathbb{E}_{p_{\mathrm{x},\mathrm{z}}(\mathbf{x},\mathbf{z})}\left[\log \frac{q_{\phi}(\mathbf{z}|\mathbf{x})}{p_{\mathrm{z}}(\mathbf{z})}\frac{\tilde{q}_{\phi}(\mathbf{z})}{\tilde{q}_{\phi}(\mathbf{z})}\right] \\
&= \mathbb{E}_{p_{\mathrm{x},\mathrm{z}}(\mathbf{x},\mathbf{z})}[\log q_{\phi}(\mathbf{z}|\mathbf{x})] - \mathbb{E}_{p_{\mathrm{x},\mathrm{z}}(\mathbf{x},\mathbf{z})}\left[\log \frac{p_{\mathrm{z}}(\mathbf{z})}{\tilde{q}_{\phi}(\mathbf{z})}\right] - \mathbb{E}_{p_{\mathrm{x},\mathrm{z}}(\mathbf{x},\mathbf{z})}[\log \tilde{q}_{\phi}(\mathbf{z})] \\
&= \mathbb{E}_{p_{\mathrm{x}}(\mathbf{x})}\mathbb{E}_{p_{\mathrm{z}|\mathrm{x}}(\mathbf{z}|\mathbf{x})}[\log q_{\phi}(\mathbf{z}|\mathbf{x})] - \mathbb{E}_{p_{\mathrm{z}}(\mathbf{z})}\left[\log \frac{p_{\mathrm{z}}(\mathbf{z})}{\tilde{q}_{\phi}(\mathbf{z})}\right] - \mathbb{E}_{p_{\mathrm{z}}(\mathbf{z})}[\log \tilde{q}_{\phi}(\mathbf{z})] \\
&= \mathbb{E}_{p_{\mathrm{x}}(\mathbf{x})}\mathbb{E}_{p_{\mathrm{z}|\mathrm{x}}(\mathbf{z}|\mathbf{x})}[\log q_{\phi}(\mathbf{z}|\mathbf{x})] - D_{\mathrm{KL}}(p_{\mathrm{z}}(\mathbf{z})\|\tilde{q}_{\phi}(\mathbf{z})) + H(p_{\mathrm{z}}(\mathbf{z});\tilde{q}_{\phi}(\mathbf{z})) \\
&\geq \mathbb{E}_{p_{\mathrm{x}}(\mathbf{x})}\mathbb{E}_{p_{\mathrm{z}|\mathrm{x}}(\mathbf{z}|\mathbf{x})}[\log q_{\phi}(\mathbf{z}|\mathbf{x})] - D_{\mathrm{KL}}(p_{\mathrm{z}}(\mathbf{z})\|\tilde{q}_{\phi}(\mathbf{z})) \\
&\simeq \mathbb{E}_{p_{\mathrm{x}}(\mathbf{x})}\mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})}[\log q_{\phi}(\mathbf{z}|\mathbf{x})] - D_{\mathrm{KL}}(p_{\mathrm{z}}(\mathbf{z})\|\tilde{q}_{\phi}(\mathbf{z})),
\end{aligned}
\tag{10}
$$

where the inequality holds because $H(p_{\mathrm{z}}(\mathbf{z});\tilde{q}_{\phi}(\mathbf{z})) \geq 0$. Analogous derivations can be performed for the three remaining approximations:

$$
\begin{aligned}
\mathcal{I}^{\mathrm{x}}_{\phi,\theta}(\mathbf{Z};\mathbf{X}) &= \mathbb{E}_{p_{\mathrm{x},\mathrm{z}}(\mathbf{x},\mathbf{z})}\left[\log \frac{p_{\theta}(\mathbf{x}|\mathbf{z})}{p_{\mathrm{x}}(\mathbf{x})}\frac{\hat{p}_{\theta}(\mathbf{x})}{\hat{p}_{\theta}(\mathbf{x})}\right] \\
\mathcal{I}^{\mathrm{x}}_{\theta}(\mathbf{Z};\mathbf{X}) &= \mathbb{E}_{p_{\mathrm{x},\mathrm{z}}(\mathbf{x},\mathbf{z})}\left[\log \frac{p_{\theta}(\mathbf{x}|\mathbf{z})}{p_{\mathrm{x}}(\mathbf{x})}\frac{\tilde{p}_{\theta}(\mathbf{x})}{\tilde{p}_{\theta}(\mathbf{x})}\right] \\
\mathcal{I}^{\mathrm{z}}_{\phi,\theta}(\mathbf{X};\mathbf{Z}) &= \mathbb{E}_{p_{\mathrm{x},\mathrm{z}}(\mathbf{x},\mathbf{z})}\left[\log \frac{q_{\phi}(\mathbf{z}|\mathbf{x})}{p_{\mathrm{z}}(\mathbf{z})}\frac{\hat{q}_{\phi}(\mathbf{z})}{\hat{q}_{\phi}(\mathbf{z})}\right]
\end{aligned}
$$

Societal impacts

Since Turbo-Sim aims to improve the performance of generative models such as AAEs and GANs, it may lead to any negative usage of these models. However, the framework is here developed for particle physics simulations, which do not pose any ethical problem. In addition, the goal of such fast simulator is to minimise the computing resources needed to generated many samples, which is a step forward in environmental protection, as long as the training process does not compensate for all the gain.

Paper checklist

1. For authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [YES]
   (b) Have you read the ethics review guidelines and ensured that your paper conforms to them? [YES]
   (c) Did you discuss any potential negative societal impacts of your work? [YES]
   (d) Did you describe the limitations of your work? [YES] (Self-explained by definition of the scope of the framework and in Sec. 4)
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [YES] (In Sec. 2)
   (b) Did you include complete proofs of all theoretical results? [YES] (In Sec. 2)
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [YES] (URL in Sec. 1)
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [YES] (In the provided code)
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [YES]
   (d) Did you include the amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [YES] (In Sec. 3)
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [YES] (In Sec. 3)
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [YES] (In Sec. 3)
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] (No such possibility in particle physics data)
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]