DEEP VARIATIONAL INFORMATION BOTTLENECK

Alexander A. Alemi, Ian Fischer, Joshua V. Dillon, Kevin Murphy
Google Research
{alemi,iansf,jvdillon,kpmurphy}@google.com

ABSTRACT

We present a variational approximation to the information bottleneck of Tishby et al. (1999). This variational approach allows us to parameterize the information bottleneck model using a neural network and leverage the reparameterization trick for efficient training. We call this method "Deep Variational Information Bottleneck", or Deep VIB. We show that models trained with the VIB objective outperform those that are trained with other forms of regularization, in terms of generalization performance and robustness to adversarial attack.

1 INTRODUCTION

We adopt an information theoretic view of deep networks. We regard the internal representation of some intermediate layer as a stochastic encoding Z of the input source X, defined by a parametric encoder $p(\mathbf{z}|\mathbf{x};\boldsymbol{\theta})$.[1] Our goal is to learn an encoding that is maximally informative about our target Y, measured by the mutual information between our encoding and the target $I(Z,Y;\boldsymbol{\theta})$, where

$$I(Z,Y;\boldsymbol{\theta}) = \int dx\; dy\; p(z,y|\boldsymbol{\theta}) \log \frac{p(z,y|\boldsymbol{\theta})}{p(z|\boldsymbol{\theta})p(y|\boldsymbol{\theta})}.^{2} \tag{1}$$

Given the data processing inequality, and the invariance of the mutual information to reparameterizations, if this was our only objective we could always ensure a maximally informative representation by taking the identity encoding of our data ($Z = X$), but this is not a useful representation of our data. Instead we would like to find the best representation we can obtain subject to a constraint on its complexity. A natural and useful constraint to apply is on the mutual information between our encoding and the original data, $I(X,Z) \leq I_c$, where I_c is the information constraint. This suggests the objective:

$$\max_{\boldsymbol{\theta}} I(Z,Y;\boldsymbol{\theta}) \text{ s.t. } I(X,Z;\boldsymbol{\theta}) \leq I_c\,. \tag{2}$$

Equivalently, with the introduction of a Lagrange multiplier β, we can maximize the objective function

$$R_{IB}(\boldsymbol{\theta}) = I(Z,Y;\boldsymbol{\theta}) - \beta I(Z,X;\boldsymbol{\theta}). \tag{3}$$

Here our goal is to learn an encoding Z that is maximally expressive about Y while being maximally compressive about X, where $\beta \geq 0$ controls the tradeoff.[3] This approach is known as the information bottleneck (IB), and was first proposed in Tishby et al. (1999). Intuitively, the first term in R_{IB} encourages Z to be predictive of Y; the second term encourages Z to "forget" X. Essentially it forces Z to act like a minimal sufficient statistic of X for predicting Y.

The IB principle is appealing, since it defines what we mean by a good representation, in terms of the fundamental tradeoff between having a concise representation and one with good predictive power (Tishby & Zaslavsky, 2015a). The main drawback of the IB principle is that computing mutual information is, in general, computationally challenging. There are two notable exceptions: the first

[1] In this work, X, Y, Z are random variables, x, y, z and $\mathbf{x}, \mathbf{y}, \mathbf{z}$ are instances of random variables, and $F(\cdot;\boldsymbol{\theta})$ and $f(\cdot;\boldsymbol{\theta})$ are functionals or functions parameterized by $\boldsymbol{\theta}$.

[2] Note that in the present discussion, Y is the ground truth label which is independent of our parameters so $p(y|\boldsymbol{\theta}) = p(y)$.

[3] Note that, in our notation, large β results in a highly compressed representation. In some works, the IB principle is formulated as the minimization of $I(Z,X) - \beta I(Z,Y)$, in which case large β corresponds to high mutual information between Z and Y, and hence low compression.

is when X, Y and Z are all discrete, as in Tishby et al. (1999); this can be used to cluster discrete data, such as words. The second case is when X, Y and Z are all jointly Gaussian (Chechik et al., 2005). However, these assumptions both severely constrain the class of learnable models.

