Semantics Preserving Adversarial Examples

Sanjay Kariyappa
Georgia Institute of Technology
Atlanta GA, USA
sanjaykariyappa@gatech.edu

Ousmane Dia
Facebook
Menlo Park CA, USA
ousamdia@fb.com

Abstract

While progress has been made in crafting adversarial examples with visually imperceivable changes, constructing semantically meaningful ones remains a challenge. In this paper, we propose a framework to create semantics preserving adversarial examples. The motivating principle behind our proposal is that semantics are better captured in the low-level embedding space of the inputs than in the noisy high-level feature space. Our proposal uses a manifold learning method to first learn the low-dimensional geometric summaries of the inputs via statistical inference. Then, we perturb the elements of the learned manifold to create adversarial examples. To ensure that the semantics of the adversarial examples are preserved, we develop a manifold-invariant adversarial perturbation technique that induces the perturbed elements to remain in the manifold, while satisfying adversarial constraints. Our evaluations on toy data, images and text show that our proposed attack can produce adversarial examples that retain a greater degree of semantic similarity compared to existing attacks. Furthermore, our attack also has a high attack success rate against various certified and non-certified defenses.

1. Introduction

Deep learning models are fragile, as small, inconspicuous noise injected to the input data can cause a highly accurate model to suddenly make erroneous predictions [12, 34]. This problem known as adversarial examples has sparked keen research interests resulting in a number of approaches for attacking deep learning models [8, 23, 3, 26, 6]. In general, an adversarial attack is considered successful if the examples one crafts can mislead a model and are *perceptually similar* to or indistinguishable from the benign inputs. Existing attacks use ℓ_p norm distances to measure the degree of similarity between the adversarial example and its source input [3, 10, 6]. However, it is well documented that using *nearness* (in the input space) according to an ℓ_p norm does not guarantee semantic similarity [18, 32]. In the image domain specifically, perturbations generated by existing attacks do not always preserve the true nature of the image [32]; for instance *the resulting image is sometimes perceptually distorted, blurry, or unrealistic* as exemplified in Figure 1. In text, small changes to a vectorized input sentence can lead to large differences in meaning [38]. Thus, searching for adversarial examples in uniformly bounded regions [6, 22, 12, 34] using ℓ_p norms, or confining the adversarial examples to a simple latent structure (*e.g., using a Gaussian distribution*), are insufficient for creating adversarial examples that preserve the semantic content of the inputs.

(a) PGD [26] (b) [33]

Figure 1: The adversarial images in (a) and (b) are supposed to represent the digits 0 and 1, yet they look heavily distorted, blurry, and not quite legitimate 0s or 1s. Images from [33].

In this paper we introduce a method to address the limitations of previous approaches by constructing adversarial examples that explicitly preserve the semantics of the inputs. *Our premise is that semantics are better captured in the low-level feature space than in the noisy high-level input space. Specifically, if two input images or sentences are semantically similar in the low-level feature space, then they should be close to each other in the input space*. We achieve this by characterizing and aligning the low dimensional geometric summaries of the inputs and the adversarial examples. The summaries capture the semantics of the inputs and the adversarial examples. The alignment ensures that the adversarial examples reflect the semantics of the inputs. We explain the difference between existing attacks and our proposal using an illustration of the original (x) and adversarial examples (x') produced by these attacks in Figure 2. Existing attacks only consider ℓ_p norm constraint $\|x' - x\|_p \leq \epsilon_{\text{attack}}$ while generating adversarial examples. Consequently, the resulting adversarial examples can lie far from the data manifold (Figure 2a), leading to a loss of semantic similarity. In contrast, our proposed attack additionally ensures that the adversarial

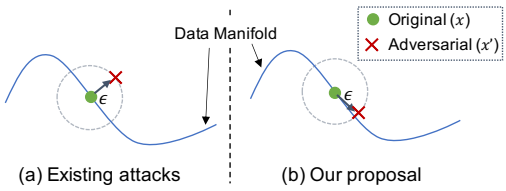


Figure 2: Existing attacks create adversarial examples that may not lie on the data manifold. In contrast, our proposed attack creates semantically similar adversarial examples that lie closer to the data manifold.

example lies closer to the data-manifold (Figure 2b) to preserve semantic similarity. The framework of our proposed attack can be decomposed into two key steps:

1. *Manifold Learning*: To capture the semantics of the input, we first learn the low dimensional geometric summaries of the inputs via statistical inference. We propose a manifold learning technique using Stein variational inference to learn the low-dimensional latent representation of the input.

2. *Semantics Preserving Adversarial Attack*: We propose a learning algorithm to perturb the latent-space representations of the inputs, which can be projected back to the input space to create semantically similar adversarial examples. The perturbation generated by our algorithm needs to satisfy two key requirements. First, we want the examples produced from the perturbed latent codes to lie close to the data manifold. We leverage the *manifold invariance concept* of [31] to generate the perturbations, which ensures that the resulting adversarial example preserves semantic similarity with the original input. Second, we want the perturbation to result in an adversarial example that is misclassified by the target model. This can be achieved by optimizing the perturbation using an adversarial objective.

To evaluate our approach, we test our method on toy data, images and text, and validate it empirically against strong certified and non-certified adversarial defenses. Our experiments show that our attack can produce adversarial examples with a high attack success rate and better semantic similarity compared to existing attacks.

2. Preliminaries

Notations. Let x be a sample from the input space $\mathcal{X}$, with label y from a set of possible labels $\mathcal{Y}$, and $\mathcal{D} = \{x_n\}_{n=1}^N$ a set of N such samples. Also, let d be a distance measure on $\mathcal{X}$ capturing closeness in input space, or on $\mathcal{Z}$, the embedding space of $\mathcal{X}$, capturing semantic similarity.

Standard $\ell_\mathbf{p}$-ball Threat Models. Given a classifier g_ν and its loss function ℓ, the goal of the attacker is to produce an adversarial example by maximizing the objective below over an ϵ_{attack}-radius ball around x [4, 3, 10, 6].

$$x' = \underset{x' \in \mathcal{X}}{\operatorname{argmax}}\, \ell(g_\nu(x'), y) \text{ such that } \|x' - x\|_\infty \leq \epsilon_{\text{attack}}.$$

Our Proposal. Above, the search region for adversarial examples is confined to a uniformly-bounded ball $\mathcal{B}(x; \epsilon_{\text{attack}})$. In reality, the constraints on the search space imposed by $\mathcal{B}$ is not a sufficient condition to ensure semantic similarity. Our main intuition in this paper is that the embedding space $\mathcal{Z}$ *better captures the semantics of* $\mathcal{D}$. By operating in the embedding space, we can craft adversarial examples with better semantic similarity compared to existing attacks.

In this paper we consider a *white-box* scenario, where we have perfect knowledge of the architecture of a classifier g_ν including its loss function and weights. As the attacker, we want to construct *semantics preserving adversarial examples* that fool g_ν. We evaluate also our attack framework against various certified and non-certified defenses.

