Alleviating Adversarial Attacks on Variational Autoencoders with MCMC

Alleviating Adversarial Attacks on Variational Autoencoders with MCMC

Anna Kuzina
Vrije Universiteit Amsterdam
a.kuzina@vu.nl

Max Welling
Universiteit van Amsterdam
m.welling@uva.nl

Jakub M. Tomczak
Vrije Universiteit Amsterdam
j.m.tomczak@vu.nl

Abstract

Variational autoencoders (VAEs) are latent variable models that can generate complex objects and provide meaningful latent representations. Moreover, they could be further used in downstream tasks such as classification. As previous work has shown, one can easily fool VAEs to produce unexpected latent representations and reconstructions for a visually slightly modified input. Here, we examine several objective functions for adversarial attack construction proposed previously and present a solution to alleviate the effect of these attacks. Our method utilizes the Markov Chain Monte Carlo (MCMC) technique in the inference step that we motivate with a theoretical analysis. Thus, we do not incorporate any extra costs during training, and the performance on non-attacked inputs is not decreased. We validate our approach on a variety of datasets (MNIST, Fashion MNIST, Color MNIST, CelebA) and VAE configurations (β-VAE, NVAE, β-TCVAE), and show that our approach consistently improves the model robustness to adversarial attacks.

1 Introduction

Variational Autoencoders (VAEs) [27, 34] are latent variable models parameterized by deep neural networks and trained with variational inference. Recently, it has been shown that VAEs with hierarchical structures of latent variables [33], coupled with skip-connections [30, 37], can generate high-quality images [15, 39]. An interesting trait of VAEs is that they allow learning meaningful latent space that could be further used in downstream tasks [7, 22]. These successes of VAEs motivate us to explore the *robustness* of the resulting latent representations to better understand the capabilities and potential vulnerabilities of VAEs. Here, we focus on *adversarial attacks* on VAEs to verify robustness of latent representations that is especially important in such applications as anomaly detection [1, 30] or data compression [5, 20].

The main questions about adversarial attacks for VAEs are mainly focused on how they could be formulated and alleviated. In [18], it is proposed to minimize the KL-divergence between an adversarial input and a target input to learn an adversarial attack for the vanilla VAE. Further, in [28], it is shown that a similar strategy can be used to attack hierarchical VAEs. To counteract the adversarial attacks, the authors of [42] suggest using a modified VAE objective, namely, β-TCVAE, that increases VAE robustness, especially when coupled with a hierarchical structure. It was shown in [10] that β-VAEs tend to be more robust to adversarial attacks in terms of the r-metric proposed therein. The authors of [6] presented that the adversarial robustness can be achieved by constraining the Lipschitz constant of the encoder and the decoder. [12, 13] introduced modifications in the VAE framework that allow for better robustness against the adversarial attacks on downstream classification tasks. In our work, we consider β-VAE, β-TCVAE and a hierarchical VAE, and outline a defence strategy that improves robustness to attacks on the encoder and the downstream classification task. The proposed method is applied during inference and, therefore, can be combined with other known techniques to get more robust latent representations.

An adversarial attack on a VAE is usually formulated as an additive perturbation ε of the real data point $\mathbf{x}^r$ so that the resulting point is perceived by a model as if it is a totally different image (either during reconstruction or in the downstream classification task) [18]. In Figure 1, we depict an example of an attack on the encoder. The reference point $\mathbf{x}^r$ and the adversarial point $\mathbf{x}^a$ are almost indistinguishable, but they are encoded into different regions in the latent space. As a result, their reconstructions also differ significantly.

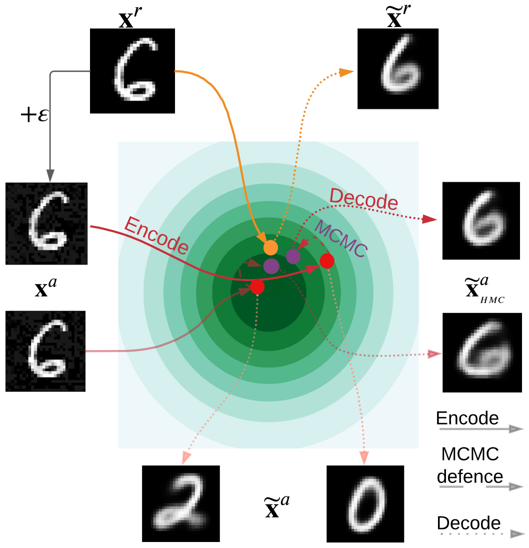

$\mathbf{x}^r$
$\widetilde{\mathbf{x}}^r$
$+\varepsilon$
Encode
Decode
MCMC
$\mathbf{x}^a$
$\widetilde{\mathbf{x}}^a_{HMC}$
$\widetilde{\mathbf{x}}^a$
Encode
MCMC defence
Decode

Figure 1: An example of an unsupervised encoder attack on VAE with 2D latent space and the proposed defence. Given a single reference point $\mathbf{x}^r$ we learn additive perturbation ε, s.t. perturbed input $\mathbf{x}^a$ has the most different latent code and, therefore, the reconstruction $\widetilde{\mathbf{x}}^a$. We observe that a single reference point can be mapped to extremely different regions of the latent space but using MCMC we are able to move them closer to the initial position so that the reconstruction $\widetilde{\mathbf{x}}^a_{HMC}$ is similar to the initial one $\widetilde{\mathbf{x}}^r$.

In this paper, we propose the method motivated by the following hypothesis: *An adversarial attack maps the input to a latent region with a lower probability mass assigned by the true posterior (proportional to the conditional likelihood times the marginal over latents) and, eventually, we obtain incorrect reconstructions*. Therefore, a potential manner to alleviate the effect of an attack may rely on running a Markov chain to move the latent representation back to a more probable latent region. Such a defence is reasonable because we do not modify the training procedure or the model itself, we only insert a correction procedure. As a result, we propose to counteract adversarial attacks by enhancing the variational inference with Markov Chain Monte Carlo (MCMC) sampling. The illustrative example depicted in the Figure 1 shows that the latent code of the adversarial input (red circle) moves closer to the latent code of the reference point (orange circle) after applying the MCMC (purple circle).

The contribution of this work is the following:

- We propose to use an MCMC technique during inference to correct adversarial attacks on VAEs.
- We show theoretically that the application of an MCMC technique could indeed help to counteract adversarial attacks (Theorem 1).
- We indicate empirically that the previously proposed strategies to counteract adversarial attacks do not generalize well across various datasets.
- We show empirically that the proposed approach (i.e., a VAE with an MCMC during inference) outperforms all baselines by a significant margin.

2 Background

2.1 Variational Autoencoders

Let us consider a vector of observable random variables, $\mathbf{x} \in \mathcal{X}^D$ (e.g., $\mathcal{X} = \mathbb{R}$) sampled from the empirical distribution $p_e(\mathbf{x})$, and vectors of latent variables $\mathbf{z}_k \in \mathbb{R}^{M_k}$, $k = 1, 2, \ldots, K$, where M_k is the dimensionality of each latent vector. First, we focus on a model with $K = 1$ and the joint distribution $p_\theta(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})$. The marginal likelihood is then equal to $p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}, \mathbf{z})\mathrm{d}\mathbf{z}$. VAEs exploit variational inference [25] with a family of variational posteriors $\{q_\phi(\mathbf{z}|\mathbf{x})\}$, also referred to as encoders, that results in a tractable objective function, i.e., the Evidence Lower BOund (ELBO): $\mathcal{L}(\phi, \theta) = \mathbb{E}_{p_e(\mathbf{x})}\left(\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \ln p_\theta(\mathbf{x}|\mathbf{z}) - D_{\mathrm{KL}}\left[q_\phi(\mathbf{z}|\mathbf{x}) \| p(\mathbf{z})\right]\right)$.

β-VAE [21] uses a modified objective by weighting the D_{KL} term by $\beta > 0$. In the case of $K > 1$, we consider a hierarchical latent structure with the generative model of the following form: $p_\theta(\mathbf{x}, \mathbf{z}_1, \ldots, \mathbf{z}_K) = p_\theta(\mathbf{x}|\mathbf{z}_1, \ldots, \mathbf{z}_K) \prod_{k=1}^{K} p_\theta(\mathbf{z}_k|\mathbf{z}_{>k})$. There are various possible formulations

Table 1: Different types of attacks on the VAE. We denote $g_\theta(z)$ the deterministic mapping induced by decoder $p_\theta(x|z)$ and as $p_\psi(y|z)$ classification model in the latent space (downstream task).
[*] Only used during VAE training

	REFERENCE	$f(x)$	$\Delta[A,B]$	$\|\cdot\|_p$	TYPE
Latent Space Attack	[6, 18, 42]	$q_\phi(\cdot|x)$	$D_{\text{KL}}[A\|B]$	2	Supervised
Unsupervised Encoder Attack	[28]	$q_\phi(\cdot|x)$	SKL $[A\|B]$	2	Unsupervised
Targeted Output Attack	[18]	$g_\theta(\tilde{z}), \tilde{z} \sim q_\phi(\cdot|x)$	$\|A-B\|_2^2$	2	Supervised
Maximum Damage Attack	[6, 10]	$g_\theta(\tilde{z}), \tilde{z} \sim q_\phi(\cdot|x)$	$\|A-B\|_2^2$	2	Unsupervised
Projected Gradient Descent Attack[*]	[13]	$q_\phi(\cdot|x)$	$\mathcal{WD}[A,B]$	inf	Unsupervised
Adversarial Accuracy	[12, 13]	$p_\psi(y|\tilde{z}), \tilde{z} \sim q_\phi(\cdot|x)$	CROSS ENTROPY	inf	Unsupervised

of the family of variational posteriors. However, here we follow the proposition of [37] with the autoregressive inference model, namely, $q_\phi(\mathbf{z}_1,\ldots,\mathbf{z}_K|\mathbf{x}) = q_\phi(\mathbf{z}_K|\mathbf{x})\prod_{k=1}^{K-1} q_{\theta,\phi}(\mathbf{z}_k|\mathbf{z}_{>k},\mathbf{x})$. This formulation was used, among others, in NVAE [39]. Because of the top-down structure, it allows sharing data-dependent information between the inference model and the generative model.

2.2 Adversarial attacks

An *adversarial attack* is a slightly deformed data point that results in an undesired or unpredictable performance of a model [19]. In this work, we focus on the attacks that are constructed as an additive perturbation of the real data point $\mathbf{x}^r$ (which we will refer to as *reference*), namely:

$$\mathbf{x}^a = \mathbf{x}^r + \varepsilon, \text{where} \tag{1}$$
$$\|\varepsilon\|_p \leq \delta, \tag{2}$$

where δ is the radius of the attack. The additive perturbation ε is chosen in such a way that the attacked point does not differ from the reference point too much in a sense of a given similarity measure. It is a solution to an optimization problem solved by the attacker. The optimization problem could be formulated in various manners by optimizing different objectives and by having specific constraints and/or assumptions.

Attack construction Let $f(x)$ be part of the model available to the attacker. In the case of VAEs, this, for example, may be an encoder network or an encoder with the downstream classifier in the latent space. The attacker uses a similarity measure Δ to learn an additive perturbation to a reference point. We consider two settings: *unsupervised* and *supervised*. In the former case, the perturbation is supposed to incur the largest possible change in f:

$$\varepsilon = \arg\max_{\|\varepsilon\|_p\leq\delta} \Delta\left[f(\mathbf{x}^r+\varepsilon), f(\mathbf{x}^r)\right]. \tag{3}$$

The latter setting requires a *target point* $\mathbf{x}^t$. The perturbation attempts to match the output for the target and the attacked points, namely:

$$\varepsilon = \arg\min_{\|\varepsilon\|_p\leq\delta} \Delta\left[f(\mathbf{x}^r+\varepsilon), f(\mathbf{x}^t)\right]. \tag{4}$$

There are different ways to select f and Δ in the literature. Furthermore, different L_p-norms can be used to restrict the radius of the attack. Table 1 summarizes recent work on the topic.

In this work, we focus on attacking the encoder and the downstream classification task using unsupervised adversarial attacks. In the former, we maximize the symmetric KL-divergence to get the point with the most unexpected latent code and, therefore, the reconstruction. In the latter, the latent code is passed to a classifier. The attack is trained to change the class of the point by maximizing the cross-entropy loss. However, the method we propose in Section 3 is not limited to these setups since it is agnostic to how the attack was trained.

Robustness measures To measure the robustness of the VAE as well as the success of the proposed defence strategy, we focus on two metrics: MSSSIM and Adversarial accuracy.

For latent space attacks we follow [28] in using Multi-Scale Structural Similarity Index Measure (MSSSIM) [41]. We calculate $\mathrm{MSSSIM}[\widetilde{\mathbf{x}}^r, \widetilde{\mathbf{x}}^a]$, i.e., the similarity between reconstructions of $\mathbf{x}^r$ and the corresponding $\mathbf{x}^a$. We do not report the similarity between a reference and the corresponding adversarial input, since this value is the same for all the considered models (for a given attack radius). A successful adversarial attack would have a small value of $\mathrm{MSSSIM}[\widetilde{\mathbf{x}}^r, \widetilde{\mathbf{x}}^a]$.

For the attacks on the downstream classifier, we follow [12, 13] and calculate the adversarial accuracy. For a given trained VAE model, we first train a linear classifier using latent codes as features. Afterwards, the attack is trained to fool the classifier. Adversarial accuracy is the proportion of points for which the attack was unsuccessful. Namely, when the predicted class of the reference and corresponding adversarial point are the same.

3 Preventing adversarial attacks with MCMC

Assumptions and the hypothesis We consider a scenario in which an attacker has access to the VAE encoder and, where relevant, to the downstream classification model. Above, we presented in detail how an attack could be performed. We assume that the defender cannot modify these components, but it is possible to add elements that the attacker has no access to.