In this paper, we propose to use variational inference to construct a lower bound on the IB objective in Equation 3. We call the resulting method VIB (variational information bottleneck). By using the reparameterization trick (Kingma & Welling, 2014), we can use Monte Carlo sampling to get an unbiased estimate of the gradient, and hence we can optimize the objective using stochastic gradient descent. This allows us to use deep neural networks to parameterize our distributions, and thus to handle high-dimensional, continuous data, such as images, avoiding the previous restrictions to the discrete or Gaussian cases.

We also show, by a series of experiments, that stochastic neural networks, fit using our VIB method, are robust to overfitting, since VIB finds a representation Z which ignores as many details of the input X as possible. In addition, they are more robust to adversarial inputs than deterministic models which are fit using (penalized) maximum likelihood estimation. Intuitively this is because each input image gets mapped to a distribution rather than a unique Z, so it is more difficult to pass small, idiosyncratic perturbations through the latent bottleneck.

2 Related work

The idea of using information theoretic objectives for deep neural networks was pointed out in Tishby & Zaslavsky (2015b). However, they did not include any experimental results, since their approach for optimizing the IB objective relied on the iterative Blahut Arimoto algorithm, which is infeasible to apply to deep neural networks.

Variational inference is a natural way to approximate the problem. Variational bounds on mutual information have previously been explored in Agakov (2004), though not in conjunction with the information bottleneck objective. Mohamed & Rezende (2015) also explore variational bounds on mutual information, and apply them to deep neural networks, but in the context of reinforcement learning. We recently discovered Chalk et al. (2016), who independently developed the same variational lower bound on the IB objective as us. However, they apply it to sparse coding problems, and use the kernel trick to achieve nonlinear mappings, whereas we apply it to deep neural networks, which are computationally more efficient. In addition, we are able to handle large datasets by using stochastic gradient descent, whereas they use batch variational EM.

In the supervised learning literature, our work is related to the recently proposed confidence penalty (entropy regularization) method of (Pereyra et al., 2016). In this work, they fit a deterministic network by optimizing an objective that combines the usual cross entropy loss with an extra term which penalizes models for having low entropy predictive distributions. In more detail, their cost function has the form

$$J_{CP} = \frac{1}{N}\sum_{n=1}^{N}\left[H(p(y|y_n), p(y|x_n)) - \beta H(p(y|x_n))\right] \tag{4}$$

where $H(p,q) = -\sum_y p(y)\log q(y)$ is the cross entropy, $H(p) = H(p,p)$ is the entropy, $p(y|y_n) = \delta_{y_n}(y)$ is a one-hot encoding of the label y_n, and N is the number of training examples. (Note that setting $\beta = 0$ corresponds to the usual maximum likelihood estimate.) In (Pereyra et al., 2016) they show that CP performs better than the simpler technique of label smoothing, in which we replace the zeros in the one-hot encoding of the labels by $\epsilon > 0$, and then renormalize so that the distribution still sums to one. We will compare our VIB method to both the confidence penalty method and label smoothing in Section 4.1.

In the unsupervised learning literature, our work is closely related to the work in Kingma & Welling (2014) on variational autoencoders. In fact, their method is a special case of an unsupervised version of the VIB, but with the β parameter fixed at 1.0, as we explain in Appendix B. The VAE objective, but with different values of β, was also explored in Higgins et al. (2016), but from a different perspective.

The method of Wang et al. (2016b) proposes a latent variable generative model of both x and y; their variational lower bound is closely related to ours, with the following differences. First, we do

not have a likelihood term for x, since we are in the discriminative setting. Second, they fix $\beta = 1$, since they do not consider compression.

Finally, the variational fair autoencoder of Louizos et al. (2016) shares with our paper the idea of ignoring parts of the input. However, in their approach, the user must specify which aspects of the input (the so-called "sensitive" parts) to ignore, whereas in our method, we can discover irrelevant parts of the input automatically.