3. Attack Framework

Given a sample $x \in \mathcal{D}$ and its class $y \in \mathcal{Y}$, our goal is to construct an adversarial example x' that shares the same semantic content as x. We assume the semantics of x (resp. x') is modeled by a learned latent variable model $p(z)$ (resp. $p'(z')$), where $z, z' \in \mathcal{Z}$. In this setting, observing x (resp. x') is conditioned on the observation model $p(x|z)$ (resp. $p(x'|z')$), that is: $x \sim p(x|z)$ and $x' \sim p(x'|z')$, with $z \sim p(z)$ and $z' \sim p'(z')$. To ensure that x' retains the semantics of the original x, we learn this model in a way that $d(z, z')$ is small and $g_\nu(x) \neq g_\nu(x')$. We illustrate our attack framework in Figure 3.

Our framework is essentially a variational auto-encoder (VAE) with a set of encoders E, which are used (i.) to learn the geometric summaries of $\mathcal{D}$ via *manifold learning* using *Stein variational gradient descent* (SVGD) [25], and (ii.) to perturb such summaries using *Gram-Schmidt basis sign method* [9] (please see Appendix A for background on VAE and SVGD). We define the encoder E as a set of M embedding maps $h_1, ..., h_M$, where $h_m \colon \mathcal{X} \to \mathcal{Z}$ for $m = 1, ..., M$, parameterized by $\Theta = \{\theta_m\}_{m=1}^M$ that generates a set of M latent codes $\{z_m\}_{m=1}^M$. From these latent codes, we compute $z'_m = z_m + \delta'_m$, the perturbed version of z_m, such that $d(z_m, z'_m)$ is small and z'_m lies in the manifold that supports $p(z_m)$. Note that a *manifold* here refers to a set of points in $\mathcal{Z}$ where every point is locally Euclidean [31]. We devise our perturbation procedure by generalizing the *manifold invariance concept* of [31] to the space $\mathcal{Z}$. We average the set of perturbed latent codes $\{z'_m\}$ to compute z'. Using the decoder $dec_\phi \colon \mathcal{Z} \to \mathcal{X}$, we craft $x' = dec_\phi(z')$ such that $g_\nu(x) = y$ and $g_\nu(x') \neq y$. Then, we say that x' is adversarial to x and preserves its semantics.

Efficiently Storing Encoder Parameters. For large M, maintaining Θ can be computationally prohibitive because of the large memory footprint. To sidestep this issue, we maintain only one (recognition) network f_η that takes as input $\xi_m \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and outputs a particle θ_m. We describe our manifold learning method using SVGD and the training

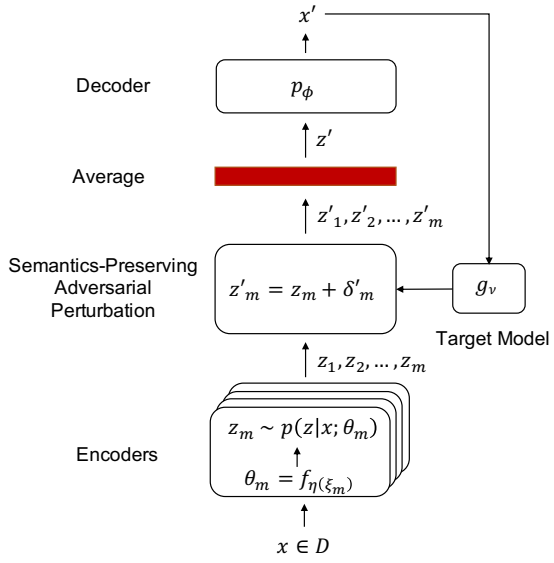


Figure 3: The set of encoder model parameters $\Theta = \{\theta_m\}_{m=1}^M$ are obtained from the recognition network f_η using inputs $\{\xi_m\}_{m=1}^M$. Given an input $x \in \mathcal{D}$, we use the encoders to sample the latent codes $z_1, ..., z_M$ via Θ. We learn semantics preserving adversarial perturbations to generate perturbed versions of the latent codes $z'_1, ..., z'_M$. The adversarial example $x' \sim p_\phi(x'|z')$ is generated via posterior sampling of the averaged latent code z'.

process for the recognition network f_η in Section 4.

4. Manifold Learning via SVGD

To characterize the manifold $\mathcal{M}$ of $\mathcal{D}$, we learn the encoding function $q(.;\psi)$ (as explained in Appendix A.1). Similar to [28], we optimize the divergence $\mathbb{KL}(q(z|x;\psi)\|p(z|x;\phi))$ using SVGD. Learning $q(.;\psi)$, however, induces inherent uncertainty we ought to capture in order to learn $\mathcal{M}$ efficiently. One way to capture such uncertainty is to use dropout [28]. However, Bayesian methods, provide a more principled way to model uncertainty through the posterior distribution over model parameters [15]. [19] have shown that SVGD can be cast as a Bayesian approach for parameter estimation and uncertainty quantification. As SVGD maintains M particles, we use each of the M instances of model parameters $\Theta = \{\theta_m\}_{m=1}^M$ to define the weights and biases of a Bayesian Neural network (BNN).

Encoder Training. We train each θ_m via SVGD using the operator $\tau(.)$ as described in Equation 1. In the following, we use the notation $\text{SVGD}_\tau(\Theta)$ to denote one SVGD training update of all the model parameters $\Theta = \{\theta_m\}_{m=1}^M$.

$$\theta_m^{t+1} \leftarrow \theta_m^t + \alpha_t \tau(\theta_m^t), \text{ where } \tau(\theta_m^t) \text{ is defined in Equation 6 with } z^t \text{ replaced by } \theta_m^t \text{ and } z_i^t \text{ by } \theta_i^t. \quad (1)$$

As each θ parameterizes a BNN, upon observing $\mathcal{D}$, we update the prior $p(\theta_j^t)$ to obtain the posterior $p(\theta_j^t|\mathcal{D}) \propto p(\mathcal{D}|\theta_j^t)p(\theta_j^t)$, which captures the uncertainty. We refer the reader to the Appendix D for a formulation of $p(\theta_j^t|\mathcal{D})$ and $p(\mathcal{D}|\theta_j^t)$. By definition, the likelihood $p(\mathcal{D}|\theta_j^t)$ is evaluated over all pairs $(x, \tilde{z})$ where $x \in \mathcal{D}$ and $\tilde{z}$ is a dependent variable. However, since $\tilde{z}$ is not given, we introduce the *inversion process* in Appendix B (Figure 4) to generate such $\tilde{z}$ using Algorithm 1. Given $x \in \mathcal{D}$, we sample its latent code z from $p(z|x;\mathcal{D})$ using Monte Carlo:

$$\begin{aligned} p(z|x;\mathcal{D}) &= \int p(z|x;\theta)p(\theta|\mathcal{D})\mathrm{d}z \\ &\approx \frac{1}{M}\sum_{m=1}^{M} p(z|x;\theta_m) \text{ with } \theta_m \sim p(\theta|\mathcal{D}). \end{aligned} \quad (2)$$

Recognition Network Training. As mentioned before, we introduce the recognition network f_η to minimize the memory footprint of storing Θ. f_η learns the trajectories of the model parameters θ_m as they get updated via SVGD. The recognition network f_η serves as a proxy to SVGD sampling strategy, and is refined through a small number of gradient steps as shown in Equation 3 to get good generalization.

$$\eta^{t+1} \leftarrow \underset{\eta}{\text{argmin}} \sum_{m=1}^{M} \left\| \theta_m^{t+1} - \underbrace{f(\xi_m;\eta^t)}_{\theta_m^t} \right\|_2 \quad (3)$$

In Section 5, we describe our approach to generate semantics-preserving adversarial perturbations.