We hypothesize that *the adversarial attacks result in incorrect reconstructions because the latent representation of the adversarial input lands in a region with a lower probability mass assigned by the true posterior.* This motivates us to use a method which brings the latents "back" to a highly probable region as a potential defence strategy. Since the adversarial attack destroys the input irreversibly, at the first sight it seems impossible to reconstruct the latent representation of the reference point. We aim at showing that this is possible to some degree theoretically and empirically (Appendix C.1).

The proposed solution In order to steer the latents towards high probability regions, we propose to utilize an MCMC method during inference time. Since we are not allowed to modify the VAE or its learning procedure, this is a reasonable procedure from the defender's perspective. Another positive outcome of such an approach is that in a case of no attack, the latents given by the MCMC sampling will be closer to the mean of the posterior, thus, the reconstruction should be sharper or, at least, not worse. Note that the variational posterior approximates the true posterior from which the MCMC procedure samples. We propose to sample from $q^{(t)}(\mathbf{z}|\mathbf{x}) = \int Q^{(t)}(\mathbf{z}|\mathbf{z}_0) q_\phi(\mathbf{z}_0|\mathbf{x}) d\mathbf{z}_0$, where $Q^{(t)}(\mathbf{z}|\mathbf{z}_0)$ is a transition kernel of MCMC with t steps and the target distribution $\pi(\mathbf{z}) = p_\theta(\mathbf{z}|\mathbf{x}) \propto p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})$. Alternatively, we can use optimization to find the mode of the posterior, however, MCMC can add extra benefits such as exploration of the typical set and randomness (see Appendix B.3 for details).

To further analyze the proposed approach, we start with showing the following lemma:

Lemma 1 *Consider true posterior distributions of the latent code* $\mathbf{z}$ *for a data point* $\mathbf{x}$ *and its corrupted version* $\mathbf{x}^a$. *Assume also that* $\ln p_\theta(\mathbf{z}|\mathbf{x})$ *is twice differentiable over* $\mathbf{x}$ *with continuous derivatives at the neighbourhood around* $\mathbf{x} = \mathbf{x}^r$. *Then the KL-divergence between these two posteriors could be expressed using the small o notation of the radius of the attack, namely:*

$$D_{\mathrm{KL}}\left[p_\theta(\mathbf{z}|\mathbf{x}^r) \| p_\theta(\mathbf{z}|\mathbf{x}^a)\right] = o(\|\varepsilon\|). \tag{5}$$

Proof See Appendix A.

According to Lemma 1, the difference (in the sense of the Kullback-Leibler divergence) between the true posteriors for a data point $\mathbf{x}$ and its corrupted version $\mathbf{x}^a$ decreases faster than the norm of the attack radius. However, it is important to note that this result is only valid for asymptotically small attack radius.

Next, the crucial step to show is whether we can quantify somehow the difference between the distribution over latents for $\mathbf{x}^a$ after running MCMC with t steps, $q^{(t)}(\mathbf{z}|\mathbf{x}^a)$, and the variational distribution for $\mathbf{x}^r$, $q_\phi(\mathbf{z}|\mathbf{x}^r)$. We provide an important upper-bound for the Total Variation distance (TV)[1] between $q^{(t)}(\mathbf{z}|\mathbf{x}^a)$ and $q_\phi(\mathbf{z}|\mathbf{x}^r)$ in the following lemma:

[1]The Total Variation fulfills the triangle inequality and it is a proper distance measure.

Lemma 2 *The Total Variation distance (*TV*) between the variational posterior with MCMC for a given corrupted point* $\mathbf{x}^a$*,* $q^{(t)}(\mathbf{z}|\mathbf{x}^a)$*, and the variational posterior for a given data point* $\mathbf{x}^r$*,* $q_\phi(\mathbf{z}|\mathbf{x}^r)$*, can be upper bounded by the sum of the following three components:*

$$\begin{aligned}\mathrm{TV}\left[q^{(t)}(\mathbf{z}|\mathbf{x}^a), q_\phi(\mathbf{z}|\mathbf{x}^r)\right] \leq\ & \mathrm{TV}\left[q^{(t)}(\mathbf{z}|\mathbf{x}^a), p_\theta(\mathbf{z}|\mathbf{x}^a)\right] \\ &+ \sqrt{\tfrac{1}{2}D_{\mathrm{KL}}\left[p_\theta(\mathbf{z}|\mathbf{x}^r)\|p_\theta(\mathbf{z}|\mathbf{x}^a)\right]} \\ &+ \sqrt{\tfrac{1}{2}D_{\mathrm{KL}}\left[q_\phi(\mathbf{z}|\mathbf{x}^r)\|p_\theta(\mathbf{z}|\mathbf{x}^r)\right]}. \end{aligned} \tag{6}$$

Proof See Appendix A.

The difference expressed by $\mathrm{TV}\left[q^{(t)}(\mathbf{z}|\mathbf{x}^a), q_\phi(\mathbf{z}|\mathbf{x}^r)\right]$ is thus upper-bounded by the following three components:

- The TV between $q^{(t)}(\mathbf{z}|\mathbf{x}^a)$ and the real posterior for the corrupted image, $p_\theta(\mathbf{z}|\mathbf{x}^a)$. Theoretically, if $t \to \infty$, $q^{(\infty)}(\mathbf{z}|\mathbf{x}^a) = p_\theta(\mathbf{z}|\mathbf{x}^a)$ and, hence, $\mathrm{TV}\left[q^{(\infty)}(\mathbf{z}|\mathbf{x}^a), p_\theta(\mathbf{z}|\mathbf{x}^a)\right] = 0$.
- The second component is the square root of the KL-divergence between the real posteriors for the image and its corrupted counterpart. Lemma 1 gives us information about this quantity.
- The last element, the square root of $D_{\mathrm{KL}}\left[q_\phi(\mathbf{z}|\mathbf{x}^r)\|p_\theta(\mathbf{z}|\mathbf{x}^r)\right]$, quantifies the *approximation gap* [16], i.e., the difference between the best variational posterior from a chosen family, and the true posterior. This quantity has no direct connection with adversarial attacks. However, as we can see, using a rich family of variational posteriors can help us to obtain a tighter upper-bound. In other words, taking flexible variational posteriors allows to counteract attacks. This finding is in line with the papers that propose to use hierarchical VAEs as the means for preventing adversarial attacks [42].

Eventually, by applying Lemma 1 to Lemma 2, we obtain the following result:

Theorem 1 *The upper bound on the total variation distance between samples from MCMC for a given corrupted point* $\mathbf{x}^a$*,* $q^{(t)}(\mathbf{z}|\mathbf{x}^a)$*, and the variational posterior for the given real point* $\mathbf{x}^r$*,* $q_\phi(\mathbf{z}|\mathbf{x}^r)$*, is the following:*

$$\begin{aligned}\mathrm{TV}\left[q^{(t)}(\mathbf{z}|\mathbf{x}^a), q_\phi(\mathbf{z}|\mathbf{x}^r)\right] \leq\ & \mathrm{TV}\left[q^{(t)}(\mathbf{z}|\mathbf{x}^a), p_\theta(\mathbf{z}|\mathbf{x}^a)\right] \\ &+ \sqrt{\tfrac{1}{2}D_{\mathrm{KL}}\left[q_\phi(\mathbf{z}|\mathbf{x}^r)\|p_\theta(\mathbf{z}|\mathbf{x}^r)\right]} \\ &+ o(\sqrt{\|\varepsilon\|}). \end{aligned} \tag{7}$$

Proof See Appendix A.

As discussed already, the first component gets smaller with more steps of the MCMC. The second component could be treated as a *bias* of the family of variational posteriors. Finally, there is the last element that corresponds to a constant error that is unavoidable. However, this term decays faster than the square root of the attack radius for the asymptotically small attack radius.

Specific implementation of the proposed approach In this paper, we use a specific MCMC method, namely, the Hamiltonian Monte Carlo (HMC) [8, 17]. Once the VAE is trained, the attacker calculates an adversarial point $\mathbf{x}^a$ using the encoder of the VAE. After the attack, the latent representation of $\mathbf{x}^a$ is calculated, $\mathbf{z}^a$, and used as the initialization of the HMC.

In the HMC, the target (unnormalized) distribution is $p(\mathbf{x}^a|\mathbf{z})p(\mathbf{z})$. The Hamiltonian is then the energy of the joint distribution of $\mathbf{z}$ and the auxilary variable $\mathbf{p}$, that is:

$$H(\mathbf{z}, \mathbf{p}) = U(\mathbf{z}) + K(\mathbf{p}), \tag{8}$$

$$U(\mathbf{z}) = -\ln p_\theta(\mathbf{x}^a|\mathbf{z}) - \ln p(\mathbf{z}), \tag{9}$$

$$K(\mathbf{p}) = -\tfrac{1}{2}\mathbf{p}^T\mathbf{p}. \tag{10}$$

When applying the proposed defence to hierarchical models, we update all the latent variables simultaneously. That is, we have $\mathbf{z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_K\}$ and $U(\mathbf{z}) = -\ln p_\theta(\mathbf{x}^a|\mathbf{z}) - \sum_{k=1}^{K} \ln p_\theta(\mathbf{z}_k|\mathbf{z}_{k+1})$.

Eventually, the resulting latents from the HMC are decoded. The steps of the whole process are presented below:

1. (*Defender*) Train a VAE: $q_\phi(\mathbf{z}|\mathbf{x})$, $p(\mathbf{z})$, $p_\theta(\mathbf{x}|\mathbf{z})$.
2. (*Attacker*) For given $\mathbf{x}^r$, calculate the attack $\mathbf{x}^a$ using the criterion equation 3 or equation 4.
3. (*Defender*) Initialize the latent code $\mathbf{z} := \mathbf{z}_0$, where $\mathbf{z}_0 \sim q_\phi(\mathbf{z}|\mathbf{x}^a)$. Then, run T steps of HMC (Algorithm 1) with the step size η and L *leapfrog* steps.

The resulting latent code $\mathbf{z}$ can be passed to the decoder to get a reconstruction or to the downstream classification task.

Algorithm 1 One Step of HMC.

```
Input: $\mathbf{z}, \eta, L$
  $\mathbf{p} \sim \mathcal{N}(0, I)$.    ▷ Sample the auxiliary variable
  $\mathbf{z}^{(0)} := \mathbf{z}, \mathbf{p}^{(0)} := \mathbf{p}$.
  for $l = 1 \dots, L$ do    ▷ Make $L$ steps of leapfrog.
      $\mathbf{p}^{(l)} = \mathbf{p}^{(l-1)} - \frac{\eta}{2}\nabla_{\mathbf{z}} U(\mathbf{z}^{(l)})$.
      $\mathbf{z}^{(l)} = \mathbf{z}^{(l)} + \eta \nabla_{\mathbf{p}} K(\mathbf{p}^{(l)})$.
      $\mathbf{p}^{(l)} = \mathbf{p}^{(l)} - \frac{\eta}{2}\nabla_{\mathbf{z}} U(\mathbf{z}^{(l)})$.
  end for
                    ▷ Accept new point with prob. $\alpha$.
  $\alpha = \min\left(1, \exp\left(-H(\mathbf{z}^{(L)}, \mathbf{p}^{(L)}) + H(\mathbf{z}^{(0)}, \mathbf{p}^{(0)})\right)\right)$
  $\mathbf{z} = \begin{cases} \mathbf{z}^{(L)} & \text{with probability } \alpha, \\ \mathbf{z}^{(0)} & \text{otherwise.} \end{cases}$
Return: $\mathbf{z}$
```

4 Experiments

4.1 Posterior ratio

We motivate our method by the hypothesis that the adversarial attack "shifts" a latent code to the region of a lower posterior density, while our approach moves it back to a high posterior probability region. In Section 3 we theoretically justify our hypothesis, while here we provide an additional empirical evidence. The true posterior $p(\mathbf{z}|\mathbf{x}^r)$ is not available due to the cumbersome marginal distribution $p(\mathbf{x}^r)$, however, we can calculate the ratio of posteriors because the marginal will cancel out. In our case, we are interested in calculating the posterior ratio between the reference and adversarial latent codes ($\mathbf{z}_1 = \mathbf{z}^r$, $\mathbf{z}_2 = \mathbf{z}^a$) as the baseline, and the posterior ratio between the reference and adversarial code after applying the HMC ($\mathbf{z}_1 = \mathbf{z}^r$, $\mathbf{z}_2 = \mathbf{z}^a_{\text{HMC}}$). The lower the posterior ratio, the better. For practical reasons, we use the logarithm of the posterior ratio since the logarithm does not change the monotonicity and turns products to sums:

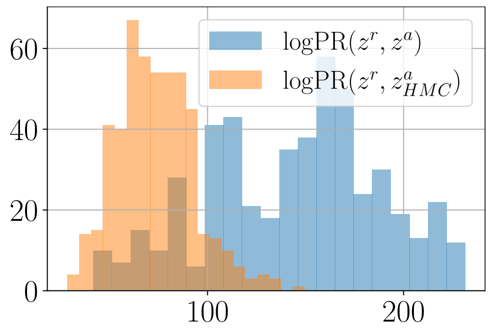


Figure 2: Histograms of the log posterior ratios before HMC (blue) and after HMC (orange) evaluated on the MNIST dataset.

$$\log \text{PR}(\mathbf{z}_1, \mathbf{z}_2) = \log p_\theta(\mathbf{x}^r|\mathbf{z}_1) + \log p(\mathbf{z}_1) - \log p_\theta(\mathbf{x}^r|\mathbf{z}_2) - \log p(\mathbf{z}_2). \tag{11}$$

In Figure 2 we show a plot with two histograms: one with the posterior ratio between the reference and adversarial latent codes ($\mathbf{z}_1 = \mathbf{z}^r$, $\mathbf{z}_2 = \mathbf{z}^a$) in blue, and the second histogram of the posterior ratio between the reference and adversarial code after applying the HMC ($\mathbf{z}_1 = \mathbf{z}^r$, $\mathbf{z}_2 = \mathbf{z}^a_{\text{HMC}}$) in orange. We observe that the histogram has moved to the left after applying the HMC. This indicates that posterior of the adversarial (in the denominator) is increasing when the HMC is used. This is precisely the effect we hoped for and this result provides an empirical evidence in favor of our hypothesis. For more details see C.1.

