Resisting Adversarial Attacks Using Gaussian Mixture Variational Autoencoders

Partha Ghosh,[*1] **Arpan Losalka,**[*2] **Michael J Black**[1]
[1]Max Planck Institute of Intelligent Systems, [2]IBM Research AI
partha.ghosh@tuebingen.mpg.de, arlosalk@in.ibm.com, black@tuebingen.mpg.de

Abstract

Susceptibility of deep neural networks to adversarial attacks poses a major theoretical and practical challenge. All efforts to harden classifiers against such attacks have seen limited success till now. Two distinct categories of samples against which deep neural networks are vulnerable, "adversarial samples" and "fooling samples", have been tackled separately so far due to the difficulty posed when considered together. In this work, we show how one can defend against them both under a unified framework. Our model has the form of a variational autoencoder with a Gaussian mixture prior on the latent variable, such that each mixture component corresponds to a single class. We show how selective classification can be performed using this model, thereby causing the adversarial objective to entail a conflict. The proposed method leads to the rejection of adversarial samples instead of misclassification, while maintaining high precision and recall on test data. It also inherently provides a way of learning a selective classifier in a semi-supervised scenario, which can similarly resist adversarial attacks. We further show how one can reclassify the detected adversarial samples by iterative optimization.[1]

1 Introduction

The vulnerability of deep neural networks to adversarial attacks has generated a lot of interest and concern in the past few years. The fact that these networks can be easily fooled by adding specially crafted noise to the input, such that the original and modified inputs are indistinguishable to humans (2014), clearly suggests that they fail to mimic the human learning process. Even though these networks achieve state-of-the-art performance, often surpassing human level performance (2015; 2017) on the test data used for different tasks, their vulnerability is a cause of concern when deploying them in real life applications, especially in domains such as health care (2018), autonomous vehicles (2018) and defense, etc.

Adversarial Attacks and Defenses

Adversarially crafted samples can be classified into two broad categories, namely (i) adversarial samples (2014) and (ii) fooling samples as defined by (2015). Existence of adversarial samples was first shown by Szegedy et al. (2014), while fooling samples (2015), which are closely related to the idea of "rubbish class" images (1998) were introduced by Nguyen et al. (2015). Evolutionary algorithms were applied to inputs drawn from a uniform distribution, using the predicted probability corresponding to the targeted class as the fitness function (2015) to craft such fooling samples. It has also been shown that Gaussian noise can be directly used to trick classifiers into predicting one of the output classes with very high probability (2014).

Adversarial attack methods can be classified into (i) white box attacks (2014; 2014; 2017b; 2016a; 2016; 2018), which use knowledge of the machine learning model (such as model architecture, loss function used during training, etc.) for crafting adversarial samples, and (ii) black box attacks (2017; 2016; 2017), which only require the model for obtaining labels corresponding to input samples. Both these kinds of attacks can be further split into two sub categories, (i) targeted attacks, which trick the model into producing a chosen output, and (ii) non-targeted attacks, which cause the model to produce any undesired output (2014). The majority of attacks and defenses have dealt with adversarial samples so far (2014; 2014; 2016b), while a relatively smaller literature deals with fooling samples (2015). However, to the best of our knowledge, no prior method tries to defend against both kinds of samples simultaneously under a unified framework. State-of-the-art defense mechanisms have tried to harden a classifier by one or more of the following techniques: adversarial retraining (2014), preprocessing inputs (2014), deploying auxiliary detection networks (2017a) or obfuscating gradients (2018). One common drawback of these defense mechanisms is that they do not eliminate the vulnerability of deep networks altogether, but only try to defend against previously proposed attack methods. Hence, they have been easily broken by stronger attacks, which are specifically designed to overcome their defense strategies (2016; 2018).

Szegedy et al. (2014) argue that the primary reason for the existence of adversarial samples is the presence of small "pockets" in the data manifold, which are rarely sampled in the training or test set. On the other hand, Goodfellow et al. (2014) have proposed the "linearity hypothesis" to explain the presence of adversarial samples. Under our approach as

[*]Equal contribution

[1]Accepted for publication in the Proceedings of the Thirty-Third AAAI Conference on Aritificial Intelligence (AAAI 2019)

detailed in Sec. 3, the adversarial objective poses a fundamental conflict of interest, and inherently addresses both these possible explanations.

Approach

We design a generative model that finds a latent random variable $\mathbf{z}$ such that data label $\mathbf{y}$ and the data $\mathbf{x}$ become conditionally independent given $\mathbf{z}$, i.e., $P(\mathbf{x}, \mathbf{y}|\mathbf{z}) = P(\mathbf{x}|\mathbf{z})P(\mathbf{y}|\mathbf{z})$. We base our generative model on VAEs (2014), and obtain an inference model that represents $P(\mathbf{z}|\mathbf{x})$ and a generative model that represents $P(\mathbf{x}|\mathbf{z})$. We perform label inference $P(\mathbf{y}|\mathbf{x})$ by computing $\arg\max_{\mathbf{y}} P(\mathbf{y}|\mathbf{z} = \mathbb{E}_{P(\mathbf{z}|\mathbf{x})}[\mathbf{z}])$. We choose the latent space distribution $P(\mathbf{z})$ to be a mixture of Gaussians, such that each mixture component represents one of the classes in the data. Under this construct, inferring the label given latent encoding, i.e., $P(\mathbf{y}|\mathbf{z})$ becomes trivial by computing the contribution of the mixture components. Adversarial samples are dealt with by thresholding in the latent and output spaces of the generative model and rejecting the inputs for which $P(\mathbf{x}) \approx 0$. In Figure 1, we describe our network at test and train time.

Our contributions can be summarized as follows.

- We show how VAE's can be trained with labeled data, using a Gaussian mixture prior on the latent variable in order to perform classification.
- We perform selective classification using this framework, thereby rejecting adversarial and fooling samples.
- We propose a method to learn a classifier in a semi-supervised scenario using the same framework, and show that this classifier is also resistant against adversarial attacks.
- We also show how the detected adversarial samples can be reclassified into the correct class by iterative optimization.
- We verify our claims through experimentation on 3 publicly available datasets: MNIST (1998), SVHN (2011) and COIL-100 (1996).

