CMS Physics Analysis Summary

Contact: cms-conveners-ml@cern.ch 2024/09/23

Wasserstein normalized autoencoder

The CMS Collaboration

Abstract

A novel approach to unsupervised jet tagging is presented for the CMS experiment at the CERN LHC. The Wasserstein normalized autoencoder (WNAE) is a normalized probabilistic model that minimizes the Wasserstein distance between the probability distribution of the training data and the Boltzmann distribution of the reconstruction error of the autoencoder. Trained on jets of particles from simulated standard model processes, the WNAE is shown to learn the probability distribution of the input data, in a fully unsupervised fashion, in order to effectively identify new physics jets as anomalies. This algorithm has been developed and applied in the context of a recent search for semivisible jets. The model consistently demonstrates stable, convergent training and achieves strong classification performance across a wide range of signals, improving upon standard normalized autoencoders, while remaining agnostic to the signal. The WNAE directly tackles the problem of outlier reconstruction, a common failure mode of autoencoders in anomaly detection tasks.

1 Introduction

Unsupervised machine learning algorithms have proven to be powerful tools in the search for new physics at the LHC [1, 2]. They can effectively separate known standard model (SM) physics events (background) from potential signal events arising from interactions beyond the SM, without relying on the simulation of specific signal hypotheses. Autoencoders (AEs) are frequently used for unsupervised learning tasks. Outlier reconstruction is one of the main challenges when using AE-based unsupervised machine learning algorithms for anomaly detection. If the autoencoder is able to reconstruct outliers—examples that are not part of the training data—its performance as a discriminator is degraded. The issue of complexity bias [3] can be thought of as a type of outlier reconstruction. In this note, we introduce the Wasserstein normalized autoencoder (WNAE), an evolution of the normalized autoencoder (NAE) algorithm [4], to solve these challenges. A search for semivisible jets (SVJs) [5] in the CMS experiment [6] is used as a case study to demonstrate the effectiveness of the WNAE in separating signal from background events.

The note is organized as follows. Section 2 provides a brief overview of the SVJ search and the challenges it poses for anomaly detection. Section 3 discusses the use and limitations of AEs as anomaly detection tools, focusing on the outlier reconstruction problem. In Section 4, a brief review of the NAE algorithm and its application to anomaly detection in jet physics is given. In Section 5, the WNAE algorithm is introduced and its advantages over the NAE are discussed. In Section 6, the results of the WNAE applied to the SVJ search are shown. Finally, Section 7 discusses the implications of the findings and possible future directions.

2 Semivisible jets

Hidden valley models [7] are new physics scenarios in which the SM sector is expanded with a dark sector, with its own particles and forces. The dark sector particles do not carry charge under any SM gauge group, and similarly SM particles are not charged under the dark gauge group. Hidden valley theories may, under certain conditions, lead to an experimental signature known as a semivisible jet [5]. In the case where the dark sector has a new non-Abelian SU(N) confining interaction, it would exhibit showering and hadronization qualitatively similar to SM quantum chromodynamics (QCD). In analogy with the SM sector, the new confining force and the new particles charged under it are referred to as dark QCD and dark quarks, respectively. More specifically, we consider a dark QCD interaction SU(N_c^{dark}), with a confinement scale Λ_{dark} and N_f^{dark} flavors of dark quark χ with masses m_χ, which communicates with the SM via a mediator with mass m_Φ. In such a model, QCD-like showers occur when $\Lambda_{\text{dark}} \ll m_\Phi$ and $m_\chi \sim \Lambda_{\text{dark}}$. The dark quarks therefore form bound states (dark hadrons), some of which may decay back to SM particles to produce a visible signature, depending on the symmetries of the theory. Stable dark hadrons are invisible and escape detection, giving rise to missing transverse momentum. The ensemble of stable and unstable dark hadrons produced in the dark QCD shower is referred to as an SVJ.

CMS has published a search for resonant production of pairs of SVJs [8], and t-channel production of SVJs by a bifundamental scalar mediator is also possible [9]. The latter production mode is considered in this paper. The fraction of stable dark hadrons in the dark QCD shower can be parametrized by the invisible fraction r_{inv}:

$$r_{\text{inv}} = \left\langle \frac{\text{Number of stable dark hadrons}}{\text{Total number of dark hadrons}} \right\rangle \tag{1}$$

SVJs may have different radiation patterns than SM jets, which can be exploited to identify them. The details depend on the exact spectrum of the dark hadrons, which in turn depends on N_c^{dark}, N_f^{dark}, Λ_{dark}, and the non-perturbative physics in the dark sector, all of which are unknown. Therefore, anomaly detection, which does not necessitate a detailed simulation of every target signal hypothesis in this large parameter space, is a powerful complement to traditional, supervised searches for SVJs [8].

3 Autoencoders for anomaly detection

Autoencoders [10] are a class of neural networks that is intended to learn a compressed representation of the input data. This is usually achieved by mapping the input feature space to a lower-dimensional latent space via an encoder network, and mapping this latent space to an output space with the same dimension as the input space via a decoder network. Autoencoders are typically trained by minimizing a loss function that penalizes the difference between the input and output, referred to as the reconstruction error of the AE.

Crucially, when AEs are used for new physics searches, the network is only exposed to background SM events during training. If the network is then able to learn a compressed representation of the background events, it will be able to reconstruct them with low error. However, since the new physics signal events are not part of the training set, the AE should be unable to reconstruct them well, resulting in a larger average reconstruction error. Ideally, the reconstruction error should be a function of the probability density of the training data: the lower the probability density of the training data, the higher the reconstruction error. The reconstruction error itself can therefore be used as a summary statistic to discriminate signal from background events.

While this reasoning holds for many practical applications of AEs for anomaly detection, it does not follow in general that achieving low reconstruction error on the background examples is sufficient for an AE to effectively identify anomalies, as discussed in the next section. This shortcoming is addressed by the WNAE algorithm, which is the main focus of this note.

3.1 Training setup

Simulated events are used to train and evaluate the performance of the machine learning models discussed in this note. Considerations for training directly on experimental data are discussed at the end of this section. Background and SVJ signal events are generated at parton level with MADGRAPH5_aMC@NLO 2.6.5 [11], final state quarks are hadronized with PYTHIA 8.240 [12], and the interactions of the resulting particles with the CMS detector [6] are simulated using GEANT4 [13]. The hadronization in the dark sector during signal generation follows the procedure described in Ref. [8], and all dark hadrons have a fixed mass, 20 GeV. Because of the hadronization in the dark sector and then in the SM sector, the substructure of SVJs is expected to be different than that of QCD jets. In particular, SVJs are expected to be wide, and therefore jets are reconstructed with the anti-k_{T} algorithm with radius parameter $R = 0.8$ [14, 15]. Each jet is represented by the following features, describing its substructure:

- the minor and major axes [16], which characterize the elliptical shape of the jet in the η-ϕ plane;
- the first energy flow polynomial (EFP1) [17], a multiparticle energy correlator that directly results from infrared and collinear safety;
- the $C_2^{(0.5)}$ energy correlation function [18], a ratio of two ratios of energy correlation

functions that can be used to determine if a jet has two hard subjets;

- the transverse momentum dispersion $p_{\mathrm{T}}^{\mathrm{D}}$ [16], which describes the spread of transverse momenta of the jet constituents;
- the softdrop mass [19], the mass of the jet after soft wide-angle radiation has been removed from the jet; and
- the 2- and 3-subjettiness τ_2 and τ_3 [20], describing the compatibility of the jet with a two- or three-prong structure, respectively;

for a total of $p = 8$ inputs to the AE. These input features were chosen to exploit the difference in shape (minor and major axes), energy distribution ($p_{\mathrm{T}}^{\mathrm{D}}$ and EFP1), prong structure ($C_2^{(0.5)}$, τ_2, τ_3), and mass of the high-momentum core of the jet (softdrop mass) expected between SM jets and SVJs. For each event, the two jets with highest transverse momenta are used for training and evaluation. In addition, among the two selected jets, only jets identified via generator-level information as originating from dark sector particles are considered as anomalous signal jets when evaluating the network performance. The inputs are pre-processed via quantile normalization, as implemented in the SCIKIT-LEARN package [21], to follow a normal distribution. The architecture is a fully connected network with five hidden layers. All layers have ten nodes, except for the third one (the bottleneck), which has six. The reconstruction of the AE is parametrized by the weights θ. The reconstruction error l_θ for an example x is defined as the mean squared error (MSE) between the features x_i of x and the features $\hat{x}_{\theta,i}$ of the reconstructed output example:

$$l_\theta(x) = \frac{1}{p}\sum_{i=1}^{p}\left(x_i - \hat{x}_{\theta,i}\right)^2. \tag{2}$$

The loss function of the AE is the reconstruction error averaged over the batch:

$$\mathcal{L}_{\mathrm{AE}} = \frac{1}{N}\sum_{j=1}^{N} l_\theta(x^{(j)}), \tag{3}$$

where $x^{(j)}$ denotes the j-th example in the batch and N is the number of training examples in a batch. The background dataset is split into three independent parts: a training set on which the loss is minimized, a validation set used to monitor the training, and a test set used to evaluate the performance of the network. The AE is trained until the loss function evaluated on the validation set ceases to improve. The AE is then tested using multiple SVJ signal models, as well as the background test sample.