4.2 VAE, β-VAE and β-TCVAE

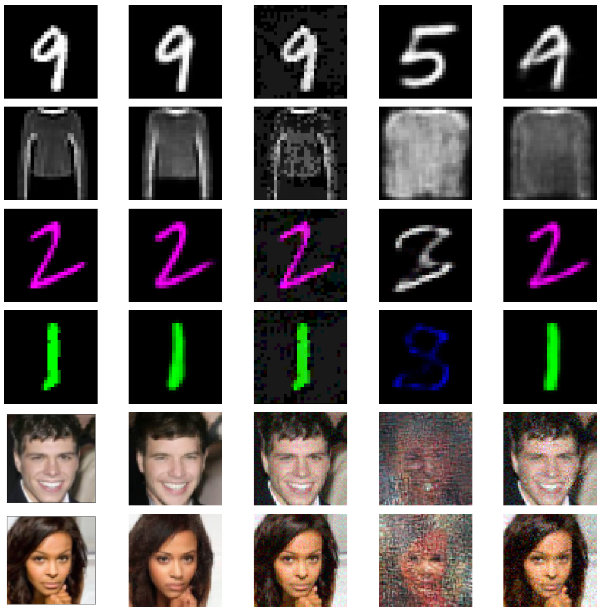
All implementation details and hyperparameters are included in the Appendix D and code repository [2].

[2] `https://github.com/AKuzina/defend_vae_mcmc`

Datasets VAEs are trained on the MNIST, Fashion MNIST [44] and Color MNIST datasets. Following [13], we construct the Color MNIST dataset from MNIST by artificially coloring each image with seven colors (all corners of RGB cube except for black).

Models We train vanilla fully convolutional VAEs, as well as β-VAE [21] and β-TCVAE [14, 26]. Both β-VAE and β-TCVAE modify the ELBO objective to encourage disentanglement. β-VAE weighs the KL-term in the ELBO with $\beta > 0$. It is said that the larger values of β encourage disentangling of latent representations [14] and improve the model robustness as observed by [10]. β-TCVAE puts a higher weight on the total correlation (TC) term of the ELBO. Penalization of the total correlation was shown to increase the robustness of VAE adversarial attacks [42].

In Appendix D.1 we provide details of the architecture, optimization, and results on the test dataset for VAE trained with different values of β. We note that the optimal value in terms of the negative log-likelihood (NLL) is always $\beta = 1$. Larger values of β are supposed to improve robustness in exchange for the reconstruction quality. When evaluating the robustness of β-VAE and β-TCVAE, we train models with $\beta \in \{2, 5, 10\}$. Then, we select the value of β that provides the most robust model in terms of the used metric. Next, we apply our defence strategy to this model to observe the potential performance improvement.



(a) $\mathbf{x}^r$ (b) $\widetilde{\mathbf{x}}^r$ (c) $\mathbf{x}^a$ (d) $\widetilde{\mathbf{x}}^a$ (e) $\widetilde{\mathbf{x}}^{\mathrm{a}}_{\mathrm{HMC}}$

Figure 3: Examples of (a) reference points, (b) reconstructions of the reference points, (c) adversarial points, (d) reconstructions of the adversarial points, (e) reconstructions of the adversarial points after the proposed defence (HMC). All the adversarial examples are unsupervised attacks on the encoder. Last two rows contain examples for the NVAE model discussed in Section 4.3

Table 2: Results for unsupervised attack with radius 0.1 and 0.2 on MNIST and Fashion MNIST datasets. We attack the encoder (left) and the downstream classification task (right).
† Our implementation.

		MSSSIM$[\widetilde{\mathbf{x}}^r, \widetilde{\mathbf{x}}^a]\uparrow$		ADVERSARIAL ACCURACY $\uparrow$			MSE $\downarrow$
	$\|\varepsilon\|$	0.1	0.2	0.0	0.1	0.2	
MNIST	VAE	0.70 (0.02)	0.36 (0.03)	**0.90** (0.04)	0.08 (0.04)	0.05 (0.03)	578.7
	VAE + HMC	**0.88** (0.01)	**0.76** (0.02)	0.76 (0.01)	**0.25** (0.03)	**0.19** (0.03)	**478.1**
	β-VAE	0.75 (0.01)	0.50 (0.03)	0.90 (0.05)	0.11 (0.04)	0.01 (0.01)	824.2
	β-TCVAE†	0.70 (0.02)	0.46 (0.03)	0.86 (0.05)	0.05 (0.03)	0.03 (0.02)	828.4
FASHION MNIST	VAE	0.59 (0.03)	0.47 (0.03)	0.78 (0.06)	0.00 (0.01)	0.01 (0.01)	814.2
	VAE + HMC	**0.66** (0.03)	**0.54** (0.03)	0.56 (0.01)	**0.14** (0.02)	**0.13** (0.02)	**764.2**
	β-VAE	0.52 (0.03)	0.41 (0.03)	0.80 (0.05)	0.00 (0.01)	0.00 (0.01)	1021.1
	β-TCVAE†	0.52 (0.03)	0.42 (0.03)	**0.84** (0.05)	0.00 (0.01)	0.02 (0.02)	980.4

Attacks on the Encoder In the first setup, we assume that the attacker has access to the encoder of the model $q_\phi(z|x)$ [6, 18, 42]. We use the projected gradient descent (PGD) with 50 steps to maximize the symmetric KL-divergence in the unsupervised setting (equation 3). We train 10 adversarial attacks (with different random initialization) for each of 50 reference points. See Appendix D.2 for the details. We report similarity between the reconstruction of the adversarial and reference point as a measure of the robustness (see Section 2.2).

Table 3: Results for unsupervised attack with radius 0.1 and 0.2 on ColorMNIST dataset. We attack the encoder (left) and the downstream classification task (right).
† Our implementation.
* Values reported in [12], VAE implementation and evaluation protocol may differ.

	MSSSIM$[\tilde{\mathbf{x}}^r, \tilde{\mathbf{x}}^a]$ ↑		ADVERSARIAL ACCURACY ↑							
			DIGIT			COLOR			MSE↓	FID↓
$\|\|\varepsilon\|\|$	0.1	0.2	0.0	0.1	0.2	0.0	0.1	0.2		
VAE	0.36 (0.03)	0.19 (0.02)	**1.00 (0.00)**	0.04 (0.03)	0.02 (0.02)	1.00 (0.00)	0.06 (0.03)	0.06 (0.03)	**261**	**2.1**
VAE + HMC	**0.96 (0.01)**	**0.90 (0.01)**	0.42 (0.01)	0.16 (0.02)	0.11 (0.02)	1.00 (0.00)	0.68 (0.03)	0.62 (0.03)	**206**	**2.1**
β-VAE	0.75 (0.01)	0.5 (0.03)	0.88 (0.04)	0.08 (0.04)	0.05 (0.03)	1.00 (0.00)	0.21 (0.06)	0.18 (0.05)	366	2.4
β-TCVAE†	0.35 (0.02)	0.23 (0.02)	0.94 (0.04)	0.08 (0.04)	0.05 (0.03)	1.00 (0.00)	0.06 (0.03)	0.05 (0.02)	366	3.0
SE$_{0.1}$*	N/A	N/A	0.94 (N/A)	0.89 (N/A)	0.02 (N/A)	1.00 (N/A)	1.00 (N/A)	0.22 (N/A)	1372	13.0
SE$_{0.2}$*	N/A	N/A	0.95 (N/A)	0.92 (N/A)	**0.87 (N/A)**	1.00 (N/A)	1.00 (N/A)	**1.00 (N/A)**	1375	11.7
AVAE*	N/A	N/A	0.97 (N/A)	0.88 (N/A)	0.55 (N/A)	1.00 (N/A)	1.00 (N/A)	0.88 (N/A)	1372	15.5
SE$_{0.1}$-AVAE*	N/A	N/A	0.97 (N/A)	**0.94 (N/A)**	0.25 (N/A)	1.00 (N/A)	1.00 (N/A)	0.60 (N/A)	1373	13.9
SE$_{0.2}$-AVAE*	N/A	N/A	0.98 (N/A)	**0.94 (N/A)**	0.80 (N/A)	1.00 (N/A)	1.00 (N/A)	0.83 (N/A)	1374	13.9
AVAE-SS*	N/A	N/A	0.94 (N/A)	0.73 (N/A)	0.21 (N/A)	1.00 (N/A)	1.00 (N/A)	0.57 (N/A)	1379	12.4

Attacks on the downstream task In this setup, we examine how the proposed approach can aleviate the effect of the attack on the downstream tasks in the latent space. For this purpose, we follow the procedure from [12, 13]. Once the VAE is trained, we learn a linear classifier using the mean mappings as features. For the MNIST and Fashion MNIST datasets, we have the 10-class classification problem (digits in the former and pieces of clothing in the latter case). For the ColorMNIST dataset, we consider two classification tasks: the digit classification (10 classes), and the color classification (7 classes). We construct the attack to fool the classifier. See Appendix D.2 for the details.

Results In Tables 2 and 3, we compare our method (VAE + HMC) with the vanilla VAE with other methods in the literature. We report more results and the extended comparison in the Appendix C.2 where we show that our method combined with β-VAE and β-TCVAE leads to the increased robustness. For MNIST and Fashion MNIST (Table 2), we observe that the vanilla VAE with the HMC is more robust than β-VAE and β-TCVAE. The latter model was shown to be more robust to the latent space attack [42]. Still, in our experiments (on different datasets), we could not observe the consistent improvement over the vanilla VAE, when using it with a single level of latent variables.

In Table 3 we report the result on the ColorMNIST dataset. Here, we additionally compare the adversarial accuracy for our method with the Smooth Encoders (SE) and Autoencoding Variational Autoencoder (AVAE) methods [12, 13]. We notice that these methods provide higher adversarial accuracy. However, we have also observed a large discrepancy in terms of the MSE and FID scores of the model itself compared to our experiments, which we suspect might be a result of a mistake in [12, 13]. [3]

Lastly, we would like to highlight that our defence strategy can be also combined with all the above VAE modifications. One advantage of our approach is that it does not require changing the training procedure of a VAE and, as a result, it does not decrease the quality of the generated images. Moreover, we can apply the same procedure to reconstruct the non-corrupted points and it will improve the reconstruction error. This result can be seen in the Tables 2 and 3 and it goes in line with the results of the [36], where MCMC was used to improve the VAE performance.

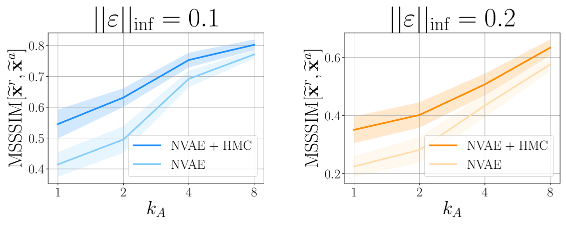
4.3 Hierarchical VAE: NVAE

Model and datasets In this section, we explore the robustness of the deep hierarchical VAE (NVAE) [39], a specific implementation of a hierarchical VAE that works well for high-dimensional data. We attack models trained on MNIST and CelebA [29] datasets. We use the weights of the pre-trained model provided in the official NVAE implementation[4].

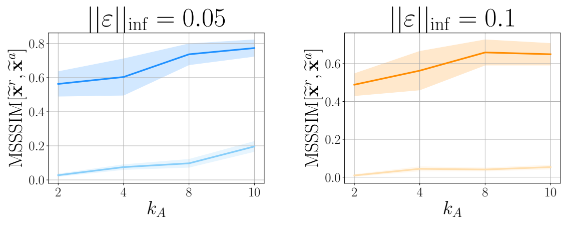
Attacks construction Following [28] we construct adversarial attacks on the hierarchical VAE by considering higher-level latent variables. That being said, we use latent variables

[3]We have gotten in touch with the authors, who are kindly working with us to address this issue

[4]The code and model weights were taken from `https://github.com/NVlabs/NVAE`

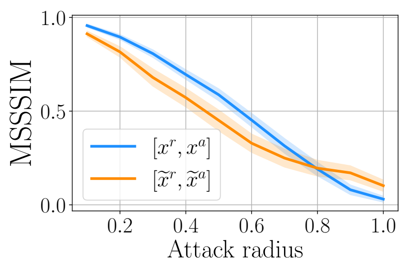


(a) The reconstruction similarity for MNIST

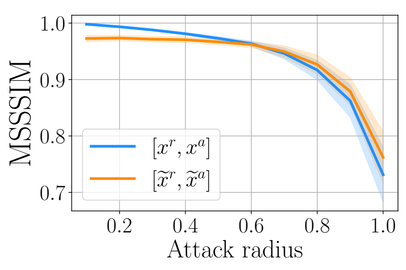


(b) The reconstruction similarity for CelebA

Figure 4: The robustness improvement for the hierarchical model (NVAE) on (a) MNIST and (b) CelebA for two different attack radii. Higher values correspond to more robust representations.



(a) An attacker does not know the defence strategy



(b) An attacker knows the defence strategy

Figure 5: Robustness to adversarial attack (with HMC defence). We report similarity of the reference and adversarial point (blue) and their reconstructions (orange).