2 Related Work

A few pieces of work in the existing literature on defense against adversarial attacks have attempted to use generative models in different ways.

Samangouei et al. (2018) propose training a Generative Adversarial Network (GAN) on the training data of a classifier, and use this network to project every test sample on to the data manifold by iterative optimization. This method does not try to detect adversarial samples, and does not tackle "fooling images". Further, this defense technique has been recently shown to be ineffective (2018). Other pieces of work have also shown that adversarial samples can lie on the output manifold of generative models trained on the training data for a classifier (2018).

PixelDefend, proposed by Song et al. (2018a) also uses a generative model to detect adversarial samples, and then rectifies the classifier output by projecting the adversarial input back to the data manifold. However, Athalye et al. have shown that this method can also be broken by bypassing the exploding/vanishing gradient problem introduced by the defense mechanism.

MagNet (2017b) uses autoencoders to detect adversarial inputs, and is similar to our detection mechanism in the way reconstruction threshold is used for detecting adversarial inputs. This defense method does not claim security in the white box setting. Further, the technique has also been broken in the grey box setting by recently proposed attack methods (2017a).

Traditional autoencoders do not constrain the latent representation to have a specific distribution like variational autoencoders. Our use of variational autoencoders allows us to defend against adversarial and fooling inputs simultaneously, by using thresholds in the latent and output spaces of the model in conjunction. This makes the method secure to white box attacks as well, which is not the case with MagNet.

Further, even state of the art defense mechanisms (2018) and certified defenses have been shown to be ineffective for simple datasets such as MNIST (2018b). We show via extensive experimentation on different datasets how our method is able to defend against strong adversarial attacks, as well as end to end white box attacks.

3 Method

Variational Autoencoders

We consider the dataset $\mathbf{X} = \{\mathbf{x}^{(i)}\}_{i=1}^{N}$ consisting of N i.i.d. samples of a random variable $\mathbf{x}$ in the space $\mathcal{X}$. Let $\mathbf{z}$ be the latent representation from which the data is assumed to have been generated. Similar to Kingma et al. (2014), we assume that the data generation process consists of two steps: (i) a value $\mathbf{z}^{(i)}$ is sampled from a prior distribution $P_{\theta^*}(\mathbf{z})$; (ii) a value $\mathbf{x}^{(i)}$ is generated from a conditional distribution $P_{\theta^*}(\mathbf{x}|\mathbf{z})$. We also assume that the prior $P_{\theta^*}(\mathbf{z})$ and likelihood $P_{\theta^*}(\mathbf{x}|\mathbf{z})$ come from parametric families of distributions $P_\theta(\mathbf{z})$ and $P_\theta(\mathbf{x}|\mathbf{z})$ respectively. In order to maximize the data likelihood $P_\theta(\mathbf{x}) = \int P_\theta(\mathbf{z})P_\theta(\mathbf{x}|\mathbf{z})d\mathbf{z}$, VAEs (2014) use an encoder network $Q_\phi(\mathbf{z}|\mathbf{x})$, that approximates $P_\theta(\mathbf{z}|\mathbf{x})$. The evidence lower bound (ELBO) for VAE is given by

$$\begin{aligned} ELBO(\mathbf{x}, \theta, \phi) = \mathbb{E}_{\mathbf{z}\sim Q_\phi(\mathbf{z}|\mathbf{x})}[\log P_\theta(\mathbf{x}|\mathbf{z})] \\ -D_{KL}[Q_\phi(\mathbf{z}|\mathbf{x})||P(\mathbf{z})] \end{aligned} \tag{1}$$

where D_{KL} represents the KL divergence measure. Using a Gaussian prior $P_\theta(\mathbf{z})$ and a Gaussian posterior $Q_\phi(\mathbf{z}|\mathbf{x})$, variational autoencoders maximize this lower bound deriving a closed form expression for the KL divergence term.

Modifying the Evidence Lower Bound

VAEs do not enforce any lower or upper bound on encoder entropy $H(Q_\phi(\mathbf{z}|\mathbf{x}))$. This can result in blurry reconstruction due to sample averaging in case of overlap in the latent space. On the other hand, unbounded decrease in $H(Q_\phi(\mathbf{z}|\mathbf{x}))$ is not desirable either, as in that case the VAE can degenerate to a deterministic autoencoder leading to holes in the latent space. Hence, we seek an alternative design in which we fix this quantity to a constant value. In order to do so, we express

the KL divergence in terms of entropy.

$$\begin{aligned} & D_{KL}[Q_\phi(\mathbf{z}|\mathbf{x}) \,\|\, P_\theta(\mathbf{z})] \\ & = -\mathbb{E}_{\mathbf{z}\sim Q_\phi(\mathbf{z}|\mathbf{x})}\left[\log\, P_\theta(\mathbf{z}) - \log\, Q_\phi(\mathbf{z}|\mathbf{x})\right] \\ & = -\mathbb{E}_{\mathbf{z}\sim Q_\phi(\mathbf{z}|\mathbf{x})}\left[\log\, P_\theta(\mathbf{z})\right] + \mathbb{E}_{\mathbf{z}\sim Q_\phi(\mathbf{z}|\mathbf{x})}\left[\log\, Q_\phi(\mathbf{z}|\mathbf{x})\right] \\ & = H(Q_\phi(\mathbf{z}|\mathbf{x}), P_\theta(\mathbf{z})) - H(Q_\phi(\mathbf{z}|\mathbf{x})) \end{aligned} \tag{2}$$

where $H(Q_\phi(\mathbf{z}|\mathbf{X}), P_\theta(\mathbf{z}))$ represents the cross entropy between $Q_\phi(\mathbf{z}|\mathbf{X})$ and $P_\theta(\mathbf{z})$. It can be noted that we need to minimize the KL divergence term. Hence, if we assume that $H(Q_\phi(\mathbf{z}|\mathbf{x}))$ is constant, then we can drop this term during optimization (please refer to the next section for details of how $H(Q_\phi(\mathbf{z}|\mathbf{x}))$ is enforced to be constant). This lets us replace the KL divergence $D_{KL}[Q_\phi(\mathbf{z}|\mathbf{X}) \,\|\, P_\theta(\mathbf{z})]$ in the loss function with $H(Q_\phi(\mathbf{z}|\mathbf{X}), P_\theta(\mathbf{z}))$.