5. Semantics Preserving Adversarial Attack

Our goal is to create semantics preserving adversarial examples by perturbing the latent representation of the inputs, which can then be used by the decoder to generate adversarial examples (x'). We want the adversarial example generated from the perturbation to satisfy two properties:
(i.) *Semantics preservation: x' needs to be semantically similar to x, in addition to satisfying the $\|.\|_\infty$ constraint.*
(ii.) *Misclassification: x' should cause misclassification in the target model; i.e. $g_\nu(x') \neq g_\nu(x)$.* In this section, we explain how manifold invariant localized perturbations can be used to create semantics preserving adversarial examples.

5.1. Semantics Preserving Localized Perturbations

We want the perturbed elements to reside in $\mathcal{M}$ and exhibit the semantics of $\mathcal{D}$ that $\mathcal{M}$ captures. Formally, we seek an affine mapping $h'\colon \mathcal{M} \to \mathcal{M}$ such that for any point $z \in \mathcal{M}$, a neighborhood $\mathcal{U}$ of z is *invariant* under h': $z' \in \mathcal{U} \Rightarrow h'(z') \in \mathcal{U}$. Then, we say that $\mathcal{M}$ is *preserved under* h'. We leverage the local smoothness of $\mathcal{M}$ to learn each δ'_m in a way to encourage z' to *reside in $\mathcal{M}$ in a close neighborhood of z* using a technique called Gram-Schmidt Basis Sign Method which we describe next.

Gram-Schmidt Basis Sign Method (GBSM). Let $\mathbf{X}$ be a batch of samples of $\mathcal{D}$, $\mathbf{Z}_m$ a set of latent codes $z_m \sim$

$p(z|x;\theta_m)$ where $x \in \mathbf{X}$, and $\theta_m \in \Theta$. The intuition behind GBSM is to utilize the fact that topological spaces are closed under their basis vectors to render $\mathcal{M}$ invariant to the perturbations δ'_m. As $\mathcal{M}$ is locally Euclidean, we compute the dimensions of the subspace $\mathbf{Z}_m$ by applying Gram-Schmidt [9] to orthogonalize the span of representative local points and find its basis vectors $u_{im} \in \mathbf{U}_m$. For any $z_m \in \mathbf{Z}_m$, we learn to generate its perturbed version z'_m along the directions of an orthonormal basis $\mathbf{U}_m$. We want the perturbations δ_m to be small. Thus, we minimize:

$$\min_{\delta_m} z'_m := z_m + \delta_m \odot \text{sign}(u_{im}) \, \forall \, m \text{ s.t } \|x' - x\|_\infty < \epsilon_{\text{attack}}$$
$$\text{where } x' \sim p(x'|z';\phi) \text{ and } z' = \frac{1}{M}\sum_{m=1}^{M} z'_m. \tag{4}$$

We incorporate the constraint on x' by using $\|x' - x\|_2$ as a regularization term in the training loss. Since $\|.\|_\infty \leq \|.\|_2$ (see proof in Appendix E), we can satisfy the $\|.\|_\infty$ constraint by ensuring $\|x' - x\|_2 \leq \epsilon_{\text{attack}}$ during training.

5.2. Generating Adversarial Examples

We create adversarial examples by optimizing the log-likelihood $\mathcal{L}_{x'}$ of $y' \in \mathcal{Y} \setminus \{y\}$ where y is the class of x.

$$\mathcal{L}_{x'} = \max_{y' \in \mathcal{Y} \setminus \{y\}} \log P(y'|x';\nu). \tag{5}$$

$\mathcal{L}_{x'}$ defines the cost incurred for failing to fool g_ν. In Algorithm 2 (Appendix B), we unify our manifold learning and perturbation strategy to create adversarial examples.

6. Experiments & Results

We validate our *white-box* and *non-targeted* attack model based on three criteria: (i.) *adversarial strength*, (ii.) *soundness* via manual evaluation, and (iii.) *manifold preservation*. We refer the reader to Appendix C.1 for a study of manifold preservation using the Swiss Roll dataset.

Setup. We evaluate the strength of our MNIST, CelebA, CIFAR10, and SVHN adversaries against adversarially trained ResNets [13] with a 40-step Projected Gradient Descent (PGD) [12, 3] and noise margin $\epsilon_{\text{attack}} \leq 0.3$. We use similar ResNet models as [33]. For MNIST, we also target the certified defenses [30] and [21] with $\epsilon_{\text{attack}} = 0.1$. The accuracies of all the models we target are higher than 96.3%. Additionally, we also evaluate our attack on a text classification problem using the SNLI dataset [5] in Appendix C.2.

6.1. Adversarial Strength

Attack Success Rate (ASR) is the percentage of examples misclassified by the adversarially trained Resnet models. Our ASR for MNIST is 97.2%. Also, with $\epsilon_{\text{attack}} \approx 1.2$, we achieve an ASR of 97.6% against [21]. Finally, we achieve an ASR of 87.6% for SVHN, and 84.4% for CelebA. We give examples of adversarial images we generate in Appendix G.

6.2. Manual Evaluation

To assess the semantic soundness of the adversarial examples, we carry out a pilot study by asking three yes-or-no questions: (Q1) *are the adversarial examples semantically sound?*, (Q2) *are the adversarial inputs similar perceptually or in meaning to the corresponding true inputs?* and (Q3) *are there any interpretable visual cues in the adversarial images that support their misclassification?* We ask the human subjects to assess the soundness of the adversarial examples based on the *semantic features* (*shape, distortion, contours, class, image quality, brightness*) of the source images.

Pilot Study I. In this section, we compare the semantic soundess of the adversarial examples produced by our attack with those generated with existing attacks.To evaluate semantic similarity, we perform manual human evaluations on the adversarial examples produced by [33], [38] and PGD attacks. Our results in Table 1 show that our attack produces adversarial examples with better semantic properties compared to existing attacks. Additionally, we also report the semantic soundness of adversarial examples produced by our attack against various defenses in Appendix C.3.

Table 1: Pilot Study I. Manual evaluation results for M1 (Our Method), M2 ([33]), M3 ([38]), and M4 (PGD).

Questions	MNIST			
	M1	M2	M3	M4
Q1: Yes	100 %	85.7 %	97.9 %	75.2 %
Q2: Yes	100 %	79.1 %	91.2 %	65.6 %
Q3: No	100 %	71.6 %	94.8 %	42.1 %

6.3. Key Takeaways

As reflected in the pilot study and the attack success rates, we achieve good results in the image and text classification tasks both against the certified and non-certified defenses. Although the defenses are resilient to adversarial examples crafted in the input space, we achieve nonetheless manual success rates higher than the rates certified for when the examples are constructed in the latent space within bounded search regions. In text, we achieve better results than [38].