$\{\mathbf{z}_{L-k_A}, \mathbf{z}_{L-k_A+1}, \ldots, \mathbf{z}_L\}$ when constructing an attack (equation 3). Otherwise, we follow the same procedure we used for VAEs with a single level of latent variables. We assume that the attacker has access to the model's encoder and uses the symmetric KL-divergence as the objective. The radius of an attack is measured with the L_{inf} norm. For optimization, we use the projected gradient descent with the number of iterations limited to 50 per point. Further details are reported in the Appendix D.2.

Results In Figure 4, we present reconstruction similarity of the reference and adversarial points for both datasets. We observe that the proposed method consistently improves the robustness of the model to the adversarial attack. This result is in line with our theoretical considerations where a flexible class of variational posteriors could help to counteract adversarial attacks and, eventually, deacrease the bias of the class of models measured in terms of the KL-divergence. Additionally, applying the MCMC can further help us to counteract the attack. In Figure 3 (the two bottom rows), we show how an adversarial point (c) is reconstructed without any defence (d) and with the proposed defence (e). In the depicted samples, we have used the top 10 latent variables ($k_A = 10$) to construct the attack with a radius of 0.05.

4.4 Ablation Study: What if the attacker knows the defence strategy?

In our experiments, we rely on the assumption that the attacker does not take into account the defence strategy that we use. We believe that it is reasonable, since the defence requires access to the decoder part of the model, $p_\theta(x|z)$, which is not necessarily available to the attacker.

In this ablation study we verify how the robustness results change if we construct the attack with the access to the defence strategy. We train an adversarial attack with the modified objective 3, which takes into account the HMC step. See Appendix C.3 for more details on the experimental setup.

In Figure 5 we show the experiment results for various attack radii between 0 and 1. We observe that constructing an attack with such an objective is much harder (Figure 5 (b)).

5 Discussion

Following the previous works on attacking VAEs [6, 10, 12, 13, 18, 42], we only consider the projected gradient descent as a way to construct the attack. However, more sophisticated adaptive methods [3, 38] were proposed to attack discriminative model and can be potentially applied to VAEs as well. We believe that it is an interesting direction for the future work.

Objective Function For the unsupervised attack on the encoder, we use the symmetric KL-divergence to measure the dissimilarity. However, other options are possible, e.g., the forward or reverse KL-divergence or even L_2 distance between the means (see Table 1). In our comparative experiments (see Appendix C.6), we observe that no single objective consistently performs better than others.

Attack radius During the attack construction we seek to obtain a point that will have the most different latent representation or a different predicted class. However, it is also important that the point itself is as similar to the initial reference point as possible. In Appendix C.4, we visualize how the attacks of different radii influence the similarity between the adversarial and reference points. Based on these results, we have chosen the radii which do not allow adversarial points to deviate a lot from the reference (as measured by MSSSIM): $\|\varepsilon\|_{\text{inf}} \leq 0.2$ for the MNIST dataset (which goes in line with the previous works [12, 13]) and $\|\varepsilon\|_{\text{inf}} \leq 0.1$ for the CelebA dataset.

Number of MCMC steps and Inference Time In our approach, we have to select the number of MCMC steps that the defender performs. This parameter potentially can be critical as it influences both the inference time (see Appendix C.7) and the performance (see Appendix C.5). In Figure 6 we show the trade-off between the reconstruction similarity and the inference time. The increase in the inference time is cause by a larger number of HMC steps used (we consider 0, 100, 500 and 1000 steps for this experiment).

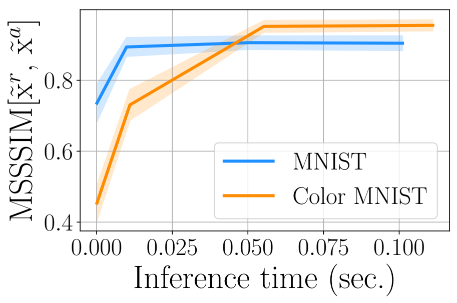


Figure 6: Trade-off between robustness and inference time.

6 Conclusion

In this work, we explore the robustness of VAEs to adversarial attacks. We propose a theoretically justified method that allows alleviating the effect of attacks on the latent representations by improving the reconstructions of the adversarial inputs and the downstream tasks accuracy. We experimentally validate our approach on a variety of datasets: both grey-scale (MNIST, Fashion MNIST) and colored (ColorMNIST, CelebA) data. We show that the proposed method improves the robustness of the vanilla VAE models and its various modifications, i.e., β-VAE, β-TCVAE and NVAE.

Acknowledgements

Anna Kuzina is funded by the Hybrid Intelligence Center, a 10-year programme funded by the Dutch Ministry of Education, Culture and Science through the Netherlands Organisation for Scientific Research. Experiments were carried out on the Dutch national e-infrastructure with the support of SURF Cooperative

References

[1] J. An and S. Cho. Variational autoencoder based anomaly detection using reconstruction probability. *Special Lecture on IE*, 2(1):1–18, 2015.

[2] C. Andrieu, N. De Freitas, A. Doucet, and M. I. Jordan. An introduction to mcmc for machine learning. *Machine learning*, 50(1):5–43, 2003.

[3] A. Athalye, N. Carlini, and D. Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International conference on machine learning*, pages 274–283. PMLR, 2018.

[4] I. A. Auzina and J. M. Tomczak. Approximate bayesian computation for discrete spaces. *Entropy*, 23(3):312, 2021.

[5] J. Ballé, D. Minnen, S. Singh, S. J. Hwang, and N. Johnston. Variational image compression with a scale hyperprior. In *International Conference on Learning Representations*, 2018.

[6] B. Barrett, A. Camuto, M. Willetts, and T. Rainforth. Certifiably robust variational autoencoders. *International Conference on Artificial Intelligence and Statistics*, pages 3663–3683, 2022.

[7] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[8] M. Betancourt. A conceptual introduction to hamiltonian monte carlo. *arXiv preprint arXiv:1701.02434*, 2017.

[9] Y. Burda, R. B. Grosse, and R. Salakhutdinov. Importance weighted autoencoders. In *International Conference on Learning Representations*, 2016.

[10] A. Camuto, M. Willetts, S. Roberts, C. Holmes, and T. Rainforth. Towards a theoretical understanding of the robustness of variational autoencoders. In *International Conference on Artificial Intelligence and Statistics*, pages 3565–3573. PMLR, 2021.

[11] A. L. Caterini, A. Doucet, and D. Sejdinovic. Hamiltonian variational auto-encoder. *Advances in Neural Information Processing Systems*, 31, 2018.

[12] T. Cemgil, S. Ghaisas, K. Dvijotham, S. Gowal, and P. Kohli. The autoencoding variational autoencoder. *Advances in Neural Information Processing Systems*, 33, 2020.

[13] T. Cemgil, S. Ghaisas, K. D. Dvijotham, and P. Kohli. Adversarially robust representations with smooth encoders. In *International Conference on Learning Representations*. openreview.net, 2019.

[14] R. T. Chen, X. Li, R. B. Grosse, and D. K. Duvenaud. Isolating sources of disentanglement in variational autoencoders. *Advances in neural information processing systems*, 31, 2018.

[15] R. Child. Very deep VAEs generalize autoregressive models and can outperform them on images. In *International Conference on Learning Representations*, 2021.

[16] C. Cremer, X. Li, and D. Duvenaud. Inference suboptimality in variational autoencoders. In *International Conference on Machine Learning*, pages 1078–1086. PMLR, 2018.

[17] S. Duane, A. D. Kennedy, B. J. Pendleton, and D. Roweth. Hybrid monte carlo. *Physics letters B*, 195(2):216–222, 1987.

[18] G. Gondim-Ribeiro, P. Tabacof, and E. Valle. Adversarial attacks on variational autoencoders. *ArXiv Preprint*, June 2018.

[19] I. J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[20] A. Habibian, T. v. Rozendaal, J. M. Tomczak, and T. S. Cohen. Video compression with rate-distortion autoencoders. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7033–7042, 2019.

[21] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. *International Conference on Learning Representations (ICLR)*, 2017.

[22] I. Higgins, A. Pal, A. Rusu, L. Matthey, C. Burgess, A. Pritzel, M. Botvinick, C. Blundell, and A. Lerchner. Darla: Improving zero-shot transfer in reinforcement learning. In *International Conference on Machine Learning*, pages 1480–1490. PMLR, 2017.

[23] M. D. Hoffman. Learning deep latent gaussian models with markov chain monte carlo. In *International conference on machine learning*, pages 1510–1519. PMLR, 2017.

[24] P. Izmailov, S. Vikram, M. D. Hoffman, and A. G. G. Wilson. What are bayesian neural network posteriors really like? In *International conference on machine learning*, pages 4629–4640. PMLR, 2021.

[25] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

[26] H. Kim and A. Mnih. Disentangling by factorising. In *International Conference on Machine Learning*, pages 2649–2658. PMLR, 2018.

[27] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations, ICLR 2014*, 2014.

[28] A. Kuzina, M. Welling, and J. M. Tomczak. Diagnosing vulnerability of variational auto-encoders to adversarial attacks. *ICLR, RobustML workshop*, 2021.

[29] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[30] L. Maaløe, M. Fraccaro, V. Liévin, and O. Winther. BIVA: A very deep hierarchy of latent variables for generative modeling. In *Advances in Neural Information Processing Systems*, Feb. 2019.

[31] R. M. Neal et al. Mcmc using hamiltonian dynamics. *Handbook of markov chain monte carlo*, 2(11):2, 2011.

[32] A. Nishimura, D. B. Dunson, and J. Lu. Discontinuous hamiltonian monte carlo for discrete parameters and discontinuous likelihoods. *Biometrika*, 107(2):365–380, 2020.

[33] R. Ranganath, D. Tran, and D. Blei. Hierarchical variational models. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 324–333, New York, New York, USA, 2016. PMLR.

[34] D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, pages 1278–1286. PMLR, 2014.

[35] F. Ruiz and M. Titsias. A contrastive divergence for combining variational inference and mcmc. In *International Conference on Machine Learning*, pages 5537–5545. PMLR, 2019.

[36] T. Salimans, D. Kingma, and M. Welling. Markov chain monte carlo and variational inference: Bridging the gap. In *International conference on machine learning*, pages 1218–1226. PMLR, 2015.

[37] C. K. Sønderby, T. Raiko, L. Maalø e, S. R. K. Sø nderby, and O. Winther. Ladder variational autoencoders. In *Advances in Neural Information Processing Systems*, volume 29, pages 3738–3746. Curran Associates, Inc., 2016.

[38] F. Tramer, N. Carlini, W. Brendel, and A. Madry. On adaptive attacks to adversarial example defenses. *Advances in Neural Information Processing Systems*, 33:1633–1645, 2020.

[39] A. Vahdat and J. Kautz. NVAE: A deep hierarchical variational autoencoder. In *Neural Information Processing Systems (NeurIPS)*, 2020.

[40] A. Van Den Oord, O. Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

[41] Z. Wang, E. P. Simoncelli, and A. C. Bovik. Multiscale structural similarity for image quality assessment. In *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, volume 2, pages 1398–1402. Ieee, 2003.

[42] M. Willetts, A. Camuto, T. Rainforth, S. Roberts, and C. Holmes. Improving VAEs' robustness to adversarial attack. In *International Conference on Learning Representations*, 2021.

[43] C. Wolf, M. Karl, and P. van der Smagt. Variational inference with hamiltonian monte carlo. *arXiv preprint arXiv:1609.08203*, 2016.

[44] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[45] R. Zhang, X. Liu, and Q. Liu. A langevin-like sampler for discrete distributions. In *International Conference on Machine Learning*, pages 26375–26396. PMLR, 2022.

Checklist

1. For all authors...
    (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
    (b) Did you describe the limitations of your work? [Yes]
    (c) Did you discuss any potential negative societal impacts of your work? [N/A]
    (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
    (a) Did you state the full set of assumptions of all theoretical results? [Yes] See Section 3
    (b) Did you include complete proofs of all theoretical results? [Yes] See Section 3 and Appendix A
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]
    (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix D
    (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 total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [No]
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] We use NVAE code and pretrained that are published on github
    (b) Did you mention the license of the assets? [Yes] The licence allows using code for non-commercial purposes
    (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? [N/A]
    (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
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]

A Theory

We consider an attack, which has an additive structure:

$$\mathbf{x}^a = \mathbf{x}^r + \varepsilon, \tag{12}$$

$$\text{such that } \|\varepsilon\| \leq \delta, \tag{13}$$

where δ is a radius of the attack.

In the vanilla VAE setup we will get the latent code by sampling from $q_\phi(\mathbf{z}|\mathbf{x})$. With our approach, instead, we get a sample from the following distribution:

$$q^{(t)}(\mathbf{z}|\mathbf{x}) = \int Q^{(t)}(\mathbf{z}|\mathbf{z}_0) q_\phi(\mathbf{z}_0|\mathbf{x}) d\mathbf{z}_0, \tag{14}$$

where $Q^{(t)}(\mathbf{z}|\mathbf{z}_0)$ is a transition kernel of MCMC with the target distribution $\pi(\mathbf{z}) = p_\theta(\mathbf{z}|\mathbf{x}) \propto p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z})$.