$$\begin{aligned} & ELBO(\mathbf{x}, \theta, \phi) \\ & = \mathbb{E}_{\mathbf{z}\sim Q_\phi(\mathbf{z}|\mathbf{x})}[\log P_\theta(\mathbf{x}|\mathbf{z})] - H(Q_\phi(\mathbf{z}|\mathbf{x}), P_\theta(\mathbf{z})) \\ & = \mathbb{E}_{\mathbf{z}\sim Q_\phi(\mathbf{z}|\mathbf{x})}[\log P_\theta(\mathbf{x}|\mathbf{z})] + \mathbb{E}_{\mathbf{z}\sim Q_\phi(\mathbf{z}|\mathbf{x})}\left[\log\, P_\theta(\mathbf{z})\right] \end{aligned} \tag{3}$$

The choice of fixing the entropy of $Q_\phi(\mathbf{z}|\mathbf{x})$ is further justified via experiments in section 4.

Supervision using a Gaussian Mixture Prior

In this section, we modify the above ELBO term for supervised learning by including the random variable $\mathbf{y}$ denoting labels. The following expression can be derived for the log-likelihood of the data.

$$\begin{aligned} & \log(P_\theta(\mathbf{x}, \mathbf{y})) = \mathbb{E}_{\mathbf{z}\sim Q_\phi(\mathbf{z}|\mathbf{x})}[\log(P_\theta(\mathbf{x}, \mathbf{y}|\mathbf{z}))] \\ & - D_{KL}[Q_\phi(\mathbf{z}|\mathbf{x})||P_\theta(\mathbf{z})] + D_{KL}[Q_\phi(\mathbf{z}|\mathbf{x})||P_\theta(\mathbf{z}|\mathbf{x}, \mathbf{y})] \end{aligned} \tag{4}$$

Noting that $D_{KL}[Q_\phi(\mathbf{z}|\mathbf{x})||P_\theta(\mathbf{z}|\mathbf{x}, \mathbf{y})] \geq 0$, and replacing $D_{KL}[Q_\phi(\mathbf{z}|\mathbf{x})||P_\theta(\mathbf{z})]$ with $\mathbb{E}_{\mathbf{z}\sim Q_\phi(\mathbf{z}|\mathbf{x})}\left[\log\, P_\theta(\mathbf{z})\right]$ by assuming $H(Q_\phi(\mathbf{z}|\mathbf{x}))$ to be constant (as shown in Eqn. 3), we get the following lower bound on the data likelihood.

$$\begin{aligned} ELBO(\mathbf{x}, \mathbf{y}, \theta, \phi) = \mathbb{E}_{\mathbf{z}\sim Q_\phi(\mathbf{z}|\mathbf{x})}[\log(P_\theta(\mathbf{x}, \mathbf{y}|\mathbf{z}))] \\ + \mathbb{E}_{\mathbf{z}\sim Q_\phi(\mathbf{z}|\mathbf{x})}\left[\log\, P_\theta(\mathbf{z})\right] \end{aligned} \tag{5}$$

We choose our VAE to use a Gaussian mixture prior for the latent variable $\mathbf{z}$. We further choose the number of mixture components to be equal to the number of classes k in the training data. The means of each of these components, $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, ..., \boldsymbol{\mu}_k$ are assumed to be the one-hot encodings of the class labels in the latent space. It can be noted here that although this choice enforces the latent dimensionality to be k, it can be easily altered by choosing the means in a different manner. For example, means of all the mixture components can lie on a single axis in the latent space. Unlike usual VAEs, our encoder network outputs only the mean of $Q_\phi(\mathbf{z}|\mathbf{x})$. We use the reparameterization trick introduced by Kingma et al. (2014), but sample the input $\boldsymbol{\epsilon}$ from $N(0, \Sigma_{constant})$ in order to enforce the entropy of $Q_\phi(\mathbf{z}|\mathbf{x})$ to be constant. Here, each mixture component corresponds to one class and $\mathbf{x}$ is assumed to be generated from the latent space according to $P_\theta(\mathbf{x}|\mathbf{z})$ irrespective of $\mathbf{y}$. Therefore, $\mathbf{x}$ and $\mathbf{y}$ become conditionally independent given $\mathbf{z}$, i.e. $\log(P_\theta(\mathbf{x}, \mathbf{y}|\mathbf{z})) = \log(P_\theta(\mathbf{x}|\mathbf{z})) + \log(P_\theta(\mathbf{y}|\mathbf{z}))$.

$$\begin{aligned} & ELBO(\mathbf{x}, \mathbf{y}, \theta, \phi) \\ & = \mathbb{E}_{\mathbf{z}\sim Q_\phi(\mathbf{z}|\mathbf{x})}\left[\log(P_\theta(\mathbf{x}|\mathbf{z})) + \log(P_\theta(\mathbf{y}|\mathbf{z})) + \log\, P_\theta(\mathbf{z})\right] \\ & = \mathbb{E}_{\mathbf{z}\sim Q_\phi(\mathbf{z}|\mathbf{x})}\left[\log(P_\theta(\mathbf{x}|\mathbf{z})) + \log(P_\theta(\mathbf{z}|\mathbf{y})) + \log\, P_\theta(\mathbf{y})\right] \end{aligned} \tag{6}$$

Assuming the the classes to be equally likely, the final loss function for an input $\mathbf{x}^{(i)}$ with label $\mathbf{y}^{(i)}$ becomes the following.

$$\begin{aligned} \mathcal{L}(\mathbf{x^{(i)}}, \mathbf{y}^{(i)}, \boldsymbol{\epsilon}) = ||\mathbf{x}^{(i)} - g(f(\mathbf{x}^{(i)}) + \epsilon)||^2 \\ + \alpha||f(\mathbf{x}^{(i)}) - \boldsymbol{\mu}_{\mathbf{y}^{(i)}}||^2 \end{aligned} \tag{7}$$

where the encoder is represented by f, the decoder is represented by g and $\boldsymbol{\mu}_{\mathbf{y}^{(i)}}$ represents the mean of the mixture component corresponding to $\mathbf{y}^{(i)}$. α is a hyper-parameter that trades off between reconstruction fidelity, latent space prior and classification accuracy.