Though simulated events were used for the purpose of developing the WNAE, in practice it may be preferable to use observed data directly for training, in order to limit biases arising from differences between data and simulation. However, the presence of an anomaly from new physics events in the training dataset is found to reduce performance. When a control region without any new physics events can be defined, it is possible to train directly on the observed data in this region. When no assumption at all can be made about the new physics signature, such as in the case of triggering, alternative solutions may exist. The autoencoder associates low probability density regions with high reconstruction error; because anomalies necessarily have low probability density, they will still tend to have relatively high reconstruction error when training a WNAE on a dataset containing them. Therefore, the dataset made of a given fraction of examples with the lowest reconstruction error will have a reduced proportion of

anomalous data and thus could be used to retrain the WNAE in a self-supervised approach. We leave the development of such a procedure for future work.

3.2 Outlier reconstruction

Outlier reconstruction occurs when an AE, while learning to reconstruct the background examples, also learns to reconstruct pockets of examples outside the training data. More formally: let $\mathcal{B}$ and $\mathcal{S}$ be the supports of the background and signal distributions, respectively. The AE is trained to achieve minimum reconstruction error on background events $x_b \in \mathcal{B}$. Let $\mathcal{E}$ be the set of examples for which the AE achieves reconstruction error below some threshold. Since there is no constraint on the reconstruction error of signal events $x_s \in \mathcal{S}$, there can be a sizable overlap $\mathcal{S} \cap \mathcal{E}$. This overlap can in general be at least partly disjoint from the training data: $(\mathcal{S} \cap \mathcal{E}) - \mathcal{B} \neq \emptyset$. Examples drawn from this region of phase space will be assigned low reconstruction error by the AE, even though they are outside the training data. Thus, these examples and the background will both have low reconstruction error, decreasing the performance of the AE's reconstruction error as a discriminator. This is illustrated in Fig. 1.

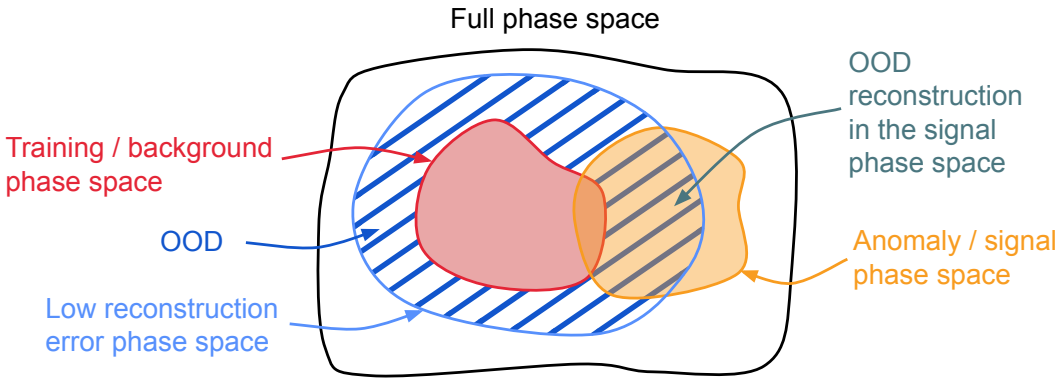

Figure 1: Schematic visualization of the outlier reconstruction failure mode. Signal events drawn from the hatched area are reconstructed well by the AE, despite not being part of the training set, and thus will not be separated from the background. The AE training is assumed to have converged such that the background is reconstructed well.

This issue is found to be especially impactful in the case of the SVJ search. While previous studies have shown that AEs are an effective tool for the task of discriminating between the signal and the SM QCD multijet background [22], the performance is found to degrade significantly when including jets originating from t quarks ($t\bar{t}$ background). This is shown by training an AE solely on the $t\bar{t}$ background.

The receiver operator characteristic (ROC) curve compares the true and false positive rates for the AE reconstruction error used as a discriminator, and the area under the ROC curve (AUC) is used to quantify the performance of the AE's reconstruction error as a discriminator. As shown in Fig. 2, when achieving minimum reconstruction error on the $t\bar{t}$ background, the AE is unable to discriminate between the SVJ signal and the $t\bar{t}$ background, and the AUC score is close to 0.5. In order to check that this behavior does not occur because the bottleneck was too large, the same experiment was repeated with a bottleneck of only two nodes. The results were found to be consistent with a bottleneck size of six, indicating that the issue does not arise from insufficient compression of the data.

The AUC scores evolve in nontrivial ways throughout the training. As shown in Fig. 2, the AUC score improves during the first stages of the training, indicating that the AE is performing as desired. However, after further training, the AUC score starts to degrade, and eventually

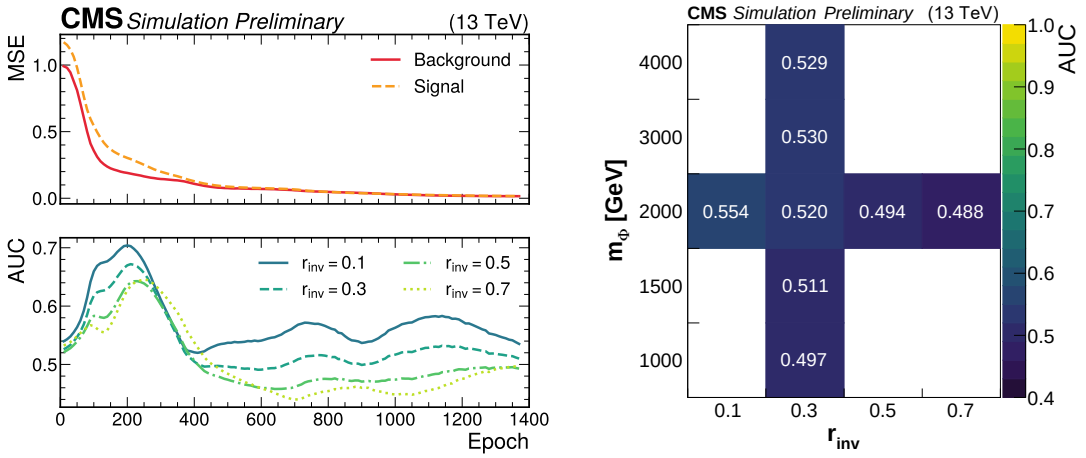

Figure 2: Left: the reconstruction error (upper panel) and the AUC scores (lower panel) for the AE trained on the $t\bar{t}$ background, evaluated during each training epoch on $t\bar{t}$ background jets and signal models with $m_\Phi = 2000\,\text{GeV}$ and $r_{\text{inv}} = 0.3$ (upper) or $r_{\text{inv}} = 0.1, 0.3, 0.5, 0.7$ (lower). Right: The AUC scores for the same AE, evaluated for the epoch with the minimal background reconstruction error, for the classification of several SVJ signal hypotheses against the $t\bar{t}$ background. The AUC scores are close to 0.5, indicating that the AE is unable to discriminate between the SVJ signal and the $t\bar{t}$ background.

converges to a value close to 0.5. This is a clear indication that the requirement to minimize the reconstruction error on the $t\bar{t}$ background is too weak to achieve good discrimination; it does not prevent the AE from also reconstructing signal events. One of the key elements of this note is a proposed metric that is able to quantify the degree of outlier reconstruction without relying on the signal in any form during training, discussed in Section 4. The second major part of this work in Section 5 is to use this metric as the direct target for the training, building a more robust anomaly detection algorithm.

4 The normalized autoencoder paradigm

4.1 Normalized autoencoders

Normalized autoencoders provide a systematic mechanism to suppress outlier reconstruction. The NAE algorithm was first proposed in Ref. [4], and it was first applied to jet tagging in high energy physics in Ref. [23]. The core principle is to force the neural network to learn the probability distribution of training events, rather than simply requiring low reconstruction error. This coincides with the requirement that the AE learns to reconstruct only examples that are present in the training data, thus directly tackling the problem of outlier reconstruction.