Lemma 1 Consider true posterior distributions of the latent code $\mathbf{z}$ for a data point $\mathbf{x}$ and its corrupted version $\mathbf{x}^a$. Assume also that $\ln p_\theta(\mathbf{z}|\mathbf{x})$ is twice differentiable over $\mathbf{x}$ with continuous derivatives at the neighbourhood around $\mathbf{x} = \mathbf{x}^r$. Then the KL-divergence between these two posteriors could be expressed using the small o notation of the radius of the attack, namely:

$$D_{\mathrm{KL}}\left[p_\theta(\mathbf{z}|\mathbf{x}^r) \| p_\theta(\mathbf{z}|\mathbf{x}^a)\right] = o(\|\varepsilon\|). \tag{15}$$

Proof
Let us use definition of the KL-divergence:

$$D_{\mathrm{KL}}\left[p_\theta(\mathbf{z}|\mathbf{x}^r) \| p_\theta(\mathbf{z}|\mathbf{x}^a)\right] = \mathbb{E}_{p_\theta(\mathbf{z}|\mathbf{x}^r)} \ln \frac{p_\theta(\mathbf{z}|\mathbf{x}^r)}{p_\theta(\mathbf{z}|\mathbf{x}^a)} \tag{16}$$

Let us introduce $\ln p_\theta(\mathbf{z}|\mathbf{x}) = g(\mathbf{x}, \mathbf{z})$. Assume that this function is differentiable at $\mathbf{x} = \mathbf{x}^r$. Then, we can apply Taylor expansion to $g(\mathbf{x}, \mathbf{z})$ in the point $\mathbf{x}^r$ which yields:

$$g(\mathbf{x}, \mathbf{z}) = g(\mathbf{x}^r, \mathbf{z}) + (\mathbf{x} - \mathbf{x}^r)^T \nabla_{\mathbf{x}} g(\mathbf{x}, \mathbf{z})\Big|_{\mathbf{x}^r} + R_1(\mathbf{x}, \mathbf{x}^r). \tag{17}$$

The remainder term in the Lagrange form can be written as

$$R_1(\mathbf{x}, \mathbf{x}^r) = \tfrac{1}{2}(\mathbf{x} - \mathbf{x}^r)^T \nabla^2_{\mathbf{x}\mathbf{x}} g(\mathbf{x} + \theta(\mathbf{x} - \mathbf{x}^r), \mathbf{z})(\mathbf{x} - \mathbf{x}^r),\ \theta \in (0, 1) \tag{18}$$

Under the assumption that g is twice differentiable with the continuous derivatives on the segment around $\mathbf{x} = \mathbf{x}^r$ the remainder term asymptotically converges to zero with $\mathbf{x} \to \mathbf{x}^r$.

$$R_1(\mathbf{x}, \mathbf{x}^r) = o(\|\mathbf{x} - \mathbf{x}^r\|). \tag{19}$$

Then, the log-ratio of two distributions is the following:

$$\ln \frac{p_\theta(\mathbf{z}|\mathbf{x}^r)}{p_\theta(\mathbf{z}|\mathbf{x}^a)} = g(\mathbf{x}^r, \mathbf{z}) - g(\mathbf{x}^a, \mathbf{z}) \tag{20}$$

$$= g(\mathbf{x}^r, \mathbf{z}) - \left(g(\mathbf{x}^r, \mathbf{z}) + (\mathbf{x}^a - \mathbf{x}^r)^T \nabla_{\mathbf{x}} g(\mathbf{x}, \mathbf{z})\Big|_{\mathbf{x}^r} + o(\|\mathbf{x}^a - \mathbf{x}^r\|)\right). \tag{21}$$

$$= -\varepsilon^T \nabla_{\mathbf{x}} g(\mathbf{x}, \mathbf{z})\Big|_{\mathbf{x}^r} + o(\|\varepsilon\|). \tag{22}$$

Notice that $\varepsilon^T \nabla_{\mathbf{x}} g(\mathbf{x}, \mathbf{z})\Big|_{\mathbf{x}^r}$ is the dot product between ε and $\nabla_{\mathbf{x}} g(\mathbf{x}, \mathbf{z})\Big|_{\mathbf{x}^r}$, i.e., $\varepsilon^T \nabla_{\mathbf{x}} g(\mathbf{x}, \mathbf{z})\Big|_{\mathbf{x}^r} = \langle \varepsilon, \nabla_{\mathbf{x}} g(\mathbf{x}, \mathbf{z})\Big|_{\mathbf{x}^r} \rangle$.

We can now plug this into the KL-divergence definition (16):

$$D_{\mathrm{KL}}\left[p_\theta(\mathbf{z}|\mathbf{x}^r) \| p_\theta(\mathbf{z}|\mathbf{x}^a)\right] = \mathbb{E}_{p_\theta(\mathbf{z}|\mathbf{x}^r)} \left[-\langle \varepsilon, \nabla_{\mathbf{x}} \ln p_\theta(\mathbf{z}|\mathbf{x})\Big|_{\mathbf{x}^r} \rangle + o(\|\varepsilon\|)\right] \tag{23}$$

$$= -\langle \varepsilon, \underbrace{\mathbb{E}_{p_\theta(\mathbf{z}|\mathbf{x}^r)} \nabla_{\mathbf{x}} \ln p_\theta(\mathbf{z}|\mathbf{x})\Big|_{\mathbf{x}^r}}_{\mathrm{A}(\mathbf{x}^r)} \rangle + o(\|\varepsilon\|) \tag{24}$$

Note that for (24) to hold we need to make sure that $\mathbb{E}_z R_1 = o(\|\varepsilon\|)$. As follows from (18), this requirement is satisfied if $\mathbb{E}_z \nabla^2_{\mathbf{x}\mathbf{x}} g(\mathbf{x} + \theta(\mathbf{x} - \mathbf{x}^r), \mathbf{z})$ is bounded around $\mathbf{x} = \mathbf{x}^r$.

Let us take a closer to look at the term $\mathrm{A}(\mathbf{x}^r)$ in the equation above:

$$
\begin{aligned}
\mathrm{A}(\mathbf{x}^r) &= \mathbb{E}_{p_\theta(\mathbf{z}|\mathbf{x}^r)} \nabla_{\mathbf{x}} \ln p_\theta(\mathbf{z}|\mathbf{x}) \Big|_{\mathbf{x}^r} && (25)\\
&= \mathbb{E}_{p_\theta(\mathbf{z}|\mathbf{x}^r)} \nabla_{\mathbf{x}} \ln \frac{p_\theta(\mathbf{x}|\mathbf{z}) p(\mathbf{z})}{p_\theta(\mathbf{x})} \Big|_{\mathbf{x}^r} && (26)\\
&= \mathbb{E}_{p_\theta(\mathbf{z}|\mathbf{x}^r)} \nabla_{\mathbf{x}} \ln p_\theta(\mathbf{x}|\mathbf{z}) \Big|_{\mathbf{x}^r} - \mathbb{E}_{p_\theta(\mathbf{z}|\mathbf{x}^r)} \nabla_{\mathbf{x}} \ln p_\theta(\mathbf{x}) \Big|_{\mathbf{x}^r} && (27)\\
&= \int p_\theta(\mathbf{z}|\mathbf{x}^r) \frac{\nabla_{\mathbf{x}} p_\theta(\mathbf{x}|\mathbf{z}) \Big|_{\mathbf{x}^r}}{p_\theta(\mathbf{x}^r|\mathbf{z})} d\mathbf{z} - \int p_\theta(\mathbf{z}|\mathbf{x}^r) \frac{\nabla_{\mathbf{x}} p_\theta(\mathbf{x}) \Big|_{\mathbf{x}^r}}{p_\theta(\mathbf{x}^r)} d\mathbf{z} && (28)\\
&= \int \frac{p_\theta(\mathbf{z})}{p_\theta(\mathbf{x}^r)} \nabla_{\mathbf{x}} p_\theta(\mathbf{x}|\mathbf{z}) \Big|_{\mathbf{x}^r} d\mathbf{z} - \frac{\nabla_{\mathbf{x}} p_\theta(\mathbf{x}) \Big|_{\mathbf{x}^r}}{p_\theta(\mathbf{x}^r)} \underbrace{\int p_\theta(\mathbf{z}|\mathbf{x}^r) d\mathbf{z}}_{=1} && (29)\\
&= \frac{1}{p_\theta(\mathbf{x}^r)} \left[\int p(\mathbf{z}) \nabla_{\mathbf{x}} p_\theta(\mathbf{x}|\mathbf{z}) \Big|_{\mathbf{x}^r} d\mathbf{z} - \nabla_{\mathbf{x}} p_\theta(\mathbf{x}) \Big|_{\mathbf{x}^r} \right] && (30)\\
&= \frac{1}{p_\theta(\mathbf{x}^r)} \left[\mathbb{E}_{p(\mathbf{z})} \nabla_{\mathbf{x}} p_\theta(\mathbf{x}|\mathbf{z}) \Big|_{\mathbf{x}^r} - \nabla_{\mathbf{x}} \mathbb{E}_{p(\mathbf{z})} p_\theta(\mathbf{x}|\mathbf{z}) \Big|_{\mathbf{x}^r} \right] && (31)\\
&= \frac{1}{p_\theta(\mathbf{x}^r)} \underbrace{\left[\mathbb{E}_{p(\mathbf{z})} \nabla_{\mathbf{x}} p_\theta(\mathbf{x}|\mathbf{z}) \Big|_{\mathbf{x}^r} - \mathbb{E}_{p(\mathbf{z})} \nabla_{\mathbf{x}} p_\theta(\mathbf{x}|\mathbf{z}) \Big|_{\mathbf{x}^r} \right]}_{=0} = 0. && (32)
\end{aligned}
$$

where we use Bayes rule in 26, 29, log-derivative trick in 28.

We have shown that $\mathrm{A}(\mathbf{x}^r) = 0$, therefore, from equation 24 we have:

$$
D_{\mathrm{KL}} \left[p_\theta(\mathbf{z}|\mathbf{x}^r) \| p_\theta(\mathbf{z}|\mathbf{x}^a) \right] = -\langle \varepsilon, \mathrm{A}(\mathbf{x}^r) \rangle + o(\|\varepsilon\|) = o(\|\varepsilon\|). \qquad (33)
$$

■

Lemma 2 The Total Variation distance (TV) between the variational posterior with MCMC for a given corrupted point $\mathbf{x}^a$, $q^{(t)}(\mathbf{z}|\mathbf{x}^a)$, and the variational posterior for a given data point $\mathbf{x}^r$, $q_\phi(\mathbf{z}|\mathbf{x}^r)$, can be upper bounded by the sum of the following three components:

$$
\begin{aligned}
\mathrm{TV}\left[q^{(t)}(\mathbf{z}|\mathbf{x}^a), q_\phi(\mathbf{z}|\mathbf{x}^r) \right] \leq\ & \mathrm{TV}\left[q^{(t)}(\mathbf{z}|\mathbf{x}^a), p_\theta(\mathbf{z}|\mathbf{x}^a) \right] && (34)\\
&+ \sqrt{\tfrac{1}{2} D_{\mathrm{KL}} \left[p_\theta(\mathbf{z}|\mathbf{x}^r) \| p_\theta(\mathbf{z}|\mathbf{x}^a) \right]} && (35)\\
&+ \sqrt{\tfrac{1}{2} D_{\mathrm{KL}} \left[q_\phi(\mathbf{z}|\mathbf{x}^r) \| p_\theta(\mathbf{z}|\mathbf{x}^r) \right]}. && (36)
\end{aligned}
$$

Proof
Total variation is a proper distance, thus, the triangular inequality holds for it. For the proof, we apply the triangular inequality twice. First, we use the triangle inequality for $\mathrm{TV}\left[q^{(t)}(\mathbf{z}|\mathbf{x}^a), q_\phi(\mathbf{z}|\mathbf{x}^r) \right]$, namely:

$$
\mathrm{TV}\left[q^{(t)}(\mathbf{z}|\mathbf{x}^a), q_\phi(\mathbf{z}|\mathbf{x}^r) \right] \leq \mathrm{TV}\left[q^{(t)}(\mathbf{z}|\mathbf{x}^a), p_\theta(\mathbf{z}|\mathbf{x}^r) \right] + \mathrm{TV}\left[p_\theta(\mathbf{z}|\mathbf{x}^r), q_\phi(\mathbf{z}|\mathbf{x}^r) \right]. \qquad (37)
$$

Second, we utilize the triangle inequality for $\mathrm{TV}\left[q^{(t)}(\mathbf{z}|\mathbf{x}^a), p_\theta(\mathbf{z}|\mathbf{x}^r) \right]$, that is:

$$
\mathrm{TV}\left[q^{(t)}(\mathbf{z}|\mathbf{x}^a), p_\theta(\mathbf{z}|\mathbf{x}^r) \right] \leq \mathrm{TV}\left[q^{(t)}(\mathbf{z}|\mathbf{x}^a), p_\theta(\mathbf{z}|\mathbf{x}^a) \right] + \mathrm{TV}\left[p_\theta(\mathbf{z}|\mathbf{x}^a), p_\theta(\mathbf{z}|\mathbf{x}^r) \right]. \qquad (38)
$$

Combining the two gives us the following upper bound on the initial total variation:

$$\mathrm{TV}\left[q^{(t)}(\mathbf{z}|\mathbf{x}^a), q_\phi(\mathbf{z}|\mathbf{x}^r)\right] \leq \mathrm{TV}\left[q^{(t)}(\mathbf{z}|\mathbf{x}^a), p_\theta(\mathbf{z}|\mathbf{x}^a)\right] \tag{39}$$

$$+ \mathrm{TV}\left[p_\theta(\mathbf{z}|\mathbf{x}^a), p_\theta(\mathbf{z}|\mathbf{x}^r)\right] \tag{40}$$

$$+ \mathrm{TV}\left[p_\theta(\mathbf{z}|\mathbf{x}^r), q_\phi(\mathbf{z}|\mathbf{x}^r)\right] \tag{41}$$

Moreover, the Total Variation distance is a lower bound of the KL-divergence (by Pinsker inequality):

$$\mathrm{TV}\left[p(\mathbf{x}), q(\mathbf{x})\right] \leq \sqrt{\tfrac{1}{2} D_{\mathrm{KL}}\left[p(\mathbf{x}) \| q(\mathbf{x})\right]}. \tag{42}$$