The label $\mathbf{y}$ for an input sample $\mathbf{x}$ can be obtained following the Bayes Decision rule.

$$\begin{aligned} \arg\max_{\mathbf{y}}\, P_\theta(\mathbf{y}|\mathbf{x}) &= \arg\max_{\mathbf{y}} P_\theta(\mathbf{x}, \mathbf{y}) \\ &= \arg\max_{\mathbf{y}} \int_{\mathbf{z}} P_\theta(\mathbf{x}, \mathbf{y}|\mathbf{z}) P_\theta(\mathbf{z}) d\mathbf{z} \\ &= \arg\max_{\mathbf{y}} \int_{\mathbf{z}} P_\theta(\mathbf{x}|\mathbf{z}) P_\theta(\mathbf{y}|\mathbf{z}) P_\theta(\mathbf{z}) d\mathbf{z} \\ &= \arg\max_{\mathbf{y}} \int_{\mathbf{z}} P_\theta(\mathbf{z}|\mathbf{x}) P_\theta(\mathbf{y}|\mathbf{z}) P_\theta(\mathbf{x}) d\mathbf{z} \\ &= \arg\max_{\mathbf{y}} \int_{\mathbf{z}} P_\theta(\mathbf{z}|\mathbf{x}) P_\theta(\mathbf{y}|\mathbf{z}) d\mathbf{z} \end{aligned} \tag{8}$$

$P_\theta(\mathbf{z}|\mathbf{x})$ can be approximated by $Q_\phi(\mathbf{z}|\mathbf{x})$, i.e., the encoder distribution. This corresponds to the Bayes decision rule, in the scenario where there is no overlap among the classes in the input space, ϕ has enough variability and $Q_{\phi^*}(\mathbf{z}|\mathbf{x})$ is able to match $P_{\theta^*}(\mathbf{z}|\mathbf{x})$ exactly.

Semi-supervised learning follows automatically, by using the loss function in Eqn. 7 for labeled samples, and the loss corresponding to Eqn. 3 for unlabeled samples.

In order to compute the class label as defined in equation 8, we use a single sample estimate of the integration by simply using the mean of $Q_\phi(\mathbf{z}|\mathbf{x})$ as the $\mathbf{z}$ value in our experiments. This choice does not affect the accuracy as long as the mixture components representing the classes are well separated in the latent space.

Resisting adversarial attacks

In order to successfully reject adversarial samples irrespective of the method of its generation, we use thresholding at the encoder and decoder outputs. This allows us to reject any sample $\mathbf{x}$ whose encoding $\mathbf{z}$ has low probability under $P_\theta(\mathbf{z})$, i.e., if the distance between its encoding and the encoding of the predicted class label in the latent space exceeds a threshold value, τ_{enc} (since $P_\theta(\mathbf{z})$ is a mixture of Gaussians). We further reject those input samples which have low probability under $P_\theta(\mathbf{x}|\mathbf{z})$, i.e., if the reconstruction error

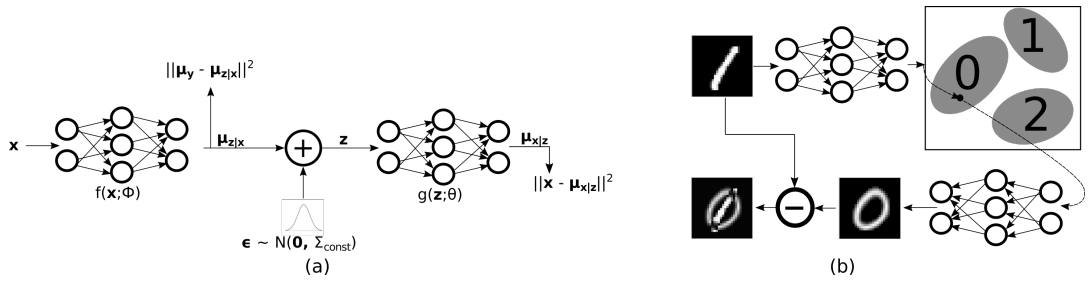

Figure 1: (a) The model at training time. All the inputs are in green, while all the losses are in brown. (b) Model pipeline at inference time. The red dot shows that the attacker is successful in fooling the encoder by placing its output in the wrong class. However, it results in a high reconstruction error, since the decoder generates an image of the target class.

exceeds a certain threshold, τ_{dec} (since $P_\theta(\mathbf{x}|\mathbf{z})$ is Gaussian). Essentially, a combination of these two thresholds ensures that $P_\theta(\mathbf{x}) = \int_{\mathbf{z}} P_\theta(\mathbf{x}|\mathbf{z})P_\theta(\mathbf{z})d\mathbf{z}$ is not low.

Both τ_{enc} and τ_{dec} can be determined based on statistics obtained while training the model. In our experiments, we implement thresholding in the latent space as follows: we calculate the Mahalanobis distance between the encoding of the input and the encoding of the corresponding mixture component mean, and reject the sample if it exceeds the critical chi-square value (3σ rule in the univariate case). Similarly, for τ_{dec}, we use the corresponding value for the reconstructions errors. However, in general, any value can be assigned to these two thresholds, and they determine the risk to coverage trade-off for this selective classifier.