In analogy with energy-based models, often used for generative modeling, the NAE is assigned a normalized probability distribution $p_\theta(x)$, parametrized by the AE weights θ. Following the maximum entropy principle [24], the probability distribution is chosen as the Boltzmann distribution from statistical mechanics:

$$p_\theta(x) = \frac{1}{\Omega_\theta} \exp\left(-E_\theta(x)\right), \tag{4}$$

where $E_\theta(x)$ is the energy of the system and Ω_θ is the partition function:

$$\Omega_\theta = \int_{\mathcal{B}} \mathrm{d}x \exp(-E_\theta(x)). \tag{5}$$

The energy of the system is defined as the reconstruction error of the AE:

$$E_\theta(x) = l_\theta(x). \tag{6}$$

The problem is then recast as a maximum likelihood estimation problem, where the goal is to find the weights θ that maximize the likelihood of the training data, with probability distribution denoted p_{data}. As is common, the negative log likelihood is minimized instead:

$$\mathbb{E}_{x\sim p_{\text{data}}(x)}[-\log p_\theta(x)] = \mathbb{E}_{x\sim p_{\text{data}}(x)}[E_\theta(x)] + \log \Omega_\theta. \tag{7}$$

Calculating the partition function Ω_θ is in general intractable, but the problem can be circumvented when using gradient descent to find the optimum, since the gradient of the partition function can be calculated:

$$\begin{aligned}
\nabla_\theta \log \Omega_\theta &= \frac{1}{\Omega_\theta}\nabla_\theta \Omega_\theta = \frac{1}{\Omega_\theta}\int_{\mathcal{B}} \mathrm{d}x \nabla_\theta \exp(-E_\theta(x)) && (8)\\
&= \int_{\mathcal{B}} \mathrm{d}x \frac{1}{\Omega_\theta}\exp(-E_\theta(x))\nabla_\theta(-E_\theta(x)) && (9)\\
&= \int_{\mathcal{B}} \mathrm{d}x p_\theta(x)\nabla_\theta(-E_\theta(x)) && (10)\\
&= -\mathbb{E}_{x\sim p_\theta(x)}[\nabla_\theta E_\theta(x)], && (11)
\end{aligned}$$

which is the expectation value of the AE loss over the probability distribution $p_\theta(x)$. The gradient of the loss can thus be calculated by sampling from the probability distribution $p_\theta(x)$. This can be done via a Markov chain Monte Carlo (MCMC) algorithm. Tuning the parameters of the MCMC is a crucial and delicate step in applying the NAE approach, which will be discussed in more detail in Section 4.2.3. The loss function to be minimized is then:

$$\mathcal{L}_{\text{NAE}} = \mathbb{E}_{x\sim p_{\text{data}}(x)}[E_\theta(x)] - \mathbb{E}_{x'\sim p_\theta(x')}[E_\theta(x')]. \tag{12}$$

Both terms in Eq. (12) are expectation values of the AE loss. The first term is the expectation value over the training data x (the usual AE loss), referred to as the positive energy. The second term is the expectation value over samples x' drawn from the probability distribution $p_\theta(x)$, referred to as the negative energy. The NAE loss is thus the difference between the positive and negative energy, respectively denoted E_+ and E_-:

$$\mathcal{L}_{\text{NAE}} \equiv E_+ - E_-. \tag{13}$$

In analogy, the training examples are referred to as positive examples, and examples drawn from $p_\theta(x)$ are referred to as negative examples. In this way, the NAE is not trained just to reconstruct background events well, but rather to learn the probability distribution of the training data. The NAE is therefore penalized for having low reconstruction error in regions with low probability density of the training data, hence suppressing the reconstruction of examples outside the background support $\mathcal{B}$. It is worth emphasizing that the whole procedure is still fully unsupervised. The NAE is trained only on background events, and the signal is not used in any form during training.

4.2 Markov chain Monte Carlo

The key feature of the normalized autoencoder paradigm is to estimate the autoencoder probability p_θ. This is achieved by running an MCMC algorithm to obtain a set of examples following the p_θ distribution. In this section, we describe the chosen MCMC algorithm, an approach to MCMC initialization, and finally the tuning of the MCMC hyperparameters.

4.2.1 MCMC algorithm

The MCMC algorithm employed for this model is the Langevin Monte Carlo algorithm. First, a starting set of points $\{x\}_0$ is drawn from an initial distribution, following the procedure laid out in Section 4.2.2. Using Langevin dynamics, a new set of points is then proposed from the gradient of the target probability distribution. The process is repeated N_{MCMC} times, until $\{x\}_{N_{\text{MCMC}}}$ is representative of the target distribution, in this case p_θ. The Langevin dynamics MCMC equation is:

$$x_{i+1} = x_i + \lambda \nabla_x \log p_\theta(x_i) + \sigma \epsilon_i \tag{14}$$

where λ is the step size, σ is the noise coefficient, ϵ_i is an independent draw from a normal distribution on $\mathbb{R}^d$ with mean 0 and covariance matrix equal to the $d \times d$ identity matrix, and d denotes the number of dimensions of the space over which p_θ is defined, $p_\theta : \mathbb{R}^d \mapsto \mathbb{R}$. We denote the probability distribution of the sample $\{x\}_i$ as ρ_i. With the choice:

$$\sigma = \sqrt{2\lambda}, \tag{15}$$

and an infinite number of steps, the probability ρ_i converges towards the target probability p_θ: $\lim_{i\to\infty} \rho_i = p_\theta$. In practice, p_θ needs to be estimated at every gradient descent step, once every batch, while training an NAE. For that reason, the number of MCMC steps cannot be too large, and the probability p_θ can only be approximated. In order to enhance the gradient term for MCMC with a finite number of steps, the temperature T is introduced:

$$x_{i+1} = x_i + \frac{\lambda}{T} \nabla_x \log p_\theta(x_i) + \sigma \epsilon_i. \tag{16}$$

Temperatures $T < 1$ result in an effective increase of the gradient term.

4.2.2 Initialization of the MCMC

In theory, the convergence of the MCMC does not depend on the random initial set of points from which the MCMC starts. However, there are two motivations for using a non-random initial set of points. First, the MCMC is not infinitely long, and so there is no guarantee of convergence starting from any initial distribution. Second, from one batch to another, the neural network weights and p_θ both only change slightly, so a good initialization for batch i usually also works for batch $i+1$. Persistent contrastive divergence (PCD) [25] is used to initialize the MCMC at each batch. In PCD, the MCMC in batch i starts not from a new random set of points, but rather from the final set of points obtained from the MCMC in batch $i-1$. This procedure can be tuned by adding a restart fraction f, meaning that at every batch, a fraction f of the points sampled for the previous batch are discarded and replaced by new, random points, while the remaining points from the previous batch are reused [26]. The restart fraction f is a hyperparameter of the MCMC and was set to 5%.

4.2.3 Tuning of the MCMC

The MCMC is the crucial component of the NAE. A suboptimal MCMC would not accurately sample the NAE probability p_θ, resulting in the NAE not being able to learn the training data probability distribution p_{data}. However, tuning the hyperparameters of the MCMC—the coefficients of Eq. (14), the drift λ and the noise σ—is a complex task since the p_θ distribution is unknown.

The MCMC hyperparameters λ and σ were optimized to maximize the AUC for classifying SVJs versus $\mathrm{t\bar{t}}$ jets, using the loss function introduced in Eq. (17), under two constraints. First, at the maximum AUC, the negative and positive samples histograms of the input features should match, as discussed in Section 4.3 and Fig. 4. Second, the Wasserstein distance between the positive and negative samples should be minimal, as discussed in Section 5.1. If the MCMC is correctly set and the network is sufficiently trained, then the network learns $p_\theta = p_{\text{data}}$, so the AUC is greater than 0.5, assuming the distribution of the signal is not also equal to p_{data}, in which case no discrimination is possible. In this case, the distribution of the negative sample is representative of the p_{data} distribution.

Some MCMC settings with a considerably larger gradient coefficient λ compared to the optimum were found to reach larger AUCs; however, these failed to achieve a good match between the distributions of negative and positive samples. Such MCMC configurations, though appealing if only the AUC is considered, result in an inadequate NAE that has not learned $p_\theta = p_{\text{data}}$, but rather has a large AUC only because of the specific location of the signal hypotheses. This choice would therefore bias the performance of the NAE towards the signal hypotheses used for tuning, spoiling the unsupervised nature of the setup.