7. Conclusion

Many approaches in adversarial attacks do not generate adversarial examples that preserve the semantics of the inputs. We have presented a method to generate adversarial examples that maintain similar semantics as the source inputs by conducting the search for adversarial examples in the manifold of the inputs. Our evaluations on both image and text domain datasets show that our attack produces adversarial attacks with a high attack success rate and better semantic similarity compared to several existing attacks.

References

[1] Alexander A. Alemi, Ben Poole, Ian Fischer, Joshua V. Dillon, Rif A. Saurous, and Kevin Murphy. An information-theoretic analysis of deep latent-variable models. *CoRR*, abs/1711.00464, 2017. 17

[2] David Alvarez-Melis and Tommi S. Jaakkola. A causal framework for explaining the predictions of black-box sequence-to-sequence models. *CoRR*, abs/1707.01943, 2017. 18

[3] Anish Athalye, Nicholas Carlini, and David A. Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *CoRR*, abs/1802.00420, 2018. 1, 2, 4, 18

[4] Anish Athalye, Logan Engstrom, Andrew Ilyas, and Kevin Kwok. Synthesizing robust adversarial examples. *CoRR*, abs/1707.07397, 2017. 2, 18

[5] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. *CoRR*, abs/1508.05326, 2015. 4, 8

[6] Nicholas Carlini and David A. Wagner. Towards evaluating the robustness of neural networks. *CoRR*, abs/1608.04644, 2016. 1, 2, 18

[7] Liqun Chen, Shuyang Dai, Yunchen Pu, Chunyuan Li, Qinliang Su, and Lawrence Carin. Symmetric Variational Autoencoder and Connections to Adversarial Learning. *arXiv e-prints*, page arXiv:1709.01846, Sep 2017. 17

[8] Francesco Croce and Matthias Hein. Minimally distorted adversarial examples with a fast adaptive boundary attack. *CoRR*, abs/1907.02044, 2019. 1

[9] Kimberly A. Dukes. *Gram–Schmidt Process*. American Cancer Society, 2014. 2, 4

[10] Logan Engstrom, Dimitris Tsipras, Ludwig Schmidt, and Aleksander Madry. A rotation and a translation suffice: Fooling cnns with simple transformations. *CoRR*, abs/1712.02779, 2017. 1, 2

[11] Luca Falorsi, Pim de Haan, Tim R. Davidson, Nicola De Cao, Maurice Weiler, Patrick Forré, and Taco S. Cohen. Explorations in Homeomorphic Variational Auto-Encoding. *arXiv e-prints*, page arXiv:1807.04689, Jul 2018. 17

[12] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harvesting adversarial examples. *CoRR abs/1412.6572*, 2014. 1, 4, 18

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. 4

[14] Irina Higgins, Loic Matthey, Xavier Glorot, Arka Pal, Benigno Uria, Charles Blundell, Shakir Mohamed, and Alexander Lerchner. Early Visual Concept Learning with Unsupervised Deep Learning. *arXiv e-prints*, page arXiv:1606.05579, Jun 2016. 17

[15] Jiri Hron, Alexander G. de G. Matthews, and Zoubin Ghahramani. Variational Gaussian Dropout is not Bayesian. *arXiv e-prints*, page arXiv:1711.02989, Nov 2017. 3, 17

[16] Robin Jia and Percy Liang. Adversarial examples for evaluating reading comprehension systems. *CoRR*, abs/1707.07328, 2017. 18

[17] Danilo Jimenez Rezende and Shakir Mohamed. Variational Inference with Normalizing Flows. *arXiv e-prints*, page arXiv:1505.05770, May 2015. 7, 10, 17

[18] Matt Jordan, Naren Manoj, Surbhi Goel, and Alexand ros G. Dimakis. Quantifying Perceptual Distortion of Adversarial Examples. *arXiv e-prints*, page arXiv:1902.08265, Feb. 2019. 1

[19] Taesup Kim, Jaesik Yoon, Ousmane Dia, Sungwoong Kim, Yoshua Bengio, and Sungjin Ahn. Bayesian model-agnostic meta-learning. *CoRR*, abs/1806.03836, 2018. 3, 10

[20] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *International Conference on Learning Representations (ICLR)*, 2014. 7, 17

[21] J. Zico Kolter and Eric Wong. Provable defenses against adversarial examples via the convex outer adversarial polytope. *CoRR*, abs/1711.00851, 2017. 4, 10

[22] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *CoRR*, abs/1607.02533, 2016. 1, 14, 18

[23] Cassidy Laidlaw and Soheil Feizi. Functional adversarial attacks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 10408–10418. Curran Associates, Inc., 2019. 1

[24] Jiwei Li, Will Monroe, and Dan Jurafsky. Understanding neural networks through representation erasure. *CoRR*, abs/1612.08220, 2016. 18

[25] Qiang Liu and Dilin Wang. Stein variational gradient descent: A general purpose Bayesian inference algorithm. *Neural Information Processing Systems (NIPS)*, 2016. 2, 7

[26] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *CoRR*, abs/1706.06083, 2017. 1

[27] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. *CoRR*, abs/1610.08401, 2016. 18

[28] Yunchen Pu, Zhe Gan, Ricardo Henao, Chunyuan Li, Shaobo Han, and Lawrence Carin. VAE learning via Stein variational gradient descent. *Neural Information Processing Systems (NIPS)*, 2017. 3, 17

[29] Alec Radford. Improving language understanding by generative pre-training. In *arXiv*, 2018. 11

[30] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified defenses against adversarial examples. *CoRR*, abs/1801.09344, 2018. 4, 10

[31] Marc R Roussel. Invariant manifolds. In *Nonlinear Dynamics*, 2053-2571, pages 6–1 to 6–20. Morgan & Claypool Publishers, 2019. 2

[32] Mahmood Sharif, Lujo Bauer, and Michael K. Reiter. On the suitability of lp-norms for creating and preventing adversarial examples. *CoRR*, abs/1802.09653, 2018. 1

[33] Yang Song, Rui Shu, Nate Kushman, and Stefano Ermon. Constructing unrestricted adversarial examples with generative models. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada.*, pages 8322–8333, 2018. 1, 4, 18

[34] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014. 1

[35] Bing Yu, Jingfeng Wu, and Zhanxing Zhu. Tangent-normal adversarial regularization for semi-supervised learning. *CoRR*, abs/1808.06088, 2018. 17

[36] Junbo Zhao, Yoon Kim, Kelly Zhang, Alexander Rush, and Yann LeCun. Adversarially regularized autoencoders. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5902–5911, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR. 8, 11, 18

[37] Shengjia Zhao, Jiaming Song, and Stefano Ermon. InfoVAE: Information maximizing variational autoencoders. *CoRR*, abs/1706.02262, 2017. 7, 10, 17

[38] Zhengli Zhao, Dheeru Dua, and Sameer Singh. Generating natural adversarial examples. In *International Conference on Learning Representations*, 2018. 1, 4, 8, 9, 11, 18

A. Background on Manifold Learning

Our proposed attack uses manifold learning to first map the inputs to a low-dimensional embedding space and then uses this embedding space representation to craft an adversarial example. Manifold learning is based on the assumption that high dimensional data lies on or near lower dimensional manifolds in a latent space. Our paper uses a Variational Auto-Encoder (VAE) trained using Stein Variation Gradient Descent (SVGD) for manifold learning. We provide background information on VAEs and SVGD in this section before describing our attack framework in section 3.