If the maximum allowed L_p norm of the perturbation $\boldsymbol{\eta}$ is γ, then the adversary, trying to modify an input $\mathbf{x}$ from class c_1, must satisfy the following criteria.

1. $\arg\min_{c_i} ||f(\mathbf{x}+\boldsymbol{\eta}) - \boldsymbol{\mu}_{c_i}||_2 = c_2$ where $c_2 \neq c_1$
2. $||f(\mathbf{x}+\boldsymbol{\eta}) - \boldsymbol{\mu}_{c_2}||_2 \leq \tau_{enc}$
3. $||\boldsymbol{\eta}||_p \leq \gamma$
4. $||(\mathbf{x}+\boldsymbol{\eta}) - g(f(\mathbf{x}+\boldsymbol{\eta}) + \boldsymbol{\epsilon})||_2 \leq \tau_{dec}$ where $\boldsymbol{\epsilon} \sim N(0, \Sigma_{constant})$

By the first three constraints, the encoding of $\mathbf{x}$ and $\mathbf{x}+\boldsymbol{\eta}$ must belong to different Gaussian mixture components in the latent space. However, constraint 4 requires the distance between the reconstruction obtained from the encoding of $\mathbf{x}+\boldsymbol{\eta}$ to be close to $\mathbf{x}+\boldsymbol{\eta}$, i.e., close to $\mathbf{x}$ in the pixel space. This is extremely hard to satisfy because of the low probability of occurrence of holes in the latent space within τ_{enc} distance from the means.

Similarly, for the case of fooling samples, it can be argued that even if an attacker manages to generate a fooling sample which tricks the encoder, it will be very hard to simultaneously trick the decoder to reconstruct a similar image belonging to the rubbish class.

Reclassification

Once a sample is detected as adversarial by either or both the thresholds discussed above, we attempt to find its true label using the decoder only. By definition of adversarial images, $\mathbf{x}_{adv} = \mathbf{x}_{org} + \boldsymbol{\eta}$, where $\mathbf{x}_{adv}$ is the adversarial image corresponding to the original image $\mathbf{x}_{org}$, and $||\boldsymbol{\eta}||_p$ is small. Hence, we can conclude that for any given image $\mathbf{x}$, $||\mathbf{x} - \mathbf{x}_{adv}||_p \approx ||\mathbf{x} - \mathbf{x}_{org}||_p$. Suppose $\mathbf{z}^*$ is given by Eqn. 9.

$$\mathbf{z}^* = \arg\min_{\mathbf{z}} ||g(\mathbf{z}) - \mathbf{x}_{org}||_p \tag{9}$$

Following the argument stated above, we can approximate $\mathbf{z}^* \approx \mathbf{z}^*_{adv} = \arg\min_{\mathbf{z}} ||g(\mathbf{z}) - \mathbf{x}_{adv}||_p$. We can now find the label of the adversarial sample as $\arg\min_{c_i} ||\boldsymbol{\mu}_{c_i} - \mathbf{z}^*_{adv}||_2$. Essentially, for reclassification, we try to find the $\mathbf{z}$ in the latent space, which, when decoded, gives the minimum reconstruction error from the adversarial input. However, if Eqn. 9 returns a $\mathbf{z}$ that lies beyond τ_{enc} from the corresponding mean, or if the reconstruction error exceeds τ_{dec}, we conclude that the sample is a fooling sample and reject the sample. It can be noted here that if this network is deployed in a scenario where fooling samples are not expected to be encountered, one can choose not to reject samples during reclassification, thereby increasing coverage. Also, starting from a single value of $\mathbf{z}$ can cause the optimization process to get stuck at a local minimum. A better alternative is to run k different optimization processes with $\mathbf{z} = \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_k$ as the initial values, and choose the $\mathbf{z}$ which gives minimum reconstruction error as $\mathbf{z}^*_{adv}$. Given enough compute power is available, these k processes can be run in parallel. In our experiments, we follow these two strategies while reclassifying adversarial samples.

4 Experiments

We verify the effectiveness of our network through numerical results and visual analysis on three different datasets - MNIST, SVHN and COIL-100. For different datasets, we make minimal changes to the hyper-parameters of our network, partly due to the difference in the image size and image type (grayscale/colored) in each dataset.

Implementation details. We use an encoder network with convolution, max-pooling and dense layers to parameterize $Q_\phi(\mathbf{z}|\mathbf{x})$, and a decoder network with convolution, upsampling and dense layers to parameterize $P_\theta(\mathbf{x}|\mathbf{z})$. We choose the dimensionality of the latent space to be the same as the number of classes for MNIST and COIL-100. However, noting that the size of images is larger for SVHN compared to MNIST, and also, because the dataset contains colored images, we choose the dimensionality of the latent space

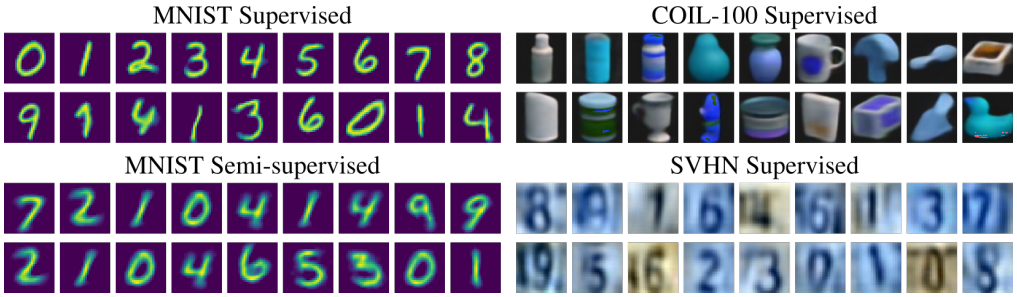

Figure 2: Generated images from different classes of MNIST, COIL-100, SVHN.

for SVHN as 32 instead of 10. The choice of means also varies slightly for this dataset, as we pad zeros to the one-hot encodings of the class labels to allow for the extra latent dimensions. The standard deviation of the encoder distribution is chosen such that the chance of overlap of the mixture components in the latent space is negligible and the classes are well separated. We use $1/3000$ as the variance for the MNIST dataset, and reduce this value as the latent dimensionality increases for the other datasets. We use the ReLU nonlinearity in our network, and sigmoid activation in the final layer so that the output lies in the allowed range $[0, 1]$. We use the Adam(2014) optimizer for training.