4.3 Failure modes of the NAE

After tuning of the MCMC, the evolution of the positive and negative energies of an NAE with the loss function as defined in Eq. (13) is presented in Fig. 3. It presents two failure modes: negative loss values, indicating the negative energy is higher than positive energy; and divergence of the two energies.

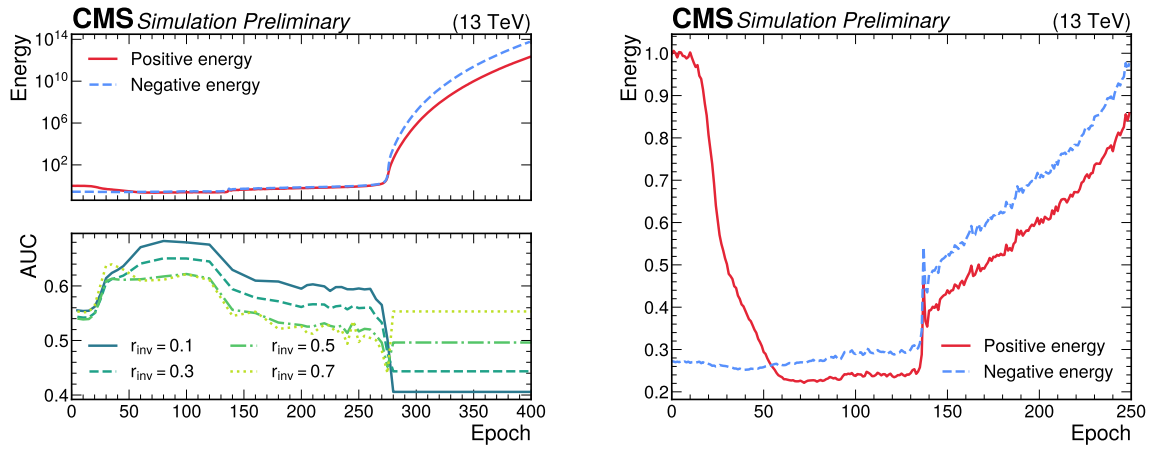

Figure 3: Left: NAE training showing the divergence of the loss function, in terms of positive and negative energy (upper panel), and the AUC for several signal hypotheses with fixed mediator mass, $m_\Phi = 2000\,\text{GeV}$, but varying invisible fraction r_{inv} (lower panel). Right: from the same training, the positive and negative energies are shown before the divergence, illustrating their differences.

Negative values of the energy difference occur when the network did not learn the background probability distribution, and the training data probability density, p_{data}, is higher than the NAE probability density, p_θ, in regions with low reconstruction error. This is a severe failure mode, illustrating that training the NAE to minimize the energy difference can incentivize the NAE to learn $p_\theta \neq p_{\text{data}}$.

The divergence of the negative energy was also reported in previous work [4, 23]. It was found to happen for multiple reasons. First, there are limitations when sampling negative samples from a finite phase space. When the region of phase space with the lowest reconstruction error lies beyond the sampling boundaries, negative samples accumulate at the boundaries, unable to reach the lowest reconstruction error phase space. This results in a failure mode in which the network is incentivized to increase the reconstruction error of positive and negative samples, effectively leading to a divergence of both energy terms and the loss itself, as shown in Fig. 3. The distributions of a selected input feature τ_3 for the positive, negative, and signal (only used for plotting purposes, not training) samples before and after the start of the divergence are provided in Fig. 4, showing that the finite sampling phase space is the cause of the divergence. However, sampling from an infinite phase space is not possible in practice. Moreover, large regions of the input feature space may be valid for computing the NAE reconstruction error but still physically meaningless, such as negative values of jet substructure variables. Therefore, sampling from an infinite, unbounded phase space is not desirable.

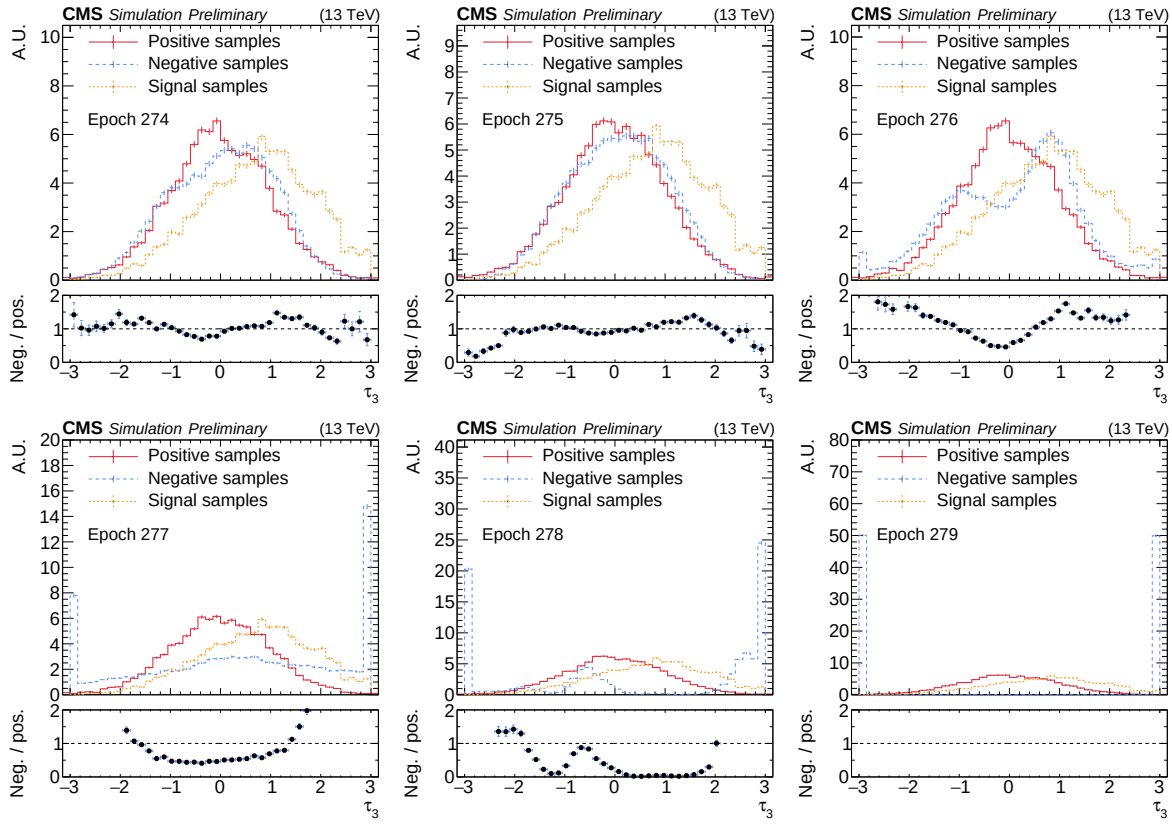

Figure 4: Histograms of the input feature τ_3 for positive, negative, and signal samples, before (epochs 274) and after (epochs 275–279) the start of the divergence of the NAE loss.

Second, both negative values of the energy difference and divergence of the energies can result from incorrect MCMC tuning. For instance, large values of λ result in MCMC steps beyond the size of the sampling phase space, such that the negative samples are pushed to the boundaries of that space. Therefore, the negative samples are not representative of the network probability

distribution, leading to negative values of the energy difference and divergences, from the mechanism explained previously. A simple way of mitigating these issues would be to modify the NAE loss to:

$$\mathcal{L} = \log\cosh(E_+ - E_-). \tag{17}$$

This removes the incentive to favor negative energy difference, and leads to a more stable training procedure. The evolution of the energies and the AUC score for a training with this loss function are shown in Fig. 5.

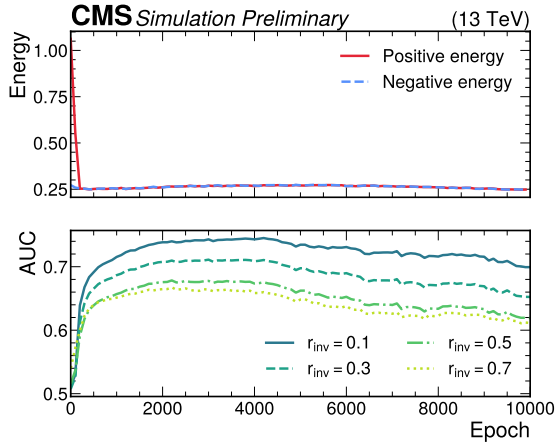

Figure 5: Positive and negative energies (upper panel) and AUC for several signal hypotheses with fixed mediator mass, $m_\Phi = 2000\,\text{GeV}$, but varying invisible fraction r_{inv} (lower panel) during training with the loss function in Eq. (17).