Applying Pinsker inequality to 40 and 41 yields:

$$\mathrm{TV}\left[q^{(t)}(\mathbf{z}|\mathbf{x}^a), q_\phi(\mathbf{z}|\mathbf{x}^r)\right] \leq \mathrm{TV}\left[q^{(t)}(\mathbf{z}|\mathbf{x}^a), p_\theta(\mathbf{z}|\mathbf{x}^a)\right] \tag{43}$$

$$+ \sqrt{\tfrac{1}{2} D_{\mathrm{KL}}\left[p_\theta(\mathbf{z}|\mathbf{x}^r) \| p_\theta(\mathbf{z}|\mathbf{x}^a)\right]} \tag{44}$$

$$+ \sqrt{\tfrac{1}{2} D_{\mathrm{KL}}\left[q_\phi(\mathbf{z}|\mathbf{x}^r) \| p_\theta(\mathbf{z}|\mathbf{x}^r)\right]}. \tag{45}$$

■

Theorem 1 The upper bound on the total variation distance between samples from MCMC for a given corrupted point $\mathbf{x}^a$, $q^{(t)}(\mathbf{z}|\mathbf{x}^a)$, and the variational posterior for the given real point $\mathbf{x}$, $q_\phi(\mathbf{z}|\mathbf{x})$, is the following:

$$\mathrm{TV}\left[q^{(t)}(\mathbf{z}|\mathbf{x}^a), q_\phi(\mathbf{z}|\mathbf{x}^r)\right] \leq \mathrm{TV}\left[q^{(t)}(\mathbf{z}|\mathbf{x}^a), p_\theta(\mathbf{z}|\mathbf{x}^a)\right] + \sqrt{\tfrac{1}{2} D_{\mathrm{KL}}\left[q_\phi(\mathbf{z}|\mathbf{x}^r) \| p_\theta(\mathbf{z}|\mathbf{x}^r)\right]} + o(\sqrt{\|\varepsilon\|}). \tag{46}$$

Proof
Combining **Lemma 1** and **2** we get:

$$\mathrm{TV}\left[q^{(t)}(\mathbf{z}|\mathbf{x}^a), q_\phi(\mathbf{z}|\mathbf{x}^r)\right] \underbrace{\leq}_{\text{Lemma 2}} \mathrm{TV}\left[q^{(t)}(\mathbf{z}|\mathbf{x}^a), p_\theta(\mathbf{z}|\mathbf{x}^a)\right] \tag{47}$$

$$+ \sqrt{\tfrac{1}{2} D_{\mathrm{KL}}\left[p_\theta(\mathbf{z}|\mathbf{x}^r) \| p_\theta(\mathbf{z}|\mathbf{x}^a)\right]} \tag{48}$$

$$+ \sqrt{\tfrac{1}{2} D_{\mathrm{KL}}\left[q_\phi(\mathbf{z}|\mathbf{x}^r) \| p_\theta(\mathbf{z}|\mathbf{x}^r)\right]} \tag{49}$$

$$\underbrace{=}_{\text{Lemma 1}} \mathrm{TV}\left[q^{(t)}(\mathbf{z}|\mathbf{x}^a), p_\theta(\mathbf{z}|\mathbf{x}^a)\right] \tag{50}$$

$$+ \sqrt{\tfrac{1}{2} o(\|\varepsilon\|)} \tag{51}$$

$$+ \sqrt{\tfrac{1}{2} D_{\mathrm{KL}}\left[q_\phi(\mathbf{z}|\mathbf{x}^r) \| p_\theta(\mathbf{z}|\mathbf{x}^r)\right]}. \tag{52}$$

Note that $\sqrt{\frac{1}{2} o(\|\varepsilon\|)} = o(\sqrt{\|\varepsilon\|})$ that gives us the final expression:

$$\mathrm{TV}\left[q^{(t)}(\mathbf{z}|\mathbf{x}^a), q_\phi(\mathbf{z}|\mathbf{x}^r)\right] \leq \mathrm{TV}\left[q^{(t)}(\mathbf{z}|\mathbf{x}^a), p_\theta(\mathbf{z}|\mathbf{x}^a)\right] + \sqrt{\tfrac{1}{2} D_{\mathrm{KL}}\left[q_\phi(\mathbf{z}|\mathbf{x}^r) \| p_\theta(\mathbf{z}|\mathbf{x}^r)\right]} + o(\sqrt{\|\varepsilon\|}). \tag{53}$$

■

B Background on MCMC

B.1 Sampling from an unnormalized density with MCMC

Markov Chain Monte Carlo (MCMC) is a class of methods that are used to obtain samples from the density $p(\mathbf{v})$ (also referred to as **target**), which is only known up to a normalizing constant. That is, we have access to $\tilde{p}(\mathbf{v})$, such that $p(\mathbf{v}) = \frac{\tilde{p}(\mathbf{v})}{Z}$ and Z is a typically unknown and hard to estimate normalizing constant. Thus, we construct a Markov Chain with samples $\{\mathbf{v}^{(t)}\}_{t=1}^{T}$ so that they mimic the samples from $p(\mathbf{v})$. To ensure they are proper samples, the stationary distribution of the constructed Markov Chain should be the target distribution $p(\mathbf{v})$.

The most popular way of constructing such Markov Chains is the Metropolis-Hastings (MH) method. The majority of the MCMC methods used in practice can be formulated as a special case of the MH [2]. In the MH method, we introduce a proposal distribution $q(\mathbf{v}^{t+1}|\mathbf{v}^{t})$ to obtain a new sample and then accept it with the following probability:

$$\mathcal{A}(\mathbf{v}^t, \mathbf{v}^{t+1}) = \min\{1, \tfrac{p(\mathbf{v}^t)q(\mathbf{v}^{t+1}|\mathbf{v}^t)}{p(\mathbf{v}^{t+1})q(\mathbf{v}^t|\mathbf{v}^{t+1})}\}. \tag{54}$$

If the point is not accepted, we reuse the previous point, i.e., $\mathbf{v}^{t+1} = \mathbf{v}^t$. It can be proven that the resulting chain of correlated samples converges in the distribution to the target density [2].

It is worth mentioning that the performance of the method strongly depends on the choice of the proposal distribution. In higher dimensional spaces, it is especially important to incorporate the information about the geometry of the target distribution into the proposal density to improve the convergence time. The Hamiltonian Monte-Carlo (HMC) [31] is known to be one of the most efficient MCMC methods. It uses gradient of a target distribution in the proposal to incorporate the information about the geometry of the space.

The idea of the HMC is to introduce an auxiliary variable $\mathbf{p}$ with a known density (usually assumed to be the standard Gaussian) and the joint distribution formulated as follows:

$$p(\mathbf{v}, \mathbf{p}) = \frac{1}{Z}\exp(-U(\mathbf{v}))\exp(-K(\mathbf{p})), \tag{55}$$

with:

$$K(\mathbf{p}) = -\frac{1}{2}\mathbf{p}^T\mathbf{p}, \tag{56}$$

$$U(\mathbf{v}) = -\log\tilde{p}(\mathbf{v}). \tag{57}$$

We obtain samples $(\mathbf{v}, \mathbf{p})$ using the Hamiltonian dynamics [31] that describes how the $\mathbf{v}$ and $\mathbf{p}$ change over time for the given Hamiltonian $H(\mathbf{v}, \mathbf{p}) = U(\mathbf{v}) + K(\mathbf{p})$, namely:

$$\dot{\mathbf{v}} = \frac{\partial H}{\partial \mathbf{p}}, \tag{58}$$

$$\dot{\mathbf{p}} = -\frac{\partial H}{\partial \mathbf{v}}. \tag{59}$$

For the practical implementation, these continuous-time equations are approximated by discretizing the time using L small steps of size η. The discretization method that is often used is called the *leapfrog*.

B.2 The MCMC and Variational Autoencoders

In this paper, we use the MCMC to sample from the posterior distribution $p_\theta(\mathbf{z}|\mathbf{x}^a)$. That is, in our case $\mathbf{v} = \mathbf{z}$ and $\tilde{p}(\mathbf{v}) = p_\theta(\mathbf{x}^a|\mathbf{z})p(\mathbf{z})$. The HMC is a widely applied method to sampling from an unknown posterior distribution in deep learning (e.g. [24]). A lot of effort was already done in combining variational inference with MCMC (and more specifically with HMC). Hamiltonian Variational Inference [36, 43] was proposed in order to obtain a more flexible variational approximation.

Different approaches were proposed to use HMC during VAE training. [23] approximate the gradients of the likelihood and avoid the use of variational approximation. [11] propose an unbiased estimate for the ELBO gradient, which allows training Hamiltonian Variational Autoencoder. [35] propose an alternative objective, which uses a contrastive divergence instead of standard KL-divergence.

In this work we are not changing the training procedure, instead, we propose to only use HMC during evaluation.

A possible extension to discrete latent spaces Some VAEs operate on discrete latent spaces, a very popular example would be a VQ-VAE [40]. However, the classical HMC that we use in our experiments is not able to sample from a discrete distribution. Therefore, other MCMC methods should be used in this case, such as population-based MCMC [4], modifications of HMC[32] or Langevin Dynamics [45].

B.3 Mode optimization

In this work we hypothesise that adversarial attacks move latent codes to the region of low probability and we use HMC to get a sample from the high posterior probability region. However, another strategy can be to find the posterior mode instead. Here we explain, what was our motivation to not use this approach.

Posterior modes similarity Ideally, we would like to obtain a sample from the variational posterior $q_\phi(\mathbf{z}|\mathbf{x}^r)$, because our decoder was trained to produce reconstructions from such latent codes. At the same time, VAE was trained to match this variational posterior to the true one $p_\theta(\mathbf{z}|\mathbf{x}^r)$. However, both these distributions are not available to us, since we observe attacked point $\mathbf{x}^a$ instead of the reference $\mathbf{x}^r$.

Instead, we sample from $p_\theta(\mathbf{z}|\mathbf{x}^a)$ and show theoretically that the resulting samples are close (in terms of total variation distance) to the "goal" ones. However, that does guarantee that their modes are the same. Therefore, obtaining the mode of $p_\theta(\mathbf{z}|\mathbf{x}^a)$ is not necessarily a mode of $q_\phi(\mathbf{z}|\mathbf{x}^r)$. Thus, the fact that the HMC allows us to “wander” around that mode may be beneficial.

Concentration of measure During reconstruction, we get a sample from $q(\mathbf{z}|\mathbf{x})$ and pass it to the decoder, thus, a mode can actually be a bad latent code for these purposes. Instead, ideally, we want to get a sample from the typical set where most of the probability mass is concentrated. In theory, the HMC allows us to do that.

Randomness The HMC adds a source of randomness to our defence strategy that potentially makes it harder to attack. This is supported by our experiment in Section 4.4

C Additional results

C.1 Posterior ratio

We motivate our method by the hypothesis that the adversarial attack "shifts" a latent code to the region of a lower posterior density, while our approach moves it back to a high posterior probability region. In Section 3 we theoretically justify our hypothesis, while here we provide an additional empirical evidence.

In order to verify our claim that applying an MCMC method allows us to counteract attacks by moving a latent code from a region of a lower posterior probability mass to a region of a higher density, we propose to quantify this effect by measuring the ratio of posteriors for $\mathbf{z}_1$ and $\mathbf{z}_2$. The true posterior $p(\mathbf{z}|\mathbf{x}^r)$ is not available due to the cumbersome marginal distribution $p(\mathbf{x}^r)$, however, we can calculate the ratio of posteriors because the marginal will cancel out, namely:

$$\begin{aligned} \text{PR}(\mathbf{z}_1, \mathbf{z}_2) &= \tfrac{p_\theta(\mathbf{z}_1|\mathbf{x}^r)}{p_\theta(\mathbf{z}_2|\mathbf{x}^r)} && (60) \\ &= \tfrac{p_\theta(\mathbf{x}^r|\mathbf{z}_1)p(\mathbf{z}_1)}{p_\theta(\mathbf{x}^r|\mathbf{z}_2)p(\mathbf{z}_2)}. && (61) \end{aligned}$$

In our case, we are interested in calculating the posterior ratio between the reference and adversarial latent codes ($\mathbf{z}_1 = \mathbf{z}^r$, $\mathbf{z}_2 = \mathbf{z}^a$) as the baseline, and the posterior ratio between the reference and adversarial code after applying the HMC ($\mathbf{z}_1 = \mathbf{z}^r$, $\mathbf{z}_2 = \mathbf{z}^a_{\text{HMC}}$). The lower the posterior ratio, the better. For practical reasons, we use the logarithm of the posterior ratio (the logarithm does not change the monotonicity and turns products to sums):

$$\log \text{PR}(\mathbf{z}_1, \mathbf{z}_2) = \log p_\theta(\mathbf{x}^r|\mathbf{z}_1) + \log p(\mathbf{z}_1) - \log p_\theta(\mathbf{x}^r|\mathbf{z}_2) - \log p(\mathbf{z}_2). \quad (62)$$

We present results on the log-posterior-ratio calculated on the MNIST dataset. In Figure 7 we show a plot with two histograms: one with the posterior ratio between the reference and adversarial latent codes ($\mathbf{z}_1 = \mathbf{z}^r$, $\mathbf{z}_2 = \mathbf{z}^a$) in blue, and the second histogram of the posterior ratio between the reference and adversarial code after applying the HMC ($\mathbf{z}_1 = \mathbf{z}^r$, $\mathbf{z}_2 = \mathbf{z}^a_{\text{HMC}}$) in orange.

We observe that the histogram has moved to the left after applying the HMC. This indicates that posterior of the adversarial (in the denominator) is increasing when the HMC is used. This is precisely the effect we hoped for and this result provides an empirical evidence in favor of our hypothesis.