Qualitative evaluation. Since our algorithm relies upon the reconstruction error between the generated and the original samples, we first show a few randomly chosen images generated by the network (for both supervised ad semi-supervised scenarios) corresponding to test samples of different classes from the three datasets in Figure 2.

Numerical results. In Table 1, we present the accuracy, error and rejection percentages obtained by our method with and without thresholding. For semi-supervised learning, we have taken 100 randomly chosen labeled samples from each class for both MNIST and SVHN during training. It is important to note here that the SOTA for COIL-100 was obtained on a random train-test split of the dataset, and hence, the accuracy values are not directly comparable.

Adversarial attacks on encoder. We use the encoder part of the network trained on the MNIST dataset to generate adversarial samples using the *Fast Gradient Sign Method (FGSM)* with varying ϵ values (2014). The corresponding results are shown in Figure 3. The behavior is as desired, i.e., with increasing ϵ, percentage of misclassified samples rises to a maximum value of only 3.89% and then decreases, while the accuracy decreases monotonically and the rejection percentage increases monotonically. Similar results are obtained for the semi-supervised model, as shown in Figure 3, although the maximum error rate is higher in this case. We further tried the FGSM attack from the Cleverhans library (2017) with the default parameters on the SVHN and COIL-100 datasets, and all the generated samples were rejected by the models after thresholding. Similarly, we generated adversarial samples for all three datasets using stronger attacks from Cleverhans with default parameter settings, including the Momentum Iterative Method (2018) and Projected Gradient Descent (2018). In these cases as well, all generated adversarial samples were successfully rejected by thresholding.

This indicates that since all these attacks lack knowledge of the decoder network, they only manage to produce samples which fool the encoder network, but are easily detected at the decoder output. From this set of experiments, we conclude that the only effective method of attacking our model would be to design a complete white-box attack that has knowledge of the decoder loss as well, as well as the two thresholds. Further, since we do not use any form of gradient obfuscation in our defense mechanism, a complete white-box attacker would represent a strong adversary.

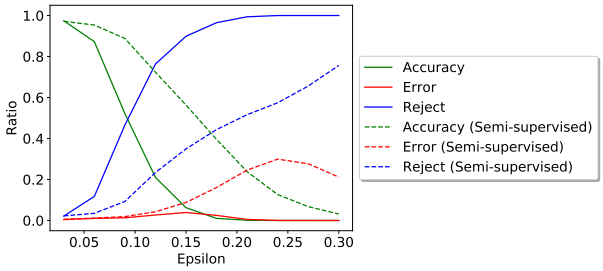

Figure 3: We run FGSM with varying ϵ on the models trained on MNIST data in both supervised and semi-supervised scenarios. Although the error rate is higher for the semi-supervised network, the rejection ratio rises monotonically for both networks with increasing ϵ, and the error rate for the supervised model stays below 5%.

White-box adversarial attack. We present the results for completely white-box targeted attack on our model for the COIL-100 and MNIST datasets in figures 4a and 4b. Here, the adversary has complete knowledge of the encoder, the decoder, as well as the rejection thresholds. The results shown correspond to random samples from the first two classes of objects for the COIL-100 dataset, and the classes 2 and 5 for MNIST dataset. We perform gradient descent on the adversarial objective as given in Eqn. 10. The target class is set to 6 for MNIST images from class 2, 9 for MNIST images from class 5, and the class other than that of the source image

	Supervised					Semi-supervised			
	Without thresholding		With thresholding			Without thresholding	With thresholding		
Dataset	SOTA	Accuracy	Accuracy	Error	Rejection	Accuracy	Accuracy	Error	Rejection
MNIST	99.79%	99.67%	97.97%	0.22%	1.81%	99.1%	98.17%	0.52%	1.31%
SVHN	98.31%	95.06%	92.80%	4.58%	2.62%	86.42%	83.54%	13.64%	2.82%
COIL-100	99.11%	99.89%	98.40%	0%	1.60%	-	-	-	-

Table 1: Comparison between the performance of the state-of-the-art (SOTA) models and our model. We show that our method, even without much fine tuning focused on achieving classification accuracy, is competitive with the SOTA. MNIST SOTA is as reported by (2013), SVHN SOTA is as given by (2016) and the SOTA for COIL-100 is given by (2015).

for the COIL-100 images.

$$\arg\min_{\boldsymbol{\eta}} \mathcal{L}_{adv} = \arg\min_{\eta}[((||\mathbf{x_o} + \boldsymbol{\eta} - g(f(\mathbf{x_o} + \boldsymbol{\eta}))||^2)/\tau_{dec})^a + ((\mu_{\mathbf{t}} - f(\mathbf{x_o} + \eta))\Sigma_t(\mu_{\mathbf{t}} - f(\mathbf{x_o} + \boldsymbol{\eta}))/\tau_{enc})^b + ||\boldsymbol{\eta}||^2] \tag{10}$$

where $\mathbf{x_o}$ is the original image we wish to corrupt, $\boldsymbol{\mu_t}$ is the mean of target class, $\boldsymbol{\eta}$ is the noise added, f, g are the encoder and decoder respectively, and Σ_t denotes target class covariance in latent space. $a > 1$ and $b > 1$ represent constant exponents which ensure that the adversarial loss grows steeply when the two threshold values are exceeded. Essentially, we aim for low reconstruction error and small change in the adversarial image while moving its embedding close to the target class mean. $\boldsymbol{\eta}$ is initialized with zeros.

We also ran the white box attack on 100 randomly sampled images from each of the 10 classes for MNIST and SVHN, by setting each of the 9 other classes as the target class. The samples generated by optimizing the adversarial objective in each of these cases were either correctly classified or rejected.