A.1. Variational Auto Encoders

VAEs [20] model the datapoints $x_n \in \mathcal{D}$ using a decoder $x_n \sim p(x_n|z_n;\phi)$. To learn ϕ, one typically maximizes a variational approximation to the empirical expected log-likelihood $1/N \sum_{n=1}^{N} \log p(x_n;\phi)$, called evidence lower bound (ELBO), defined as:

$$\begin{aligned}\mathcal{L}_{\mathrm{e}}(\phi,\psi;x) &= \mathbb{E}_{q(z|x;\psi)} \log \left[\frac{p(x|z;\phi)p(z)}{q(z|x;\psi)}\right] \\ &= -\mathbb{KL}(q(z|x;\psi)\|p(z|x;\phi)) + \log p(x;\phi).\end{aligned}$$

Here, $q(z|x;\psi)$ denotes the encoder function. The expectation $\mathbb{E}_{q(z|x;\psi)}$ can be re-expressed as a sum of a reconstruction loss, or expected negative log-likelihood of x, and $\mathbb{KL}(q(z|x;)\|p(z))$. The $\mathbb{KL}$ forces the encoder q_ψ to follow a distribution similar to $p(z)$. VAEs learn an encoding function that maps the data manifold to an isotropic Gaussian. However, [17] have shown that the Gaussian form imposed on $p(z)$ may result in uninformative latent codes; *hence to poorly learning the semantics of* $\mathcal{D}$ [37]. To sidestep this issue, we minimize the divergence $\mathbb{KL}(q(z|x;\psi)\|p(z|x;\phi))$ using Stein Variational Gradient Descent [25] instead of explicitly optimizing the ELBO.

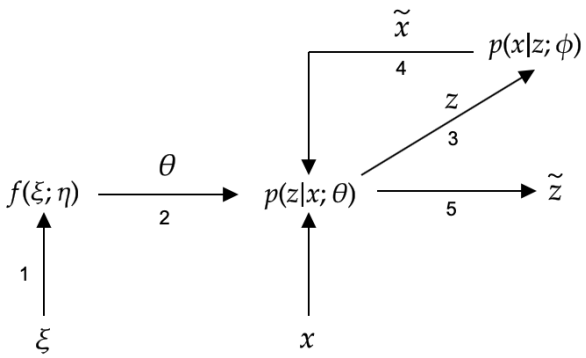
A.2. Stein Variational Gradient Descent

SVGD is a nonparametric variational inference method that combines the advantages of MCMC sampling and variational inference. Unlike ELBO [20], SVGD does not confine a target distribution $p(z)$ it approximates to simple or tractable parametric distributions. It remains an efficient algorithm. To approximate $p(z)$, SVGD maintains M particles $\mathbf{z} = \{z_i\}_{i=1}^{M}$, initially sampled from a simple distribution, it iteratively transports via functional gradient descent. At iteration t, each particle $z^t \in \mathbf{z}^t$ is updated as:

$$\begin{gathered}z^{t+1} \leftarrow z^t + \alpha^t \tau(z^t) \text{ where} \\ \tau(z^t) = \frac{1}{M}\sum_{i=1}^{M}\Big[k(z_i^t, z^t)\nabla_{z_i^t}\log p(z_i^t) + \nabla_{z_i^t} k(z_i^t, z^t)\Big],\end{gathered} \tag{6}$$

where α_t is a step-size and $k(.,.)$ is a positive-definite kernel. In Equation 6, each particle determines its update direction by consulting with other particles and asking their gradients. The importance of the latter particles is weighted according to the distance measure $k(.,.)$. Closer particles are given higher consideration than those lying further away. The term $\nabla_{z_i} k(z_i, z)$ is a regularizer that acts as a repulsive force between the particles to prevent them from collapsing into one particle. Upon convergence, the particles z_m will be unbiased samples of the true distribution $p(z)$.

B. Algorithms for Our Proposed Attack



Algorithm 1 Inversion with one particle $\boldsymbol{\theta}$.

Require: Input $x \in \mathcal{D}$
Require: Model parameters $\boldsymbol{\eta}$
1: Sample $\boldsymbol{\xi} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: Get the weight vector $\boldsymbol{\theta} = \boldsymbol{f_\eta}(\boldsymbol{\xi})$
3: Given x, sample $z \sim p(z|x;\boldsymbol{\theta})$
4: Sample $\tilde{x} \sim p(x|z,\boldsymbol{\phi})$
5: Sample $\tilde{z} \sim p(z|\tilde{x},\boldsymbol{\theta})$
6: Use x and $\tilde{z}$ to compute $p(\tilde{z}|x;\boldsymbol{\theta})$

Figure 4: As the decoder p_ϕ gets more accurate (reconstruction loss $\|x - \tilde{x}\|_2$ becomes small), we get closer to the optimal $\tilde{z}$.

Algorithm 2 *Adversarial Examples*. Lines 7 computes distances between sets keeping a one-to-one mapping.

Require: Training samples $(x, y) \in \mathcal{D} \times \mathcal{Y}$
Require: Number of model instances M and inner updates T
Require: Initialize weights $\boldsymbol{\eta}, \boldsymbol{\phi}$ ▷ recognition net f_η, decoder p_ϕ
Require: Learning rates $\boldsymbol{\alpha}, \boldsymbol{\beta}$
1: Sample $\boldsymbol{\xi_1}, ..., \boldsymbol{\xi_M}$ from $\mathcal{N}(\mathbf{0}, \mathbf{I})$ ▷ inputs to recognition net f_η
2: **for** $t = 1$ **to** T **do**
3: Sample $\boldsymbol{\Theta} = \{\boldsymbol{\theta_m}\}_{m=1}^M$ where $\boldsymbol{\theta_m} = \boldsymbol{f_\eta}(\boldsymbol{\xi_m})$
4: Sample $z_1, ..., z_M$ using $\boldsymbol{\Theta}$ in Equation 2
5: Using Equation 4, get $z'_1, .., z'_M$ and average them to get z' ▷ learn latent perturbations $\delta_1, ..., \delta_M$
6: Sample $\tilde{x} \sim p(x|z,\boldsymbol{\phi})$ and $x' \sim p(x'|z',\boldsymbol{\phi})$ ▷ clean and perturbed reconstructions
7: $\boldsymbol{\eta} \leftarrow \boldsymbol{\eta} - \boldsymbol{\alpha}\nabla_{\boldsymbol{\eta}}\|\boldsymbol{\Theta} - \texttt{SVGD}_\tau(\boldsymbol{\Theta})\|_2$ (requires $\tilde{x}$) ▷ apply *inversion* on $\tilde{x}$ and update η
8: $\boldsymbol{\mathcal{L}}_{\tilde{x}} := \|x - \tilde{x}\|_2$; $\boldsymbol{\mathcal{L}}_{x'} := -\min_{y' \in \mathcal{Y} \setminus \{y\}} \log\left(P(y'|x';\nu)\right) + \lambda * \|x - x'\|_2$ ▷ scaling reconstruction loss on x' by λ
9: $\boldsymbol{\phi} \leftarrow \boldsymbol{\phi} - \boldsymbol{\beta}\nabla_{\boldsymbol{\phi}}(\boldsymbol{\mathcal{L}}_{\tilde{x}} + \boldsymbol{\mathcal{L}}_{x'})$ ▷ decoder update using Adam optimizer

C. Additional Experiments

C.1. Manifold Preservation

We experiment with a 3D non-linear Swiss Roll dataset comprising of 1600 datapoints grouped in 4 classes. Figure 5 shows the 2D plots of the latent codes from the learnt manifold (left), latent codes of adversarial examples (with $\epsilon_{\text{attack}} \leq 0.3$) produced by our attack (center) and latent codes of PGD adversarial examples (right). These plots show that, unlike PGD, the latent codes of our adversarial examples are well-aligned with the manifold.