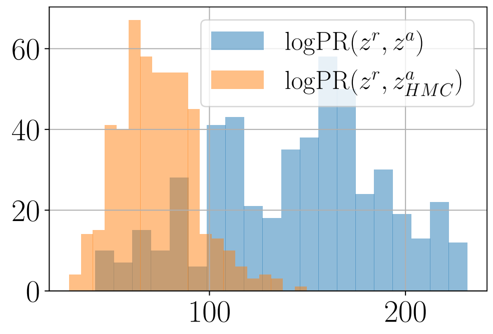


Figure 7: Histograms of the log posterior ratios without HMC (blue) and with HMC (orange) evaluated on MNIST dataset.

Experimental Details For this experiment we construct 500 adversarial attacks with the radius 0.1 on the encoder of the VAE trained on MNIST dataset. We run 500 HMC steps with the same hyperparameters as mentioned in Table 8 to obtain $\mathbf{z}^a_{HMC}$.

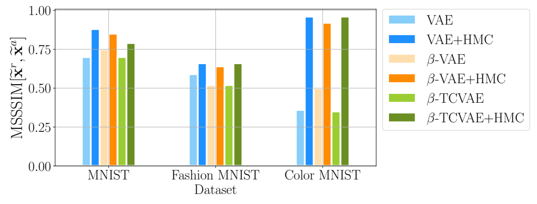
Statistical Analisys We performed a two-sample Kolmagorov-Smirnov test with the null hypothesis that two histograms are drawn from the same distribution. As an alternative hypothesis is that the underlying distributions are different. Choosing the confidence level of 95% results in the rejection of the null hypothesis (p-value is equal to 0.029) in favour of the alternative: two histograms were not drawn from the same distribution.

C.2 Detailed results for β-VAE and β-TCVAE

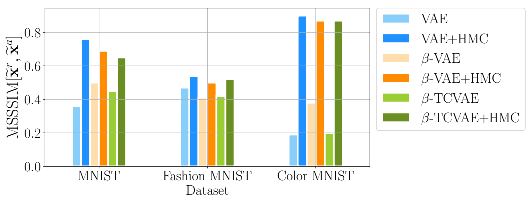
In this section we report extended results for MNIST, FashionMNIST and ColorMNIST datasets. We train VAE, β-VAE and β-TCVAE on three datasets: MNIST, FashionMNIST and ColorMNIST. Then, we compare the robustness to adversarial attack with and without HMC. We present all the result with the standard error in Table 4. On Figures 8, 9 we show visually how HMC improve the robustness for each dataset and model.

Table 4: Robustness results on MNIST, Fashion MNIST and Color MNIST datasets. We perform unsupervised attack with radius 0.1 (top) and 0.2 (bottom). We attack the encoder (left) and the downstream classification task (right). Higher values correspond to more robust models.

		MSSSIM$[\widetilde{\mathbf{x}}^r, \widetilde{\mathbf{x}}^a]$ ↑			Adversarial Accuracy ↑			
		MNIST	Fashion MNIST	Color MNIST	MNIST	Fashion MNIST	Color MNIST Digit	Color MNIST Color
$\lVert\varepsilon\rVert_{\text{inf}} = 0.1$	VAE	0.70 (0.02)	0.59 (0.03)	0.36 (0.03)	0.08 (0.04)	0.00 (0.01)	0.04 (0.03)	0.06 (0.03)
	+ HMC	**0.88 (0.01)**	**0.66 (0.03)**	**0.96 (0.01)**	0.25 (0.03)	**0.14 (0.02)**	0.16 (0.02)	0.68 (0.03)
	β-VAE	0.75 (0.01)	0.52 (0.03)	0.50 (0.04)	0.11 (0.04)	0.00 (0.02)	0.08 (0.04)	0.21 (0.06)
	+ HMC	0.84 (0.01)	0.64 (0.03)	0.92 (0.03)	**0.30 (0.03)**	0.13 (0.02)	0.14 (0.02)	0.66 (0.04)
	β-TCVAE	0.70 (0.02)	0.52 (0.03)	0.35 (0.02)	0.05 (0.03)	0.01 (0.01)	0.08 (0.04)	0.06 (0.03)
	+ HMC	0.79 (0.02)	**0.66 (0.03)**	**0.96 (0.01)**	0.25 (0.04)	0.13 (0.02)	**0.22 (0.03)**	**0.81 (0.02)**
$\lVert\varepsilon\rVert_{\text{inf}} = 0.2$	VAE	0.36 (0.03)	0.47 (0.03)	0.19 (0.02)	0.05 (0.03)	0.01 (0.01)	0.02 (0.02)	0.06 (0.03)
	+ HMC	**0.76 (0.02)**	**0.54 (0.03)**	**0.90 (0.01)**	**0.19 (0.03)**	**0.13 (0.02)**	0.11 (0.02)	0.62 (0.03)
	β-VAE	0.50 (0.03)	0.41 (0.03)	0.38 (0.04)	0.01 (0.01)	0.00 (0.01)	0.05 (0.03)	0.18 (0.05)
	+ HMC	0.69 (0.03)	0.50 (0.03)	0.87 (0.01)	0.16 (0.03)	0.12 (0.02)	**0.15 (0.02)**	0.56 (0.04)
	β-TCVAE	0.45 (0.03)	0.42 (0.03)	0.20 (0.02)	0.03 (0.02)	0.02 (0.02)	0.05 (0.03)	0.05 (0.03)
	+ HMC	0.65 (0.03)	0.52 (0.03)	0.87 (0.01)	0.16 (0.04)	0.11 (0.02)	0.14 (0.02)	**0.72 (0.03)**

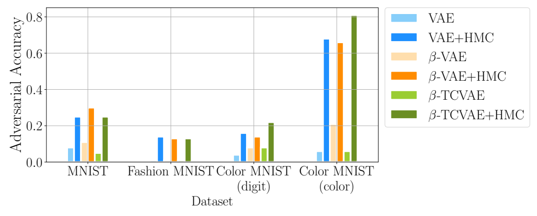


(a) $||\varepsilon||_{\text{inf}} = 0.1$

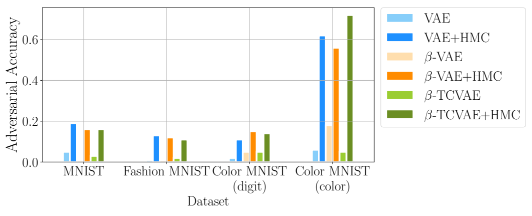


(b) $||\varepsilon||_{\text{inf}} = 0.2$

Figure 8: Improvement of the Reconstruction Similarity after the proposed defence. We fix the attack radius to be equal to (a) 0.1 and (b) 0.2. Higher values correspond to a more robust representations.



(a) $||\varepsilon||_{\text{inf}} = 0.1$



(b) $||\varepsilon||_{\text{inf}} = 0.2$

Figure 9: Improvement of the Adversarial Accuracy after proposed defence. We fix the attack radius to be equal to (a) 0.1 and (b) 0.2.

C.3 What if the attacker knows the defence strategy?

In our experiments we relied on the assumption that attack does not take into account the defence strategy that we use. We believe that it is reasonable, since defence requires access to the decoder part of the model ($p_\theta(x|z)$), which is not necessarily available to the attacker.

However, one may assume that the defence strategy is known to the attacker. In this case, it is reasonable to verify whether the robustness results change. In the conducted experiment we show that it is vastly more complicated to attack the encoder with taking the MCMC defence into account. We train the unsupervised attack (3). The attack has access to the encoder and MCMC defence:

$$f(x) = q^{(t)}(\mathbf{z}|\mathbf{x}) = \int Q^{(t)}(\mathbf{z}|\mathbf{z}_0) q_\phi(\mathbf{z}_0|\mathbf{x}) d\mathbf{z}_0, \tag{63}$$

where $Q^{(t)}(\mathbf{z}|\mathbf{z}_0)$ is MCMC kernel.

Then, given the attack radius δ, we train the attack using the following objective:

$$\varepsilon^* = \arg \max_{\|\varepsilon\|_{\inf} < \delta} \|\widetilde{z}^a - \widetilde{z}^r\|^2, \tag{64}$$

$$\widetilde{z}^a \sim q^{(t)}(\mathbf{z}|\mathbf{x}^r + \varepsilon), \tag{65}$$

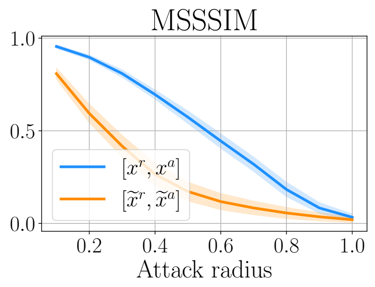
$$\widetilde{z}^r \sim q^{(t)}(\mathbf{z}|\mathbf{x}^r). \tag{66}$$

The similarity results of these attacks are plotted in Figure 11. We observe that the reconstructed reference and adversarial points have approximately the same similarity (measured by MSSSIM) as the initial points $\mathbf{x}^r$ and $\mathbf{x}^a$, which indicates that the attacks were unsuccessful.

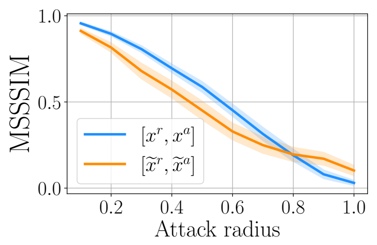
However, if we use the same objective, but omit the MCMC step (e.g $t = 0$ in eq. 65 and 66), then, as observed in Figure 10, the attack becomes much more successful (Figure 10 (a)), but we can fix it with the proposed defence (Figure 10 (b)).

It is interesting to compare how the attacked points look in both cases, especially as we increase the radius of the attack. In Figure 12, we plot attack on two reference points for radius values in $\{0.1, 0.6, 0.8, 1.0\}$. When the attacker does not use MCMC (left), it just learns to add more and more noise to the image, which eventually makes it meaningless.

When we use MCMC during an attack, the situation is different. The adversarial input is almost indistinguishable from the reference point for a small radius. After each gradient update, the attacker runs a new MCMC, which moves point closer to the region of high posterior probability, but may follow a different trajectory every time. Eventually, it makes it harder to learn an additive perturbation ε. However, as we increase the attack radius, we observe a very interesting effect. Instead of meaningless noise, the attacker learns to change the digit. For instance, we see how 4 is transformed into 0 in the first example and into 9 in the second. This way, the attacker ensures that the MCMC will move the latent far away from the reference latent code, which now has a different posterior distribution.

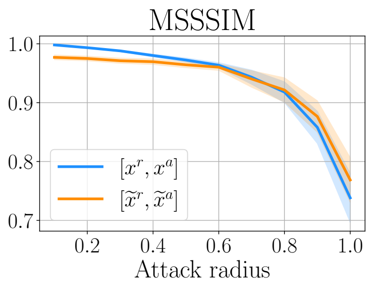


(a) No defence

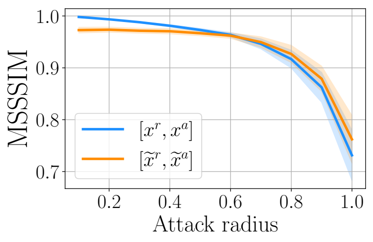


(b) MCMC defence

Figure 10: Adversarial attack, if attacker **does not use MCMC**. We report similarity of the reference and adversarial point before forward pass (blue) and after forward pass (orange).



(a) No defence



(b) MCMC defence

Figure 11: Adversarial attack, if attacker **uses MCMC**. We report similarity of the reference and adversarial point before forward pass (blue) and after forward pass (orange).

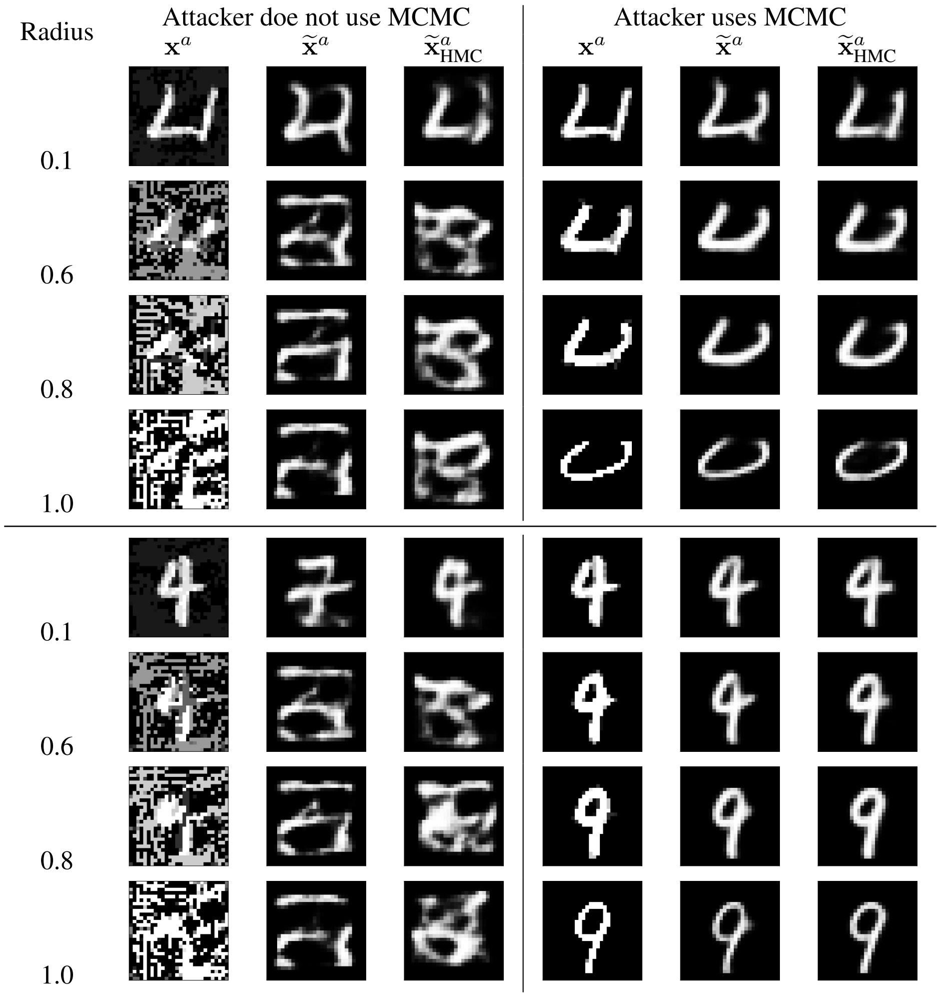
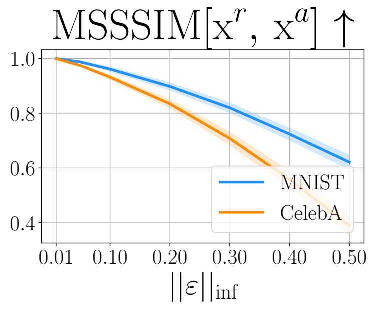

Radius
Attacker doe not use MCMC
Attacker uses MCMC
$\mathbf{x}^a$
$\widetilde{\mathbf{x}}^a$
$\widetilde{\mathbf{x}}^a_{\text{HMC}}$
$\mathbf{x}^a$
$\widetilde{\mathbf{x}}^a$
$\widetilde{\mathbf{x}}^a_{\text{HMC}}$
0.1
0.6
0.8
1.0
0.1
0.6
0.8
1.0

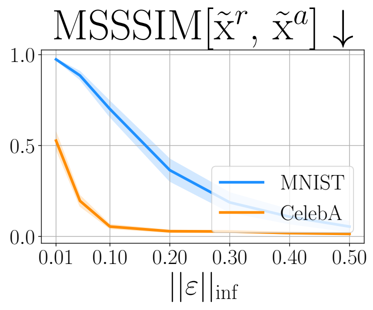
Figure 12: Examples of adversarial point and their reconstructions, when attacker does not use MCMC (left) and when attacker uses MCMC(right).