Fooling images. We take 100 images sampled from the uniform distribution as inputs and optimize the white-box fooling attack objective given by Eqn. 11, with each of the classes from the MNIST and SVHN datasets as the target classes. In Figure 4c, we visualize some of the images to which the attack converged and their reconstructions for the MNIST dataset, with the target classes $1, 2, \ldots, 6$.

$$\arg\min_{\boldsymbol{\eta}} \mathcal{L}_{fool} = \arg\min_{\boldsymbol{\eta}}[(||\boldsymbol{\eta} - g(f(\boldsymbol{\eta}))||^2/\tau_{dec})^a + ((\boldsymbol{\mu_t} - f(\boldsymbol{\eta}))\Sigma_t(\boldsymbol{\mu_t} - f(\boldsymbol{\eta}))/\tau_{enc})^b] \tag{11}$$

Here, $\boldsymbol{\eta}, a, b, f, g, \Sigma_t$ and $\boldsymbol{\mu_t}$ are as described in sec. 4.

It has been shown that fooling samples are extremely easy to generate for state-of-the-art classifier networks (2014; 2015). Our technique, by design, gains resilience against such attacks as well. Since by definition, a fooling sample cannot look like a legitimate sample, it can not have small pixel space distance with any real image. This is exactly what can be noticed in the results in Figure 4c, where reconstruction errors are very high. Hence, most of the images to which this attack converges are rejected at the decoder, although they had managed to fool the encoder when considered in isolation. For the few cases where the images are not rejected, we observe that the attack method actually converged to a legitimate image of the target class.

Reclassifying Adversarial samples. In this section we present the performance of our reclassification technique. Although one could have used our decoder network to perform both "ordinary" and "adversarial" sample classification using Eqn. 9, but this process involves an iterative optimization. Hence, we only use it for the detected adversarial samples. The results are summarized in Table 2.

ϵ	0.06	0.12	0.18	0.24	0.30
Accuracy	97%	93%	91%	87%	87%

Table 2: We present the reclassification accuracy for samples generated using FGSM on the MNIST dataset.

Following the same reclassification scheme, we also find that the method is able to correctly classify rejected test samples, thereby improving the overall accuracy achieved by the proposed method. For example, among the 181 samples rejected by the supervised model for the MNIST test dataset (as per Table 1), 110 samples are now correctly classified, improving the accuracy to 99.07%.

Entropy of $Q_\phi(\mathbf{z}|\mathbf{x})$. To compare the performance of the proposed network with the corresponding network with variable entropy of $Q_\phi(\mathbf{z}|\mathbf{x})$, we ran experiments by letting $H(Q_\phi(\mathbf{z}|\mathbf{x}))$ to be variable, and keeping all other parameters same. We tried the FGSM attack against the encoder of the model thus obtained, and observed that the adversarial sample detection capability of the network reduces drastically. This is justified by the fact that the reconstructions tend to be blurry in this case, thereby leading to a high reconstruction threshold. The results are shown in figure 5. In order to further study the difference between the two cases, we train both variants of the network on the CelebA dataset, and observe that the "Fréchet Inception Distance (FID) (2017) score is significantly better for the model with a constant $H(Q_\phi(\mathbf{z}|\mathbf{x}))$ (50.4) than the one with variable $H(Q_\phi(\mathbf{z}|\mathbf{x}))$ (58.3). The FID scores are obtained by randomly sampling $10,000$ points from the latent distribution, and comparing the distribution of the images generated from the these points with the training image distribution.

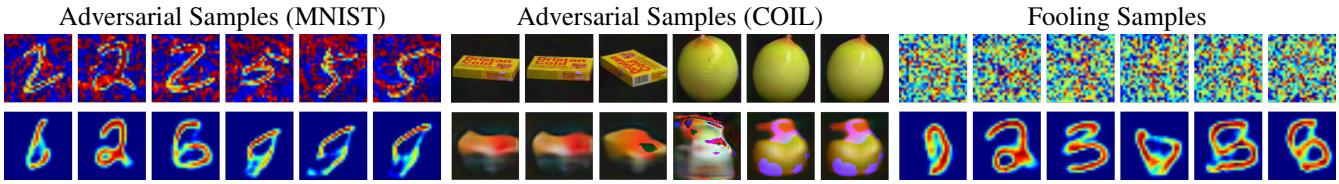

Figure 4: White box attack on MNIST and COIL dataset. (a) Targeted attack on MNIST. (b) Targeted attack on COIL. (c) Targeted fooling sample attack on MNIST. The first row represents the images to which the white-box attack converged, and the second row represents the corresponding reconstructed images.

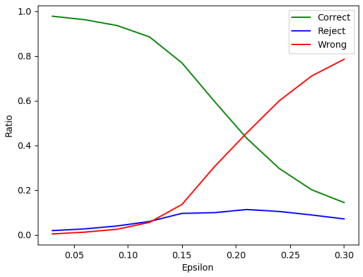

Figure 5: We run FGSM with varying ϵ on the model with variable encoder distribution entropy, trained on MNIST data. The rejection rate stays low in this case, while the error rate increases with increasing ϵ.

5 Discussion

In this work, we have successfully demonstrated how a generative model can be used to gain defensive strength against adversarial attacks on images of relatively high resolution (128x128 for the COIL-100 dataset for example). However, the proposed network is limited by the generative capability of VAE based architectures, and thus, might not scale effectively to ImageNet scale datasets (2009). In spite of this fact, keeping the underlying principles for adversarial sample detection and reclassification as described in this work, recent advances in invertible generative models such as Glow (2018) can be exploited to scale to more complex datasets. Further, as discussed earlier, the problem of defending against adversarial attacks still remains an unsolved problem even for datasets with more structured images. Hence our method can be used for practical applications such as secure medical image classification (2018), biometrics identification, etc.