The behavior can be qualitatively understood as follows: there is a first stage in which the energy difference is minimized, giving the largest improvement in AUC; once the difference between positive and negative energies is approximately zero, there is a second stage in which the network must fine-tune its knowledge of p_θ, further increasing the AUC score up to a plateau. There is, however, a third stage in which a new form of mode collapse is observed: the energy difference is still roughly constant and zero, and the positive energy is further minimized, but the AUC score begins to decrease. Crucially, it is not possible to avoid this degradation by relying solely on the positive and negative energies. On the other hand, making use of the AUC score as a stopping condition for the training would be in contradiction with the unsupervised nature of the NAE.

This form of collapse is understood as a form of overtraining of the network. While keeping the reconstruction error low, the network eventually achieves low reconstruction in a region $\mathcal{E}$ that contains, but is larger than, the support of the training data $\mathcal{B}$, while keeping $\mathbb{E}_{x\sim\mathcal{E}}[E(x)]$ comparable to $\mathbb{E}_{x\sim\mathcal{B}}[E(x)]$. As the overlap of $\mathcal{E}$ with the support of the signal $\mathcal{S}$ becomes larger, the AUC score decreases. This behavior is shown schematically in Fig. 6. A solution to this problem, leading to the development of the WNAE algorithm, is discussed in the following section.

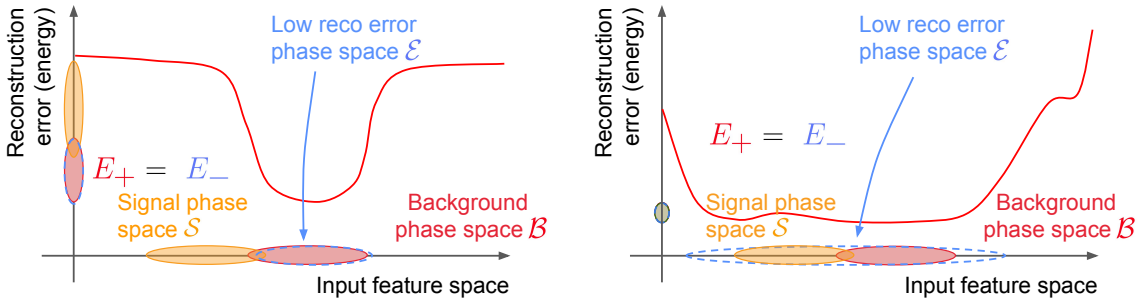

Figure 6: Schematic representation of the mode collapse when using the loss function described in Eq. (17). Left: before the mode collapse, $\mathcal{E}$ and $\mathcal{B}$ overlap, while $\mathcal{E}$ and $\mathcal{S}$ do not. Right: after the mode collapse, $\mathcal{E}$ expands and can partially include $\mathcal{S}$, reducing the difference in AE reconstruction error and thus lowering the AUC score. E_+ and E_- respectively denote the positive and negative energies. In both cases, the difference between the positive and the negative energies is zero.

5 The Wasserstein normalized autoencoder

5.1 The Wasserstein distance as a measure of outlier reconstruction

The failure mode described at the end of Section 4 is a consequence of the fact that the energy difference is unable to distinguish the two cases shown in Fig. 6. In fact, there is no stopping condition based solely on combinations of positive and negative energies that was found to be a good indicator of the start of this mode collapse. In other words, without explicitly monitoring the AUC during training, there is no clear stopping condition; choosing a stopping condition based on the AUC, however, violates the principle of unsupervised learning. In order to overcome this problem, a new metric was introduced, using the Wasserstein distance W (also known as the earth mover's distance) between the probability distribution of the training data p_{data} and the probability distribution learned by the network p_θ, in analogy to what was applied successfully to generative adversarial networks [27]:

$$\mathrm{W}(p_{\text{data}}, p_\theta) = \inf_{\gamma \in \Pi(p_{\text{data}}, p_\theta)} \mathbb{E}_{(x,x')\sim\gamma} \left[\|x - x'\| \right], \tag{18}$$

where $\Pi(p_{\text{data}}, p_\theta)$ denotes the set of all joint distributions whose marginal distributions are p_{data} and p_θ, respectively. The Wasserstein distance is found to be a good indicator of the start of the mode collapse, as can be seen in Fig. 7: the Wasserstein distance increases significantly when this collapse occurs and the AUC simultaneously starts to decrease. The kink in the Wasserstein distance, quantified as the point in which its derivative changes by more than a given threshold, can then be identified as a stopping condition for the training of the NAE. It is worth stressing that the Wasserstein distance, being calculated between the positive and negative samples, is completely agnostic to the signal, and yet it is found to be closely correlated with the performance of the NAE as an outlier detector. With this prescription, the NAE was able to successfully learn the $\mathrm{t\bar{t}}$ distribution and effectively detect the signal, as shown in Fig. 7.

5.2 Obtaining a differentiable Wasserstein distance

Having shown that the Wasserstein distance is a good indicator of the performance of the NAE as an outlier detection algorithm, it is natural to use it directly as a loss function. This is common practice in energy-based models for generative applications. Such a generative model g is