C.2. Evaluations on Text Classification Task using SNLI Dataset

We consider the SNLI [5] dataset. SNLI consists of sentence pairs where each pair contains a premise (*P*) and a hypothesis (*H*), and a label indicating the relationship (*entailment, neutral, contradiction*) between the premise and hypothesis. For instance, the following pair is assigned the label *entailment* to indicate that the premise entails the hypothesis.
Premise: A soccer game with multiple males playing. Hypothesis: Some men are playing a sport.

Setup. We perturb the hypotheses while keeping the premises unchanged. Similar to [38], we generate adversarial text at word level using a vocabulary of 11,000 words. We also use ARAE [36] for word embedding, and a CNN for sentence embedding.

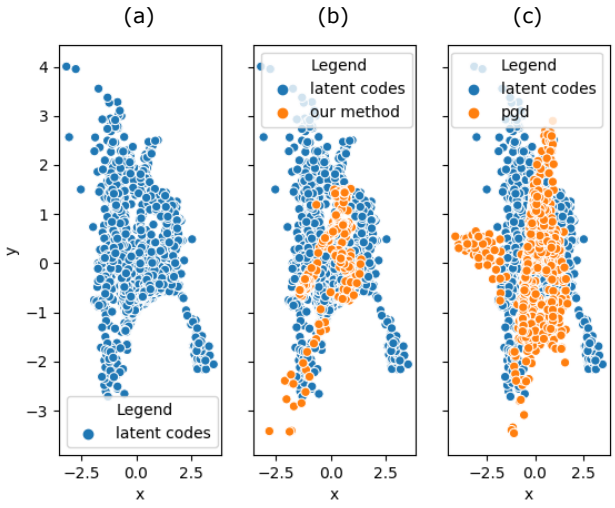


Figure 5: Swiss Roll manifold learned with encoder E (left), and after perturbing its elements with GBSM (middle) vs. that of PGD adversarial examples (right) learned using E.

To generate perturbed hypotheses, we consider three types of decoders p_ϕ: (i.) a transpose CNN, (ii.) a language model, and (iii.) we use the decoder of a pre-trained ARAE model. The transpose CNN generates more meaningful hypotheses (see examples in Table 8) than the language model and the pre-trained ARAE model although we notice sometimes changes in the meaning of the original hypotheses. We discuss these limitations in Appendix where we provide also more examples of adversarial text.

Attack Success Rate (ASR). We attack an SNLI classifier that has a test accuracy of 89.42%. Given a pair (*P, H*) with label l, its perturbed version (*P, H'*) is adversarial if the classifier assigns the label l to (*P, H*), (*P, H'*) is manually found to retain the label of (*P, H*), and such label differs from the one the classifier assigns to (*P, H'*).

Unlike for adversarial images, in order to compute the ASR for our text adversaries, we rely on human evaluation. The reason is that the sentence pair (*P, H'*) we adversarially generate *might not be adversarial to a human* since the new class the target model assigns to (*P, H'*) may actually reflect the true relationship between *P* and *H'*. Consequently, we run a pilot study which we detail in Section 7.3.

Pilot Study II - SNLI. Using the transpose CNN as decoder p_ϕ, we generate adversarial hypotheses for the SNLI sentence pairs with the premises kept unchanged. Then, we manually select 100 pairs of clean sentences (premise, hypothesis), and adversarial hypotheses. We also pick 100 pairs of sentences and adversarial hypotheses generated using [38]'s method against their treeLSTM classifier. We choose this classifier as its accuracy (89.04%) is close to ours (89.42%). Finally, to quantify the percentages of coherent text adversaries we generate, we randomly pick 100 pairs of sentences, and 100 pairs of sentences generated using [38]'s treeLSTM to carry out a pilot study where we ask three questions (Q1) *are the adversarial samples semantically sound?*, (Q2) *are they similar to the true inputs?* and (Q3) *what is the percentage of examples that are adversarial and semantically sound out of the random samples we select?* We report the evaluation results in Table 2.

Table 2: Pilot Study II. Manual evaluation results for SNLI.

QUESTIONS	OUR METHOD	[38]
Q1: YES	82.9 %	78.8 %
Q2: YES	61.2 %	55.9 %
Q3 (PCT.)	60.3 %	53.7 %

C.3. Evaluation against Defenses

We study the semantic soundness of the adversarial examples produced by our attack by attacking various defenses. For MNIST, we pick 100 images (10 for each digit) and generate adversarial examples against a 40-step PGD ResNet (M1) with

$\epsilon_{\text{attack}} \leq 0.3$. We also target the certified defenses of [30] (M2) and [21] (M3) with $\epsilon_{\text{attack}} = 0.1$. We repeat this study for the SVHN and CelebA datasets (for gender classification) by attacking a 40-step PGD ResNet. For all three datasets, we hand the images and the questionnaire (Q1, Q2, Q3) to 10 human subjects for manual evaluation. We report the results for MNIST in Table 3, and the results for CelebA and SVHN in Table 4. Our results show that our attack can produce semantics preserving adversarial examples with a high success rate against various defenses as well as different target datasets.

Table 3: Pilot Study I. Manual evaluation results of our attacks against M1 (40-step PGD), M2 ([30]) and M3 ([21]).

QUESTIONS	MNIST		
	M1	M2	M3
Q1: YES	100 %	100 %	100 %
Q2: YES	100 %	100 %	100 %
Q3: NO	100 %	100 %	100 %

Table 4: Pilot Study I. Our results for CelebA and SVHN.

QUESTIONS	CELEBA	SVHN
Q1: YES	100 %	94.3 %
Q2: YES	100 %	96.8 %
Q3: NO	100 %	100 %

D. Posterior Formulation

Similar to [19], we formalize $p(\theta|\mathcal{D})$ for every $\theta \in \Theta$ as:

$$p(\theta|\mathcal{D}) \propto p(\mathcal{D}|\theta)p(\theta) = \prod_{(x,\tilde{z})} p(\tilde{z}|x;\theta)p(\theta) \text{ where } x \in \mathcal{D} \text{ and } \tilde{z} \text{ is generated using Algorithm 1}$$
$$= \prod_{(x,\tilde{z})} \mathcal{N}(\tilde{z}|f_W(x), \gamma^{-1})\mathcal{N}(W|f_\eta(\xi), \lambda^{-1})\text{Gamma}(\gamma|a,b)\text{Gamma}(\lambda|a',b').$$

Note that θ consists in fact of network parameters $W \sim f_\eta$ and scaling parameters γ and λ. For notational simplicity, we used before the shorthands $\theta \sim f_\eta$. The parameters γ and λ are initially sampled from a Gamma distribution and updated as part of the training. In our experiments, we set the hyper-parameters of the Gamma distributions a and b to 1.0 and 0.1, and a' and b' to 1.0.