Human perception involves both discriminative and generative capabilities. Similarly, our work proposes a modification to VAEs to incorporate discriminative ability, besides using its generative ability to gain robustness against adversarial samples. The input space dimensionality (to the decoder) is drastically smaller compared to the input space dimensionality of image classifiers. Hence, it is much easier to attain dense coverage in the latent space, thereby minimizing the possibility of the occurrence of holes, leading to defensive capability against both adversarial and fooling images. With our construct, selective classification and semi-supervised learning become feasible under the same framework. A possible direction of future research would be to study how effectively the proposed approach can be scaled to more complex datasets by using recently proposed invertible generative modeling techniques.

6 Acknowledgement

We are extremely grateful to Mr. Arnav Acharyya for his invaluable contribution to the discussions that helped shape this work.

References

Athalye, A.; Carlini, N.; and Wagner, D. A. 2018. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, 274–283.

Carlini, N., and Wagner, D. 2016. Defensive distillation is not robust to adversarial examples. *arXiv preprint arXiv:1607.04311*.

Carlini, N., and Wagner, D. 2017a. Magnet and" efficient defenses against adversarial attacks" are not robust to adversarial examples. *arXiv preprint arXiv:1711.08478*.

Carlini, N., and Wagner, D. 2017b. Towards evaluating the robustness of neural networks. In *Security and Privacy (SP), 2017 IEEE Symposium on*, 39–57. IEEE.

Chen, P.-Y.; Zhang, H.; Sharma, Y.; Yi, J.; and Hsieh, C.-J. 2017. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, 15–26. ACM.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*.

Dong, Y.; Liao, F.; Pang, T.; Su, H.; Zhu, J.; Hu, X.; and Li, J. 2018. Boosting adversarial attacks with momentum. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Eykholt, K.; Evtimov, I.; Fernandes, E.; Li, B.; Rahmati, A.; Xiao, C.; Prakash, A.; Kohno, T.; and Song, D. 2018. Robust Physical-World Attacks on Deep Learning Visual Classification. In *Computer Vision and Pattern Recognition (CVPR)*.

Finlayson, S. G.; Kohane, I. S.; and Beam, A. L. 2018. Adversarial attacks against medical deep learning systems. *arXiv preprint arXiv:1804.05296*.

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

Gu, S., and Rigazio, L. 2014. Towards deep neural network architectures robust to adversarial examples. *arXiv preprint arXiv:1412.5068*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, 1026–1034.

Heusel, M.; Ramsauer, H.; Unterthiner, T.; Nessler, B.; and Hochreiter, S. 2017. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems*, 6626–6637.

Huang, G.; Liu, Z.; Weinberger, K. Q.; and van der Maaten, L. 2017. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, volume 1, 3.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kingma, D. P., and Dhariwal, P. 2018. Glow: Generative flow with invertible 1x1 convolutions. *arXiv preprint arXiv:1807.03039*.

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

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324.

Lee, C.-Y.; Gallagher, P. W.; and Tu, Z. 2016. Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree. In *Artificial Intelligence and Statistics*, 464–472.

Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2018. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*.

Meng, D., and Chen, H. 2017a. Magnet: A two-pronged defense against adversarial examples. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, 135–147. New York, NY, USA: ACM.

Meng, D., and Chen, H. 2017b. Magnet: a two-pronged defense against adversarial examples. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 135–147. ACM.

Moosavi Dezfooli, S. M.; Fawzi, A.; and Frossard, P. 2016. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, number EPFL-CONF-218057.

Nayar, S.; Nene, S.; and Murase, H. 1996. Columbia object image library (coil 100). *Department of Comp. Science, Columbia University, Tech. Rep. CUCS-006-96*.

Netzer, Y.; Wang, T.; Coates, A.; Bissacco, A.; Wu, B.; and Ng, A. Y. 2011. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, volume 2011, 5.

Nguyen, A.; Yosinski, J.; and Clune, J. 2015. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 427–436.

Nicolas Papernot, Nicholas Carlini, I. G. R. F. F. F. A. M. K. H. Y.-L. J. A. K. R. S. A. G. Y.-C. L. 2017. cleverhans v2.0.0: an adversarial machine learning library. *arXiv preprint arXiv:1610.00768*.

Papernot, N.; McDaniel, P.; Jha, S.; Fredrikson, M.; Celik, Z. B.; and Swami, A. 2016a. The limitations of deep learning in adversarial settings. In *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, 372–387. IEEE.

Papernot, N.; McDaniel, P.; Wu, X.; Jha, S.; and Swami, A. 2016b. Distillation as a defense to adversarial perturbations against deep neural networks. In *Security and Privacy (SP), 2016 IEEE Symposium on*, 582–597. IEEE.

Papernot, N.; McDaniel, P.; Goodfellow, I.; Jha, S.; Celik, Z. B.; and Swami, A. 2017. Practical black-box attacks against machine learning. In *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*, 506–519. ACM.

Papernot, N.; McDaniel, P.; and Goodfellow, I. 2016. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*.

Pouya Samangouei, Maya Kabkab, R. C. 2018. Defense-GAN: Protecting classifiers against adversarial attacks using generative models. *International Conference on Learning Representations*.

Song, Y.; Kim, T.; Nowozin, S.; Ermon, S.; and Kushman, N. 2018a. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. In *International Conference on Learning Representations*.

Song, Y.; Shu, R.; Kushman, N.; and Ermon, S. 2018b. Generative adversarial examples. In *Advances in Neural Information Processing Systems (NIPS)*.

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; and Fergus, R. 2014. Intriguing properties of neural networks. In *International Conference on Learning Representations*.

Wan, L.; Zeiler, M.; Zhang, S.; Le Cun, Y.; and Fergus, R. 2013. Regularization of neural networks using dropconnect. In *International Conference on Machine Learning*, 1058–1066.

Wu, D.; Wu, J.; Zeng, R.; Jiang, L.; Senhadji, L.; and Shu, H. 2015. Kernel principal component analysis network for image classification. *arXiv preprint arXiv:1512.06337*.

Zhao, Z.; Dua, D.; and Singh, S. 2018. Generating natural adversarial examples. In *International Conference on Learning Representations*.