C.4 Which attack radius should be considered?

In out experiments, we use attacks with the radius 0.1 and 0.2 for all the models except for CelebA dataset, where radii 0.05 and 0.1 were considered. Here, we provide additional experiment to justify this choice. In Figure 13 (a) we show the similarity between the reference point and the adversarial point. We observe that for CelebA the similarity drops faster than for the MNIST. Further, if we look at the example plotted in Figure 14, we can clearly notice that with the radius 0.2 CelebA image is already containing a lot of noise. At the same time, we observe (Figure 13 (b)) that reconstruction similarity, which indicates the success of the attack, drops relatively fast when the radius of the attack increases.



(a) Reference and adversarial point similarity



(b) Reconstruction similarity

Figure 13: Average images similarity (a) before it is passed to VAE and (b) after image is encoded and decoded back. We consider unsupervised attack on the encoder with the radiuses ranging from 0.01 to 0.5 for MNIST and CelebA datasets.

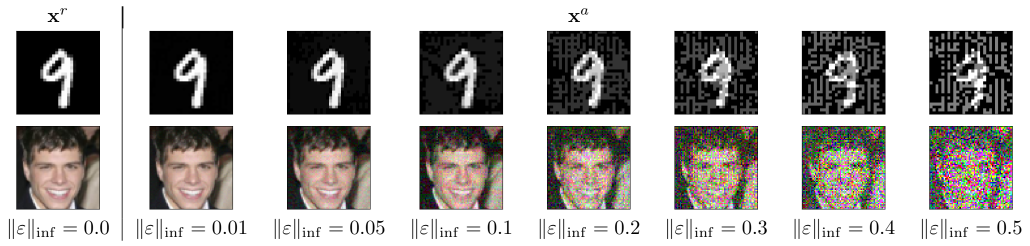
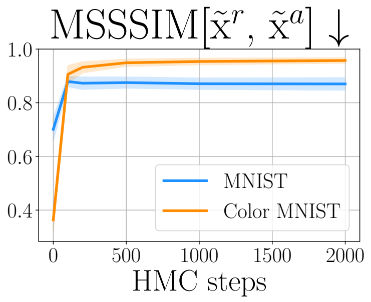

$\mathbf{x}^r$
$\mathbf{x}^a$
$\|\varepsilon\|_{\text{inf}} = 0.0$
$\|\varepsilon\|_{\text{inf}} = 0.01$
$\|\varepsilon\|_{\text{inf}} = 0.05$
$\|\varepsilon\|_{\text{inf}} = 0.1$
$\|\varepsilon\|_{\text{inf}} = 0.2$
$\|\varepsilon\|_{\text{inf}} = 0.3$
$\|\varepsilon\|_{\text{inf}} = 0.4$
$\|\varepsilon\|_{\text{inf}} = 0.5$

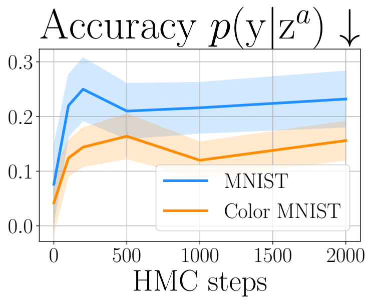
Figure 14

C.5 How many HMC steps are required for a defence?

One of the main hyperparameters of the proposed approach is number of steps of MCMC that the defender does. We have conducted experiments with MNIST and Color MNSIT dataset to see how the robustness metrics change when we increase number of HMC steps from 0 to 200. As we can see from the Figure 15, there is always a considerable jump between 0 steps (no defence) and 100 steps (lowest number of steps considered). However, as we continue making steps, we do not observe further improvement of the metrics.



(a) Reconstruction similarity.

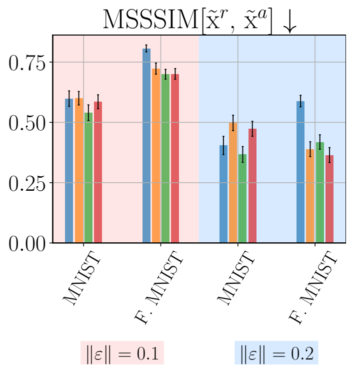


(b) Adversarial Accuracy (digit classification task).

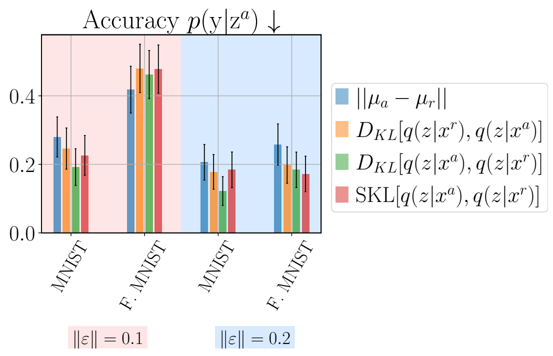
Figure 15: Example of the reference point (leftmost column) and adversarial points for different raduises of the attack.

C.6 Comparison of objective functions

This section compares different objective functions that can be used to construct adversarial attacks on VAE. In general, in both supervised and unsupervised setting, we need to measure the difference between variational posterior in the adversarial point $q(\mathbf{z}|\mathbf{x}^a)$ and a point from the dataset (either a target or reference point). We consider a Gaussian encoder, and the simplest way to compare two Gaussian distributions is to measure the distance between their means. To take into account the variances, we can use the KL-divergence. It is a non-symmetric metric. Thus, we have two options: to use the forward or reverse KL. Finally, it is also possible to consider the symmetrical KL divergence that is an average between the two.



(a) Reconstructions similarity: adversarial point and the reference.



(b) Adversarial Accuracy.

Figure 16: Comparison of different objectives function used to train an attack. Arrows represent the direction of the successful attack.

In Figure 16, we measure how successful the attacks are in terms of the proposed metrics. We use arrows in the plot titles to indicate desirable values of the metric for a successful attack. We compare supervised and unsupervised attacks on VAE trained on MNIST and fashion MNIST datasets. We observe that there is no single objective function that consistently outperforms others.

C.7 Inference Time

Even though our approach does not require changing the training procedure, it has influence on the inference time. In practice, this can be a limiting factor. Therefore, in Table 5 we report the computational overhead during the inference time. We measure the inference time in seconds per test point without HMC ($T = 0$) and for different budgets ($T = \{100, 500, 1000\}$).

Table 5: Inference wall-clock time of the VAE for various number of MCMC steps (T).

T	0	100	500	1000
	VAE			
MNIST	0.0001	0.0099	0.0505	0.1011
COLOR MNIST	0.0001	0.0110	0.0553	0.1111
	NVAE			
CELEBA	0.429	6.512	31.551	63.031

D Details of the experiments

D.1 Training VAE models

Architecture We use the same fully convolutional architecture with latent dimension 64 for MNIST, FashionMNISt and ColorMNIST datasets. In Table 6, we provide detailed scheme of the architecture. We use `Conv(3x3, 1->32)` to denote convolution with kernel size `3x3`, `1` input channel and `32` output channels. We denote stride of the convolution with `s`, padding with `p` and dilation with `d`. The same notation applied for the transposed convolutions (`ConvTranspose`). ColorMNIST has 3 input channels, so the first convolutional layer in the encoder and the last of the decoder are slightly different. In this cases values for ColorMNIST are report in parenthesis with the red color.

Table 6: Convolutional architecture for VAE trained on MNIST, Fashion MNIST and ColorMNIST datasets.

Encoder	Decoder
`Conv(3x3, 1(3)->32, s=2, p=1)`	`ConvTranspose(3x3,64->128,s=1,p=0, d=2)`
`ReLU()`	`ReLU()`
`Conv(3x3, 32->64, s=2, p=1)`	`ConvTranspose(3x3,128->96,s=1,p=0)`
`ReLU()`	`ReLU()`
`Conv(3x3, 64->96, s=2, p=1)`	`ConvTranspose(3x3,96->64,s=1,p=1)`
`ReLU()`	`ReLU()`
`Conv(3x3,96->128,s=2,p=1)`	`ConvTranspose(4x4,64->32,s=2,p=1)`
`ReLU()`	`ReLU()`
$\mu_z \leftarrow$ `Conv(3x3,128->64,s=2,p=1)`	`ConvTranspose(4x4,31->1(3),s=2,p=1)`
$\log \sigma_z^2 \leftarrow$ `Conv(3x3,128->64,s=2,p=1)`	$\mu_x \leftarrow$ `Sigmoid() (Identity())`

Optimization We use Adam to perform the optimization. We start from the learning rate $5e-4$ and reduce it by the factor of 2 if the validation loss does not decrease for 10 epochs. We train a model for 300 epochs with the batch size 128. In Table 7, we report the values of the test metrics for VAEs trained on MNIST, Fashion MNIST and Color MNIST.

For calculating the FID score, we use `torchmetrics` library: `https://torchmetrics.readthedocs.io/en/latest/references/modules.html#frechetinceptiondistance`.

Table 7: Test performance of the β-VAE and β-TCVAE with different values of β. Negative loglikelihood is estimated with importance sampling ($k = 1000$) as suggested in [9].

		MNIST		Fashion MNIST		Color MNIST		
	β	$-\log p(\mathbf{x})$	MSE	$-\log p(\mathbf{x})$	MSE	$-\log p(\mathbf{x})$	MSE	FID
	1	**88.3**	578.6	**232.8**	814.3	**54.87**	261.3	2.09
β-VAE	2	89.3	824.2	234.1	1021.1	55.6	365.6	2.4
	5	100.6	1485.1	241.8	1457.8	63.6	586.1	2.5
	10	126.8	2498.9	248.7	1842.3	88.7	936.2	2.4
β-TCVAE	2	89.3	828.4	233.6	980.4	55.8	366.4	3.0
	5	96.7	1325.4	238.2	1024.6	63.0	574.8	2.0
	10	107.2	1686.1	247.5	1570.0	76.5	806.2	2.2

D.2 Adversarial Attacks and Defence Hyperparameters

In Table 8, we report all the hyperparameter values that were used to attack and defend VAE models.

In all the experiments we randomly select reference points from the test dataset. We also ensure that the resulting samples are properly stratified — include an even number of points from each of the classes. For each reference point, we train 10 adversarial inputs with the same objective function but different initialization.

We use projected gradient descent to learn the adversarial attacks. Optimization parameters were the same for all the datasets and models. They are presented in Table 8.

We choose HMC to defend the model against the trained attack. We perform T steps of HMC with the step size η and L leapfrog steps. Where indicated, we adapt the step size after each step of HMC using the following formula:

$$\eta_t = \eta_{t-1} + 0.01 \cdot \frac{\alpha_{t-1} - 0.9}{0.9} \cdot \eta_{t-1}, \tag{67}$$

where α_t is the acceptance rate at step t. This way we increase the step size if the acceptance rate is higher than 90% and decrease it otherwise. When adaptive steps size is used, a value in the table indicates the η_0.

Table 8: Full list of hyperparameters for attack construction and the defence.

		VAE			NVAE	
		MNIST	Fashion MNIST	Color MNIST	MNIST	CelebA
	# of reference points	50	50	50	50	20
	# of adversarial points	500	500	500	500	200
	Radius norm ($\|\cdot\|_p$)	inf	inf	inf	inf	inf
	Radius	$\{0.1, 0.2\}$	$\{0.1, 0.2\}$	$\{0.1, 0.2\}$	$\{0.1, 0.2\}$	$\{0.05, 0.1\}$
Optimization (PGD)	Optimizer	SGD				
	Num. steps	50				
	ε initialization	$\mathcal{N}(0, 0.2 \cdot I)$				
	Learning rate (lr)	1				
Defence (HMC)	Num. steps (T)	500	1000	1000	2000	1000
	Step size η	0.1	0.05	0.05	1e-4	1e-4
	Num. Leapfrog steps (L)	20	20	20	20	1
	Adaptive step size	True	True	True	True	False