E. ℓ_2-norm an Upper Bound of ℓ_∞-norm

Given $x = (x_1, ..., x_n) \in \mathbb{R}^n$, $\|x\|_\infty = \max_i |x_i|$ and $\|x\|_2 = \sqrt{\sum_{i=1}^n x_i^2}$. If we let $|x_j| \coloneqq \max_i |x_i|$, given that $|x_j|^2 = x_j^2 \leq \sum_{i=1}^n x_i^2$, it follows that $\|x\|_\infty \leq \|x\|_2$. Hence, if $\|x\|_2 \leq \epsilon_{\text{attack}}$, then we have $\|x\|_\infty \leq \epsilon_{\text{attack}}$.

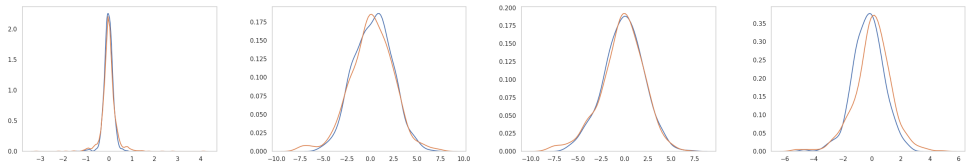
F. Discussion

Here, we discuss the choices pertaining to the design of our approach and their limitations.

Recognition Network. As noted in [17], the Gaussian prior assumption in VAEs is too restrictive to generate meaningful enough latent codes [37]. Thus, to produce informative latent codes, we use SVGD to learn the parameters of the encoder E. SVGD maintains a set of M model instances. As an ensemble method, SVGD inherits the shortcomings of ensemble models most notably in space/time complexity for large M. Thus, instead of maintaining $2 * M$ model instances, we maintain only one recognition network f_η which learns to mimic the sampling dynamics of SVGD although it diverges a bit sometimes.

Latent Noise Level. The changes to the original inputs we perturb are captured by our reconstruction loss — bounded by ϵ_{attack} (see Equation 5) — which measures the imperceptibility of our adversarial perturbations in the input space. To

get a sense of the amount of latent noise we inject to the clean codes $z_1, ..., z_M$, we compute the marginals of the clean and perturbed latent codes. As shown in Figure 6, the marginal distributions overlap relatively well. This means that the latent noise level is actually small.



(a) MNIST (b) CelebA (c) SVHN (d) SNLI

Figure 6: Marginal distributions of clean (blue) and perturbed (red) latent codes over few minibatches.

Semantics Preservation (Text). To construct adversarial text, we experiment with three architecture designs for the decoder p_ϕ: (i.) a transpose CNN, (ii.) a language model, and (iii.) the decoder of a pre-trained ARAE model [36]. The transpose CNN generates more legible text than the other two designs although we notice sometimes some changes in meaning in the generated adversarial examples. Similar to [38], a sizeable number of the examples we generate are not adversarial (see Table 9 for some examples). Adversarial text generation is challenging in that small perturbations in the latent codes can go unnoticed at generation whereas high noise levels can render the outputs nonsensical. To produce adversarial sentences that faithfully preserve the meaning of the inputs, we need good sentence generators, like GPT [29], trained on large corpora. In our experiments, we consider only a vocabulary of size 10,000 words and sentences of length no more than 10 words to align our evaluation with the experimental choices of [38].

G. Adversarial Examples

G.1. Adversarial Images: CelebA

Here, we provide few random samples of non-targeted adversarial examples we generate with our approach on the CelebA dataset as well as the clean reconstructions.

Table 5: CelebA samples, their clean reconstructions, and adversarial examples.

INPUTS		
CLEAN RECON-STRUCTIONS		
ADVERSARIAL EXAMPLES		

G.2. Adversarial Images: SVHN

Here, we provide few random samples of non-targeted adversarial examples we generate with our approach on the SVHN dataset as well as the clean reconstructions.

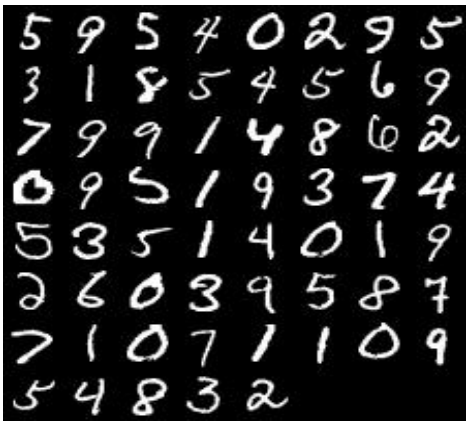
Table 6: SVHN. Images in red boxes are all adversarial.

INPUTS		
CLEAN RECON-STRUCTIONS		
ADVERSARIAL EXAMPLES		

G.3. Adversarial Images: MNIST

Here, we provide few random samples of non-targeted adversarial examples we generate with our approach on the MNIST dataset as well as the clean reconstructions. Both the reconstructed and the adversarial images look realistic and semantically correct although we notice some artifacts on the latter. Basic Iterative Methods [22], among other adversarial attacks, also suffer from this. Note that, however, in our case the marginal distributions of the latent codes of the inputs and their perturbed versions overlap quite well as illustrated in Figure 6.

Table 7: MNIST. Images in red boxes are all adversarial.

INPUTS		
CLEAN RECON-STRUCTION		
ADVERSARIAL EXAMPLES		

G.4. Adversarial Text: SNLI

Table 8: Examples of adversarially generated hypotheses with the true premises kept unchanged.

TRUE INPUT 1	*P*: A white dog is running through the snow. ***H*: A CAT STALKING THROUGH THE SNOW.** *Label*: CONTRADICTION
ADVERSARY	***H'*: A CAT HOPS IN THE SNOW.** *Label*: NEUTRAL
TRUE INPUT 2	*P*: Three dogs are searching for something outside. ***H*: THERE ARE FOUR DOGS.** *Label*: CONTRADICTION
ADVERSARY	***H'*: THERE ARE FIVE DOGS.** *Label*: NEUTRAL
TRUE INPUT 3	*P*: A man waterskis while attached to a parachute. ***H*: A BULLDOZER KNOCKS DOWN A HOUSE.** *Label*: CONTRADICTION
ADVERSARY	***H'*: A BULLDOZER KNOCKS DOWN A CAGE. *Label*: ENTAILMENT**
TRUE INPUT 4	*P*: A little girl playing with flowers. ***H*: A LITTLE GIRL PLAYING WITH A BALL.** *Label*: CONTRADICTION
ADVERSARY	***H'*: A LITTLE GIRL RUNNING WITH A BALL. *Label*: NEUTRAL**
TRUE INPUT 5	*P*: People stand in front of a chalkboard. ***H*: PEOPLE STAND OUTSIDE A PHOTOGRAPHY STORE.** *Label*: CONTRADICTION
ADVERSARY	***H'*: PEOPLE STAND IN FRONT OF A WORKSHOP. *Label*: NEUTRAL**
TRUE INPUT 6	*P*: Musician entertaining **his** audience. ***H*: THE WOMAN PLAYED THE TRUMPET.** *Label*: CONTRADICTION
ADVERSARY	***H'*: THE WOMAN PLAYED THE DRUMS. *Label*: ENTAILMENT**
TRUE INPUT 7	*P*: A kid on a slip and slide. ***H*: A SMALL CHILD IS INSIDE EATING THEIR DINNER.** *Label*: CONTRADICTION
ADVERSARY	***H'*: A SMALL CHILD IS EATING THEIR DINNER. *Label*: ENTAILMENT**
TRUE INPUT 8	*P*: A deer jumping over a fence. ***H*: A DEER LAYING IN THE GRASS.** *Label*: CONTRADICTION
ADVERSARY	***H'*: A PONY LAYING IN THE GRASS.** ***Label*: ENTAILMENT**
TRUE INPUT 9	*P*: Two vendors are on a curb selling balloons. ***H*: THREE PEOPLE SELL LEMONADE BY THE ROAD SIDE.** *Label*: CONTRADICTION
ADVERSARY	***H'*: THREE PEOPLE SELL ARTWORK BY THE ROAD SIDE.** ***Label*: ENTAILMENT**