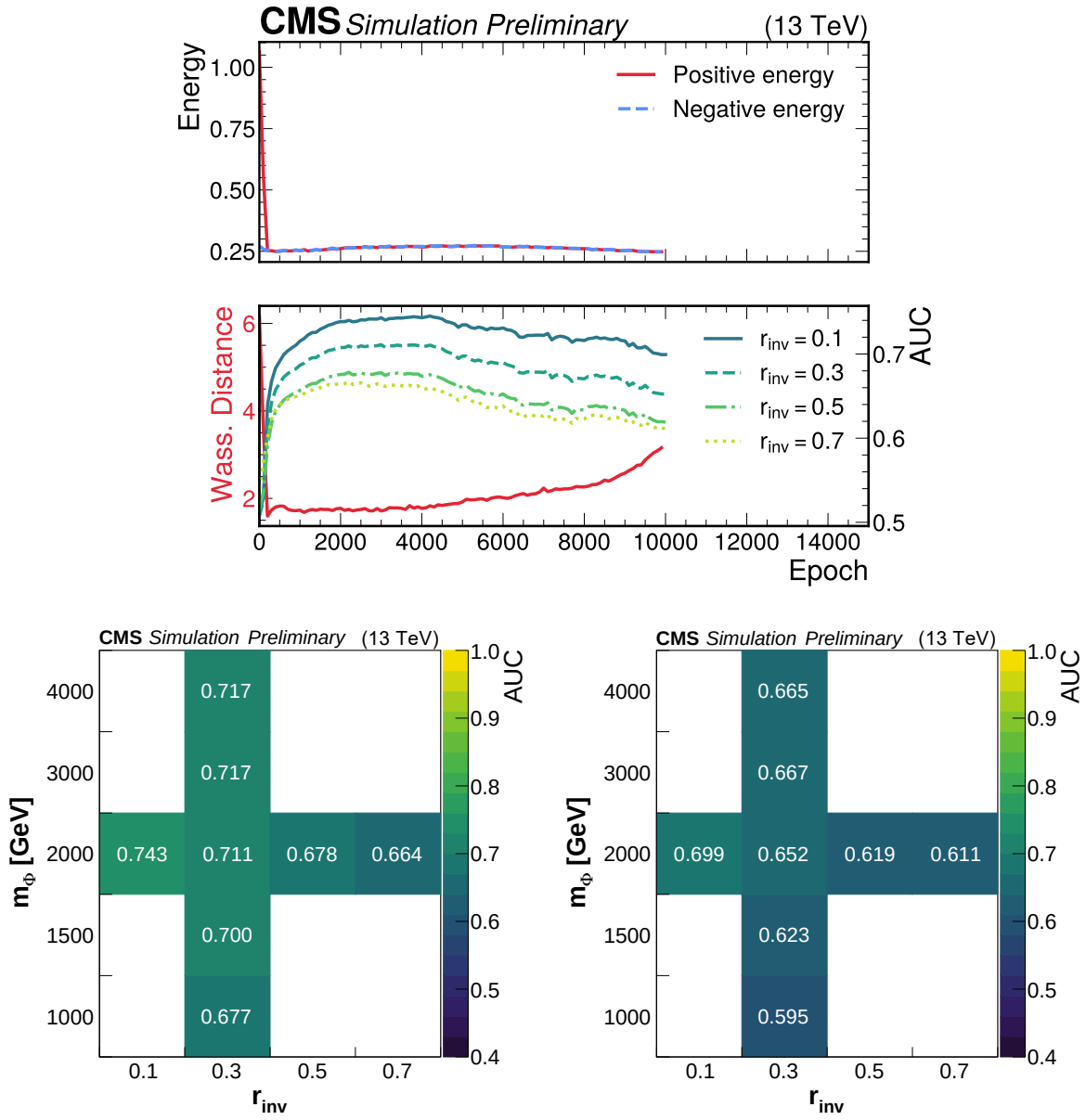

Figure 7: Upper: the positive and negative energy (upper panel), and the Wasserstein distance between the positive and negative samples with AUC scores for several signal hypotheses with fixed mediator mass, $m_\Phi = 2000\,\text{GeV}$, but varying invisible fraction r_{inv} (lower panel), during the training of an NAE with the loss function in Eq. (17). Lower left: the AUC scores for an NAE trained on the $t\bar{t}$ background and tested against a grid of possible SVJ signals, before the increase of the Wasserstein distance (at epoch 3000). Lower right: the AUC scores for the same NAE after the increase in Wasserstein distance (at epoch 10000).

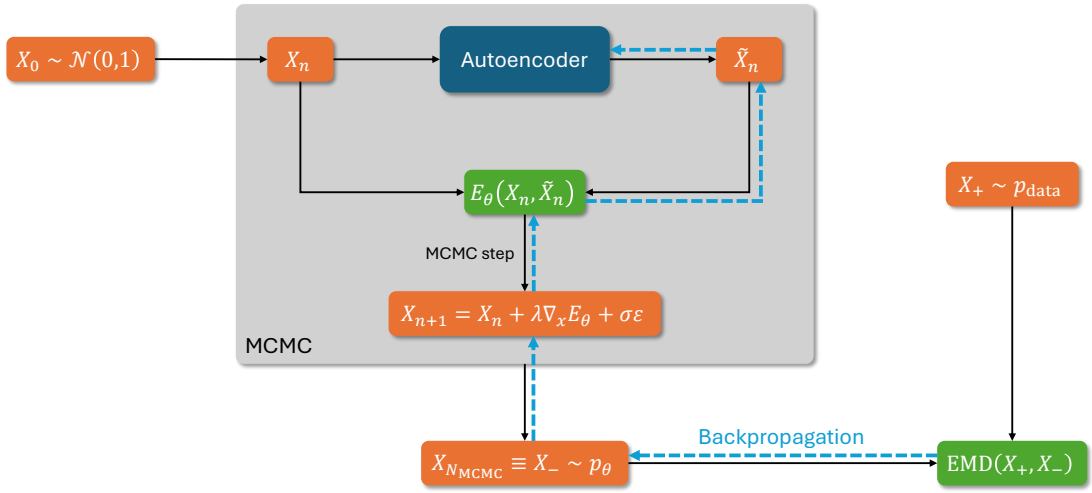

Figure 8: Flowchart of the Wasserstein normalized autoencoder training. The positive examples are passed through the autoencoder, and the negative examples are generated via MCMC. The Wasserstein distance is calculated between the positive and negative examples, and the gradients are backpropagated through the entire MCMC chain.

trained to minimize the Wasserstein distance W between the training examples x it is trying to replicate, and the samples $g(z)$ it generates from random noise z:

$$\mathcal{L}_{\text{generative}} = \mathrm{W}(p_{\text{data}}, p_g) = \inf_{\gamma \in \Pi(p_{\text{data}}, p_z)} \mathbb{E}_{(x,z)\sim\gamma} \left[\|x - g(z)\| \right]. \tag{19}$$

A differentiable implementation of the Wasserstein distance metric is provided in the POT package [28]. In the case of energy-based generative models, the Wasserstein distance depends explicitly on the network weights via $g(z)$. However, this is not the case for an AE with the loss function defined in Eq. (18), since neither the positive nor the negative examples are passed through the network explicitly. The dependency is in the generation of the negative examples, which is done via the MCMC described in Section 4. As is evident from Eq. (16), the dependency on the network weights is in the gradient of the energy function with respect to the input features. As such, in order to use the Wasserstein distance as a loss function, the dependency of each step of the MCMC on the weights of the network must be retained:

$$\nabla_\theta x_{i+1,\theta} = \nabla_\theta x_{i,\theta} - \lambda \nabla_\theta \nabla_x E_\theta(x_{i,\theta}). \tag{20}$$

In the PyTorch package [29], this is accomplished by explicitly enabling the `create_graph` flag when calling the `autograd` method on the AE energy at each MCMC step. This ensures that the computational graph, on which PyTorch relies to perform the backpropagation step, includes the dependency of the negative examples on the network weights throughout the MCMC. With this prescription, the Wasserstein distance can be used as a loss function for the AE, and the resulting model is called the Wasserstein normalized autoencoder (WNAE). The flowchart for training the WNAE is shown in Fig. 8.

5.3 Training the Wasserstein normalized autoencoder

The model discussed in the following is the same fully connected AE as in the previous sections, consisting of 5 layers with 10, 10, 6, 10, and 10 nodes. The network is trained on a sample of $t\bar{t}$ background events. The features have been preprocessed with the same quantile normalization as in the AE and NAE examples. The loss function is now the Wasserstein distance as written in Eq. (18).

Special care has to be taken in the choice of the learning rate, as the training is found to qualitatively proceed in two phases: first, a coarse adjustment in which the Wasserstein distance decreases sharply while the model adapts the negative samples to resemble physical examples, and second, a fine-tuning phase in which the model learns the specifics of the training sample. If the learning rate is too small, the training time increases significantly, while if it is too large, the training becomes unstable in the second phase. In this example, an initial learning rate of 2×10^{-4} is chosen. The `torch.optim.lr_scheduler.ReduceLROnPlateau` method is used to reduce the learning rate by a factor of 0.8 when the Wasserstein distance stops decreasing sharply. This was found to be a good compromise between training time and stability.

The computational bottleneck for training the WNAE is the calculation of the Wasserstein distance between the positive and negative samples, which takes approximately 90% of the computation time for each training batch. The second most impactful operation is the backpropagation through the MCMC chain, which takes approximately 7% of the computation time. The time needed for the calculation of the Wasserstein distance heavily depends on the batch size, which was set to 4096 to allow for a robust estimation of the distance between positive and negative samples. Lowering the batch size reduces the computation time at the expense of decreasing the stability of the loss function. Depending on the batch size, the training of the WNAE can thus be significantly slower than that of the NAE. However, the Wasserstein distance is found to be a much more reliable and stable stopping condition, as will be shown in the following section.

6 Performance of the Wasserstein normalized autoencoder

6.1 Identification of semivisible jets

The WNAE is found to be significantly more stable in training, not showing any of the forms of divergence described previously. It is able to achieve anomaly detection performance on par with or better than the NAE, while using the minimum of the validation loss, the Wasserstein distance evaluated on the validation set, as a clear stopping condition. Figure 9 shows the evolution of the Wasserstein distance and the AUC score for several SVJ signal hypotheses. The minimum of the Wasserstein distance corresponds to the maximum discrimination power. Figure 10 shows the AUC score evaluated for a grid of SVJ signals, showing good discrimination power across the parameter space. The WNAE is thus found to be a more reliable, stable, and effective tool for anomaly detection in the context of the SVJ search. In addition, it must be noted that this algorithm does not rely on the existence of a bottleneck with lower dimension than the input feature space. The same performance can be obtained with all hidden layers being of higher dimension than the input feature space.