3 METHOD

Following standard practice in the IB literature, we assume that the joint distribution $p(X, Y, Z)$ factors as follows:

$$p(X, Y, Z) = p(Z|X, Y)p(Y|X)p(X) = p(Z|X)p(Y|X)p(X) \tag{5}$$

i.e., we assume $p(Z|X, Y) = p(Z|X)$, corresponding to the Markov chain $Y \leftrightarrow X \leftrightarrow Z$. This restriction means that our representation Z cannot depend directly on the labels Y. (This opens the door to unsupervised representation learning, which we will discuss in Appendix B.) Besides the structure in the joint data distribution $p(X, Y)$, the only content at this point is our model for the stochastic encoder $p(Z|X)$, all other distributions are fully determined by these and the Markov chain constraint.

Recall that the IB objective has the form $I(Z, Y) - \beta I(Z, X)$. We will examine each of these expressions in turn. Let us start with $I(Z, Y)$. Writing it out in full, this becomes

$$I(Z, Y) = \int dy\, dz\, p(y, z) \log \frac{p(y, z)}{p(y)p(z)} = \int dy\, dz\, p(y, z) \log \frac{p(y|z)}{p(y)}\,. \tag{6}$$

where $p(y|z)$ is fully defined by our encoder and Markov Chain as follows:

$$p(y|z) = \int dx\, p(x, y|z) = \int dx\, p(y|x)p(x|z) = \int dx\, \frac{p(y|x)p(z|x)p(x)}{p(z)}. \tag{7}$$

Since this is intractable in our case, let $q(y|z)$ be a variational approximation to $p(y|z)$. This is our decoder, which we will take to be another neural network with its own set of parameters. Using the fact that the Kullback Leibler divergence is always positive, we have

$$\mathrm{KL}[p(Y|Z), q(Y|Z)] \geq 0 \implies \int dy\, p(y|z) \log p(y|z) \geq \int dy\, p(y|z) \log q(y|z)\,, \tag{8}$$

and hence

$$I(Z, Y) \geq \int dy\, dz\, p(y, z) \log \frac{q(y|z)}{p(y)} \tag{9}$$

$$= \int dy\, dz\, p(y, z) \log q(y|z) - \int dy\, p(y) \log p(y) \tag{10}$$

$$= \int dy\, dz\, p(y, z) \log q(y|z) + H(Y)\,. \tag{11}$$

Notice that the entropy of our labels $H(Y)$ is independent of our optimization procedure and so can be ignored.

Focusing on the first term in Equation 11, we can rewrite $p(y, z)$ as $p(y, z) = \int dx\, p(x, y, z) = \int dx\, p(x)p(y|x)p(z|x)$ (leveraging our Markov assumption), which gives us a new lower bound on the first term of our objective:

$$I(Z, Y) \geq \int dx\, dy\, dz\, p(x)p(y|x)p(z|x) \log q(y|z)\,. \tag{12}$$

This only requires samples from both our joint data distribution as well as samples from our stochastic encoder, while it requires we have access to a tractable variational approximation in $q(y|z)$.

We now consider the term $\beta I(Z, X)$:

$$I(Z, X) = \int dz\, dx\, p(x, z) \log \frac{p(z|x)}{p(z)} = \int dz\, dx\, p(x, z) \log p(z|x) - \int dz\, p(z) \log p(z)\,. \tag{13}$$

In general, while it is fully defined, computing the marginal distribution of Z, $p(z) = \int dx\, p(z|x)p(x)$, might be difficult. So let $r(z)$ be a variational approximation to this marginal. Since $\mathrm{KL}[p(Z), r(Z)] \geq 0 \implies \int dz\, p(z) \log p(z) \geq \int dz\, p(z) \log r(z)$, we have the following upper bound:

$$I(Z, X) \leq \int dx\, dz\, p(x)p(z|x) \log \frac{p(z|x)}{r(z)}\,. \tag{14}$$

Combining both of these bounds we have that

$$\begin{aligned} I(Z, Y) - \beta I(Z, X) \geq & \int dx\, dy\, dz\, p(x)p(y|x)p(z|x) \log q(y|z) \\ & - \beta \int dx\, dz\, p(x)p(z|x) \log \frac{p(z|x)}{r(z)} = L\,. \end{aligned} \tag{15}$$