Table 9: Misses. Some generated examples deemed adversarial by our method that are not.

TRUE INPUT 1	*P*: A man is operating some type of a vessel. ***H*: A DOG IN KENNEL.** *Label*: CONTRADICTION
GENERATED	***H'*: A DOG IN DISGUISE.** *Label*: CONTRADICTION
TRUE INPUT 2	*P*: A skier. ***H*: SOMEONE IS SKIING.** *Label*: ENTAILMENT
GENERATED	***H'*: MAN IS SKIING.** *Label*: NEUTRAL
TRUE INPUT 3	*P*: This is a bustling city street. ***H*: THERE ARE A LOT OF PEOPLE WALKING ALONG.** *Label*: ENTAILMENT
GENERATED	***H'*: THERE ARE A LOT GIRLS WALKING ALONG. *Label*:** NEUTRAL
TRUE INPUT 4	*P*: A soldier is looking out of a window. ***H*: THE PRISONER'S CELL IS WINDOWLESS.** *Label*: CONTRADICTION
GENERATED	***H'*: THE PRISONER'S HOME IS WINDOWLESS. *Label*:** CONTRADICTION
TRUE INPUT 5	*P*: Four people sitting on a low cement ledge. ***H*: THERE ARE FOUR PEOPLE.** *Label*: ENTAILMENT
GENERATED	***H'*: THERE ARE SEVERAL PEOPLE. *Label*:** NEUTRAL
TRUE INPUT 6	*P*: Three youngsters shovel a huge pile of snow. ***H*: CHILDREN WORKING TO CLEAR SNOW.** *Label*: ENTAILMENT
GENERATED	***H'*: KIDS WORKING TO CLEAR SNOW. *Label*:** NEUTRAL
TRUE INPUT 7	*P*: Boys at an amphitheater. ***H*: BOYS AT A SHOW.** *Label*: ENTAILMENT
GENERATED	***H'*: BOYS IN A SHOW. *Label*:** NEUTRAL
TRUE INPUT 8	*P*: Male child holding a yellow balloon. ***H*: BOY HOLDING BIG BALLOON.** *Label*: NEUTRAL
GENERATED	***H'*: BOY HOLDING LARGE BALLOON.** ***Label*:** NEUTRAL
TRUE INPUT 9	*P*: Women in their swimsuits sunbathe on the sand. ***H*: WOMEN UNDER THE SUN ON THE SAND.** *Label*: ENTAILMENT
GENERATED	***H'*: FAMILY UNDER THE SUN ON THE SAND.** ***Label*:** NEUTRAL

H. Experimental Settings

Table 10: Model Configurations + SNLI Classifier + Hyper-parameters.

	Name	Configuration
Recognition Networks	f_η	Input Dim: 50, Hidden Layers: [60, 70], Output Dim: Num weights & biases in θ_m
Model Instances	Particles θ_m	Input Dim: 28×28 (MNIST), 64×64 (CelebA), 32×32 (SVHN), 300 (SNLI) Hidden Layers: [40, 40] Output Dim (latent code): 100
Feature Extractor	Input Dim: $28 \times 28 \times 1$ (MNIST), $64 \times 64 \times 3$ (CelebA), $32 \times 32 \times 3$ (SVHN), 10×100 (SNLI) Hidden Layers: [40, 40] Output Dim: 28×28 (MNIST), 64×64 (CelebA), 32×32 (SVHN), 100 (SNLI)	
Decoder	Transpose CNN	For CelebA & SVHN: [filters: 64, stride: 2, kernel: 5]$\times$ 3 For SNLI: [filters: 64, stride: 1, kernel: 5]$\times$ 3
	Language Model	Vocabulary Size: 11,000 words Max Sentence Length: 10 words
SNLI classifier	Input Dim: 200, Hidden Layers: [100, 100, 100], Output Dim: 3	
Learning Rates	$\alpha = 10^{-2}$ and $\beta = 10^{-3}$	
More settings	Batch size: 64, Inner-updates: 3, Training epochs: 1000, $M = 5$	

I. Related Work

Manifold Learning. VAEs are generally used to learn manifolds [35, 11, 14] by maximizing the ELBO of the data log-likelihood [1, 7]. Optimizing the ELBO entails reparameterizing the encoder to a Gaussian distribution [20]. This reparameterization is, however, restrictive [17] as it may lead to learning poorly the manifold of the data [37]. To alleviate this issue, we use SVGD. While our approach and [28] may look similar, we use Bayesian inference which is more principled than dropout [15].

Adversarial Examples. Studies in adversarial deep learning [3, 4, 22, 12] can be categorized into two groups. The first group [4, 6, 27] proposes to generate adversarial examples directly in the input space of the source inputs by distorting, occluding or changing illumination in images to cause changes in classification. The second group [33, 38], where our work belongs, uses generative models to search for adversarial examples in the dense and continuous representations of the data rather than in its input space.

Adversarial Images [33] propose to construct unrestricted adversarial examples by training a conditional GAN that constrains the search region for a latent code z' in the neighborhood of a target z. [38] use also a GAN to map input images to a latent space where they conduct their search for adversarial examples. To our knowledge, these studies are the closest to ours. Unlike in [33, 38], however, our adversarial perturbations are learned. Moreover, we do not restrict the search for adversarial examples to uniformly-bounded regions. In contrast also to [33] and [38], where the search for adversarial examples is iterative and decoupled from the training of the GANs, our method is end-to-end. Lastly, by capturing the uncertainty induced by embedding the data, we generate more realistic adversarial examples.

Adversarial Text: Previous studies on adversarial text generation [36, 16, 2, 24] perform word erasures and replacements directly in the input space using domain-specific rules or heuristics, or they require manual curation. Similar to us, [38] propose to search for textual adversarial examples in the latent representation of the data. However, in addition to the differences aforementioned for images, the search for adversarial examples is handled in our case by an efficient gradient-based optimization method.