The fact that the WNAE behaves as desired can be seen in Figs. 11 and 12, where the distributions of the input features for the positive, negative, and signal samples are shown at the start and end of the training. Starting from an essentially multivariate distribution, induced by the random weight initialization of the network, the WNAE is able to learn the distribution of

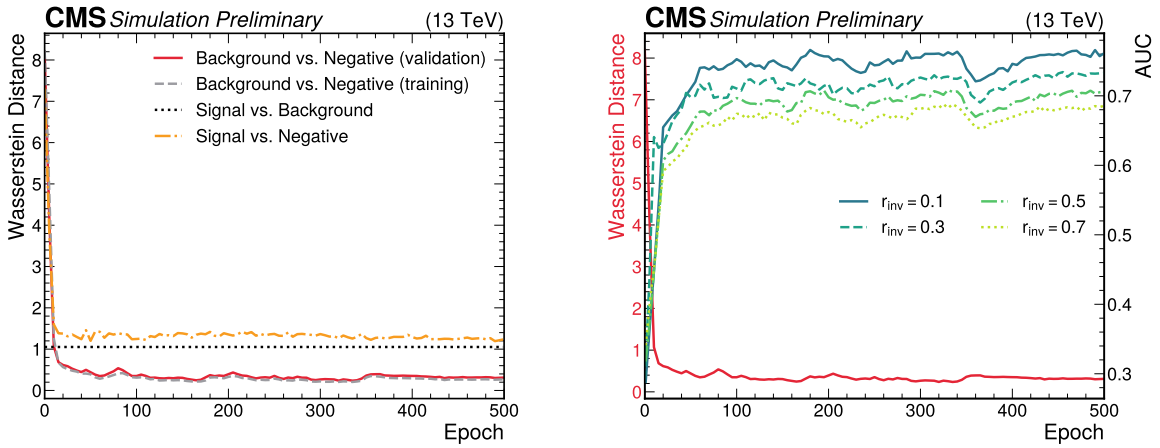

Figure 9: Left: the Wasserstein distance between pairs of the positive, negative, and signal samples during the WNAE training. Right: the AUC scores from the same WNAE for several signal hypotheses with fixed mediator mass, $m_\Phi = 2000\,\text{GeV}$, but varying invisible fraction r_{inv}.

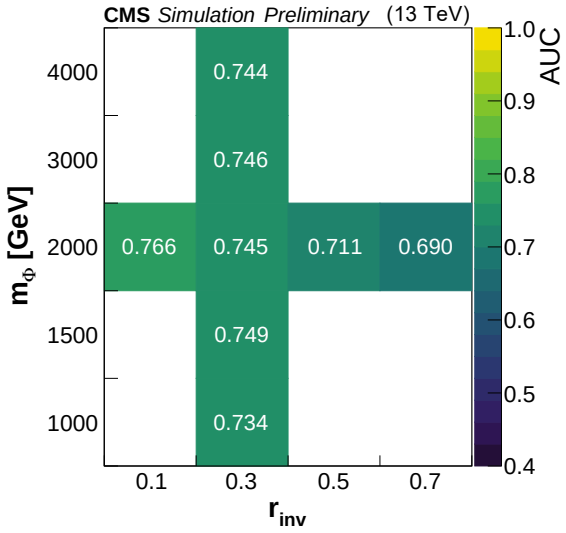

Figure 10: The AUC scores for a WNAE trained on the $\text{t}\bar{\text{t}}$ background and tested on a grid of possible SVJ signal models.

the training data, as evidenced by the negative and positive sample distributions eventually matching.

6.2 Complexity bias in the Wasserstein normalized autoencoder

A common weakness of plain AEs is the so-called complexity bias, in which the AE tends to be able to effectively discriminate only between examples whose distribution in input space is more structured than that of the training data. This can be demonstrated by inverting the background and signal samples and observing that, in this case, an AE trained on SM jets can tag SVJs as anomalous, but the reverse is not true. Because the WNAE learns the probability distribution of the training data in input space, this issue is mitigated, as shown in Fig. 13. The high AUC at the start of the training is caused by the Gaussian initialization of the network weights. The AUC then decreases as the network learns a distribution closer to the physical

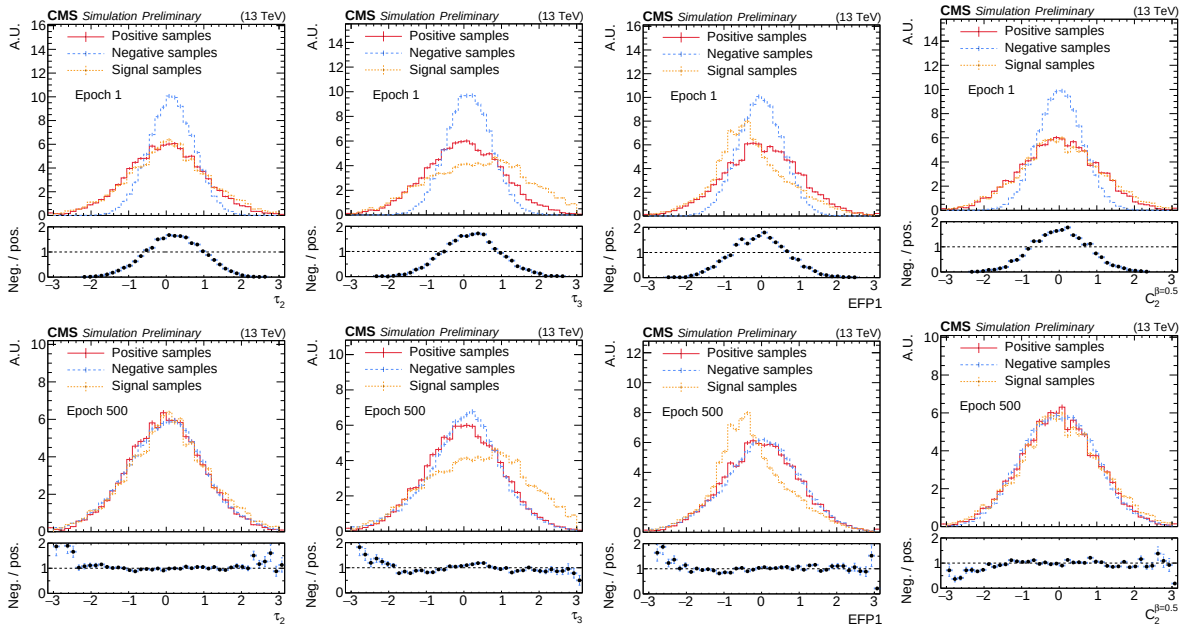

Figure 11: The distributions of half of the input variables, τ_2, τ_3, EFP1, and $C_2^{(0.5)}$, for the positive, negative, and signal samples, at the start (upper) and at the end (lower) of the WNAE training.

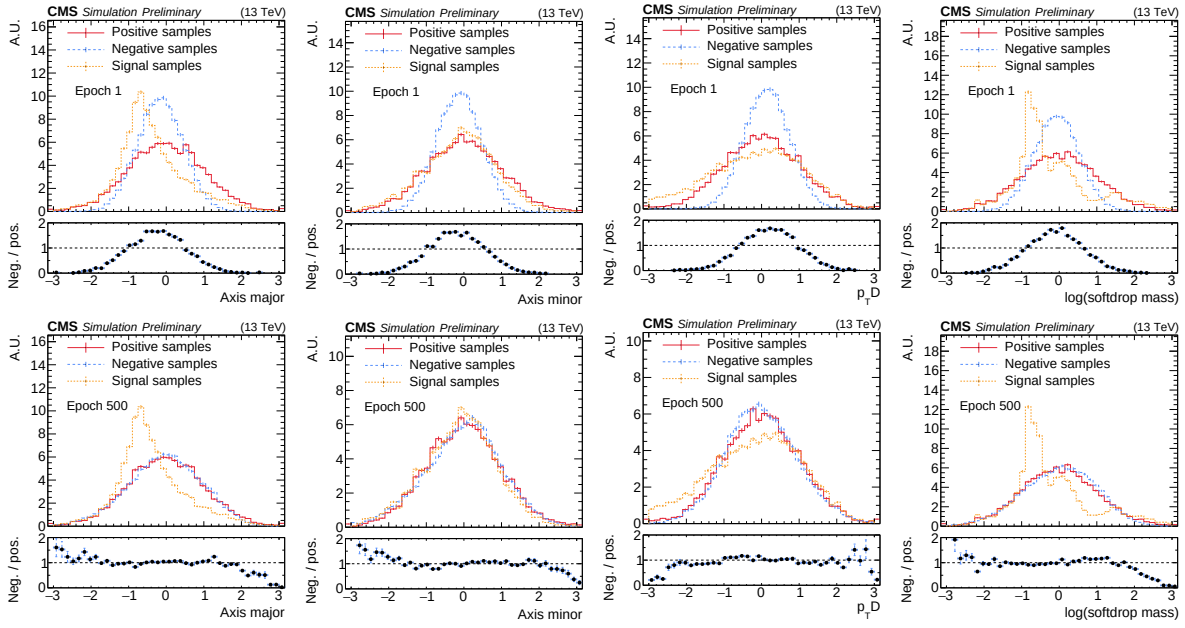

Figure 12: The distributions of the other half of the input variables, axis major, axis minor, p_T^D, and softdrop mass, for the positive, negative, and signal samples, at the start (upper) and at the end (lower) of the WNAE training.

input data, before increasing again as the network starts to more precisely learn the distribution of SVJ examples and thus becomes able to tag SM jets as anomalous.