We now discuss how to compute the lower bound L in practice. We can approximate $p(x, y) = p(x)p(y|x)$ using the empirical data distribution $p(x, y) = \frac{1}{N} \sum_{n=1}^{N} \delta_{x_n}(x)\delta_{y_n}(y)$, and hence we can write

$$L \approx \frac{1}{N} \sum_{n=1}^{N} \left[\int dz\, p(z|x_n) \log q(y_n|z) - \beta\, p(z|x_n) \log \frac{p(z|x_n)}{r(z)} \right]. \tag{16}$$

Suppose we use an encoder of the form $p(z|x) = \mathcal{N}(z|f_e^{\mu}(x), f_e^{\Sigma}(x))$, where f_e is an MLP which outputs both the K-dimensional mean μ of z as well as the $K \times K$ covariance matrix Σ. Then we can use the reparameterization trick (Kingma & Welling, 2014) to write $p(z|x)dz = p(\epsilon)d\epsilon$, where $z = f(x, \epsilon)$ is a deterministic function of x and the Gaussian random variable ϵ. This formulation has the important advantage that the noise term is independent of the parameters of the model, so it is easy to take gradients.

Assuming our choice of $p(z|x)$ and $r(z)$ allows computation of an analytic Kullback-Leibler divergence, we can put everything together to get the following objective function, which we try to minimize:

$$J_{IB} = \frac{1}{N} \sum_{n=1}^{N} \mathbb{E}_{\epsilon \sim p(\epsilon)} \left[-\log q(y_n|f(x_n, \epsilon)) \right] + \beta\, \mathrm{KL}\left[p(Z|x_n), r(Z) \right]. \tag{17}$$

As in Kingma & Welling (2014), this formulation allows us to directly backpropagate through a single sample of our stochastic code and ensure that our gradient is an unbiased estimate of the true expected gradient.[4]

4 Experimental results

In this section, we present various experimental results, comparing the behavior of standard deterministic networks to stochastic neural networks trained by optimizing the VIB objective.

4.1 Behavior on MNIST

We start with experiments on unmodified MNIST (i.e. no data augmentation). In order to pick a model with some "headroom" to improve, we decided to use the same architecture as in the (Pereyra et al., 2016) paper, namely an MLP with fully connected layers of the form 784 - 1024 - 1024 - 10, and ReLu activations. (Since we are not exploiting spatial information, this correpsonds to the "permutation invariant" version of MNIST.) The performance of this baseline is 1.38% error. (Pereyra et al., 2016) were able to improve this to 1.17% using their regularization technique. We were able to improve this to 1.13% using our technique, as we explain below.

In our method, the stochastic encoder has the form $p(z|x) = \mathcal{N}(z|f_e^{\mu}(x), f_e^{\Sigma}(x))$, where f_e is an MLP of the form $784 - 1024 - 1024 - 2K$, where K is the size of the bottleneck. The first K outputs from f_e encode μ, the remaining K outputs encode σ (after a softplus transform).

[4] Even if our choice of encoding distribution and variational prior do not admit an analytic KL, we could similarly reparameterize through a sample of the divergence (Kingma & Welling, 2014; Blundell et al., 2015).

Model	error
Baseline	1.38%
Dropout	1.34%
Dropout (Pereyra et al., 2016)	1.40%
Confidence Penalty	1.36%
Confidence Penalty (Pereyra et al., 2016)	1.17%
Label Smoothing	1.40%
Label Smoothing (Pereyra et al., 2016)	1.23%
VIB ($\beta = 10^{-3}$)	**1.13%**

Table 1: Test set misclassification rate on permutation-invariant MNIST using $K = 256$. We compare our method (VIB) to an equivalent deterministic model using various forms of regularization. The discrepancy between our results for confidence penalty and label smoothing and the numbers reported in (Pereyra et al., 2016) are due to slightly different hyperparameters.

The decoder is a simple logistic regression model of the form $q(y|z) = \mathcal{S}(y|f_d(z))$, where $\mathcal{S}(a) = [\exp(a_c)/\sum_{c'=1}^{C}\exp(a_{c'})]$ is the softmax function, and $f_d(z) = Wz + b$ maps the K dimensional latent code to the logits of the $C = 10$ classes. (In later sections, we consider more complex decoders, but here we wanted to show the benefits of VIB in a simple setting.)