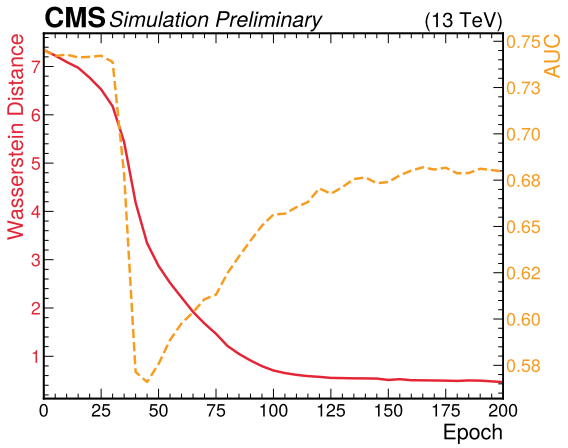

Figure 13: The Wasserstein distance between the positive and negative samples and the AUC score during the training of a WNAE on an SVJ signal ($m_\Phi = 2000\,\text{GeV}$, $r_{\text{inv}} = 0.3$), with the $\text{t}\bar{\text{t}}$ background used for testing.

7 Summary

Autoencoder-based anomaly detection relies on learning a reconstruction error such that phase space regions with low probability density have high reconstruction error and can be identified as anomalous. However, standard autoencoders are prone to learn to reconstruct outliers because they are free to minimize the reconstruction error outside the training phase space. The normalized autoencoder paradigm promotes the autoencoder reconstruction error to an energy function in the framework of energy-based models, in order to define a normalized probabilistic model. This is achieved by minimizing the negative log-likelihood of the training data, given the energy-based model probability. In practice, this construction presents a number of failure modes, such as divergence of the loss function and phase space degeneracy, leading phase space regions distinct from the training data to have low reconstruction error. The Wasserstein normalized autoencoder, an improvement over normalized autoencoders, is introduced to solve the aforementioned failure modes. This is achieved by using the Wasserstein distance to quantify the difference between the probability distribution of the training data and the Boltzmann distribution of the energy function of the model. Using simulated samples from the CMS experiment, the classification of out-of-distribution examples by the Wasserstein normalized autoencoder is shown to be on par with or better than that of the normalized autoencoder. Furthermore, the Wasserstein distance is found to be a robust metric to define a stopping condition for the training in a fully signal-agnostic fashion.

References

[1] T. Heimel, G. Kasieczka, T. Plehn, and J. M. Thompson, "QCD or what?", *SciPost Phys.* **6** (2019) 030, doi:10.21468/SciPostPhys.6.3.030, arXiv:1808.08979.

[2] M. Farina, Y. Nakai, and D. Shih, "Searching for new physics with deep autoencoders", *Phys. Rev. D* **101** (2020) 075021, doi:10.1103/PhysRevD.101.075021, arXiv:1808.08992.

[3] T. Finke et al., "Autoencoders for unsupervised anomaly detection in high energy physics", *JHEP* **06** (2021) 161, doi:10.1007/JHEP06(2021)161, arXiv:2104.09051.

[4] S. Yoon, Y.-K. Noh, and F. Park, "Autoencoding under normalization constraints", *Proc. 38th Int. Conf. Mach. Learn.* (2021) 12087, arXiv:2105.05735.

[5] T. Cohen, M. Lisanti, and H. K. Lou, "Semivisible jets: Dark matter undercover at the LHC", *Phys. Rev. Lett.* **115** (2015) 171804, doi:10.1103/PhysRevLett.115.171804, arXiv:1503.00009.

[6] CMS Collaboration, "The CMS experiment at the CERN LHC", *JINST* **3** (2008) S08004, doi:10.1088/1748-0221/3/08/S08004.

[7] M. J. Strassler and K. M. Zurek, "Echoes of a hidden valley at hadron colliders", *Phys. Lett. B* **651** (2007) 374, doi:10.1016/j.physletb.2007.06.055, arXiv:hep-ph/0604261.

[8] CMS Collaboration, "Search for resonant production of strongly coupled dark matter in proton-proton collisions at 13 TeV", *JHEP* **06** (2022) 156, doi:10.1007/JHEP06(2022)156, arXiv:2112.11125.

[9] T. Cohen, M. Lisanti, H. K. Lou, and S. Mishra-Sharma, "LHC searches for dark sector showers", *JHEP* **11** (2017) 196, doi:10.1007/JHEP11(2017)196, arXiv:1707.05326.

[10] M. A. Kramer, "Autoassociative neural networks", *Comput. Chem. Eng.* **16** (1992) 313, doi:10.1016/0098-1354(92)80051-A.

[11] 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", *JHEP* **07** (2014) 079, doi:10.1007/JHEP07(2014)079, arXiv:1405.0301.

[12] T. Sjöstrand et al., "An introduction to PYTHIA 8.2", *Comput. Phys. Commun.* **191** (2015) 159, doi:10.1016/j.cpc.2015.01.024, arXiv:1410.3012.

[13] GEANT4 Collaboration, "GEANT4—a simulation toolkit", *Nucl. Instrum. Meth. A* **506** (2003) 250, doi:10.1016/S0168-9002(03)01368-8.

[14] M. Cacciari, G. P. Salam, and G. Soyez, "The anti-k_{T} jet clustering algorithm", *JHEP* **04** (2008) 063, doi:10.1088/1126-6708/2008/04/063, arXiv:0802.1189.

[15] M. Cacciari, G. P. Salam, and G. Soyez, "FastJet user manual", *Eur. Phys. J. C* **72** (2012) 1896, doi:10.1140/epjc/s10052-012-1896-2, arXiv:1111.6097.

[16] CMS Collaboration, "Performance of quark/gluon discrimination in 8 TeV pp data", CMS Physics Analysis Summary CMS-PAS-JME-13-002, 2013.

[17] P. T. Komiske, E. M. Metodiev, and J. Thaler, "Energy flow polynomials: A complete linear basis for jet substructure", *JHEP* **04** (2018) 013, doi:10.1007/JHEP04(2018)013, arXiv:1712.07124.

[18] A. J. Larkoski, G. P. Salam, and J. Thaler, "Energy correlation functions for jet substructure", *JHEP* **06** (2013) 108, doi:10.1007/JHEP06(2013)108, arXiv:1305.0007.

[19] A. J. Larkoski, S. Marzani, G. Soyez, and J. Thaler, "Soft drop", *JHEP* **05** (2014) 146, doi:10.1007/JHEP05(2014)146, arXiv:1402.2657.

[20] J. Thaler and K. Van Tilburg, "Identifying boosted objects with N-subjettiness", *JHEP* **03** (2011) 015, doi:10.1007/JHEP03(2011)015, arXiv:1011.2268.

[21] F. Pedregosa et al., "Scikit-learn: Machine learning in Python", *J. Mach. Learn. Res.* **12** (2011) 2825, arXiv:1201.0490.

[22] F. Canelli et al., "Autoencoders for semivisible jet detection", *JHEP* **02** (2022) 074, doi:10.1007/JHEP02(2022)074, arXiv:2112.02864.

[23] B. M. Dillon et al., "A normalized autoencoder for LHC triggers", *SciPost Phys. Core* **6** (2023) 074, doi:10.21468/SciPostPhysCore.6.4.074, arXiv:2206.14225.

[24] E. T. Jaynes, "Information Theory and Statistical Mechanics", *Phys. Rev.* **106** (1957) 620, doi:10.1103/PhysRev.106.620.

[25] T. Tieleman, "Training restricted Boltzmann machines using approximations to the likelihood gradient", *Proc. 25th Int. Conf. Mach. Learn.* (2008) 1064, doi:10.1145/1390156.1390290.

[26] Y. Du and I. Mordatch, "Implicit generation and modeling with energy-based models", *Proc. 33rd Int. Conf. Neural Inf. Process. Syst.* **32** (2019) 324, arXiv:1903.08689.

[27] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN", *Proc. 34th Int. Conf. Mach. Learn.* **70** (2017) 214, arXiv:1701.07875.

[28] R. Flamary et al., "POT: Python optimal transport", *J. Mach. Learn. Res.* **22** (2021) 1.

[29] A. Paszke et al., "PyTorch: An imperative style, high-performance deep learning library", *Proc. 33rd Int. Conf. Neural Inf. Process. Syst.* **32** (2019) 721, arXiv:1912.01703.