Finally, we treat $r(z)$ as a fixed K-dimensional spherical Gaussian, $r(z) = \mathcal{N}(z|0, I)$.

We compare our method to the baseline MLP. We calso consider the following deterministic limit of our model, when $\beta = 0$. In this case, we obtain the following objective function:

$$J_{IB0} = -\frac{1}{N}\sum_{n=1}^{N} E_{z\sim\mathcal{N}(f_e^{\mu}(x_n), f_e^{\Sigma}(x_n))}\left[\log \mathcal{S}(y_n|f_d(z))\right] \tag{18}$$

When $\beta \to 0$, we observe the VIB optimization process tends to make $f_e^{\Sigma}(x) \to 0$, so the network becomes nearly deterministic. In our experiments we also train an explicitly deterministic model that has the same form as the stochastic model, except that we just use $z = f_e^{\mu}(x)$ as the hidden encoding, and drop the Gaussian layer.

4.1.1 HIGHER DIMENSIONAL EMBEDDING

To demonstrate that our VIB method can achieve competitive classification results, we compared against a deterministic MLP trained with various forms of regularization. We use a $K = 256$ dimensional bottleneck and a diagonal Gaussian for $p(z|x)$. The networks were trained using TensorFlow for 200 epochs using the Adam optimizer (Kingma & Ba, 2015) with a learning rate of 0.0001. Full hyperparameter details can be found in Appendix A.

The results are shown in Table 1. We see that we can slightly outperform other forms of regularization that have been proposed in the literature while using the same network for each. Of course, the performance varies depending on β. These results are not state of the art, nor is our main focus of our work to suggest that VIB is the best regularization method by itself, which would require much more experimentation. However, using the same architecture for each experiment and comparing to VIB as the only source of regularization suggests VIB works as a decent regularizer in and of itself. Figure 1(a) plots the train and test error vs β, averaged over 5 trials (with error bars) for the case where we use a single Monte Carlo sample of z when predicting, and also for the case where we average over 12 posterior samples (i.e., we use $p(y|x) = \frac{1}{S}\sum_{s=1}^{S} q(y|z^s)$ for $z^s \sim p(z|x)$, where $S = 12$). In our own investigations, a dozen samples seemed to be sufficient to capture any additional benefit the stochastic evaluations had to offer in this experiment[5].

We see several interesting properties in Figure 1(a). First, we notice that the error rate shoots up once β rises above the critical value of $\beta \sim 10^{-2}$. This corresponds to a setting where the mutual information between X and Z is less than $\log_2(10)$ bits, so the model can no longer represent the fact that there are 10 different classes. Second, we notice that, for small values of β, the test error

[5] A dozen samples wasn't chosen for any particular reason, except the old addage that a dozen samples are sufficient, as mirrored in David MacKay's book (MacKay, 2003). They proved sufficient in this case.

is higher than the training error, which indicates that we are overfitting. This is because the network learns to be more deterministic, forcing $\sigma \approx 0$, thus reducing the benefits of regularization. Third, we notice that for intermediate values of β, Monte Carlo averaging helps. Interestingly, the region with the best performance roughly corresponds to where the added benefit from stochastic averaging goes away, suggesting an avenue by which one could try to optimize β using purely statistics on the training set without a validation set. We have not extensively studied this possibility yet.

In Figure 1(c), we plot the IB curve, i.e., we plot $I(Z, Y)$ vs $I(Z, X)$ as we vary β. As we allow more information from the input through to the bottleneck (by lowering β), we increase the mutual information between our embedding and the label on the training set, but not necessarily on the test set, as is evident from the plot.

In Figure 1(d) we plot the second term in our objective, the upper bound on the mutual information between the images X and our stochastic encoding Z, which in our case is simply the relative entropy between our encoding and the fixed isotropic unit Gaussian prior. Notice that the y-axis is a logarithmic one. This demonstrates that our best results (when β is between 10^{-3} and 10^{-2}) occur where the mutual information between the stochastic encoding and the images is on the order of 10 to 100 bits.

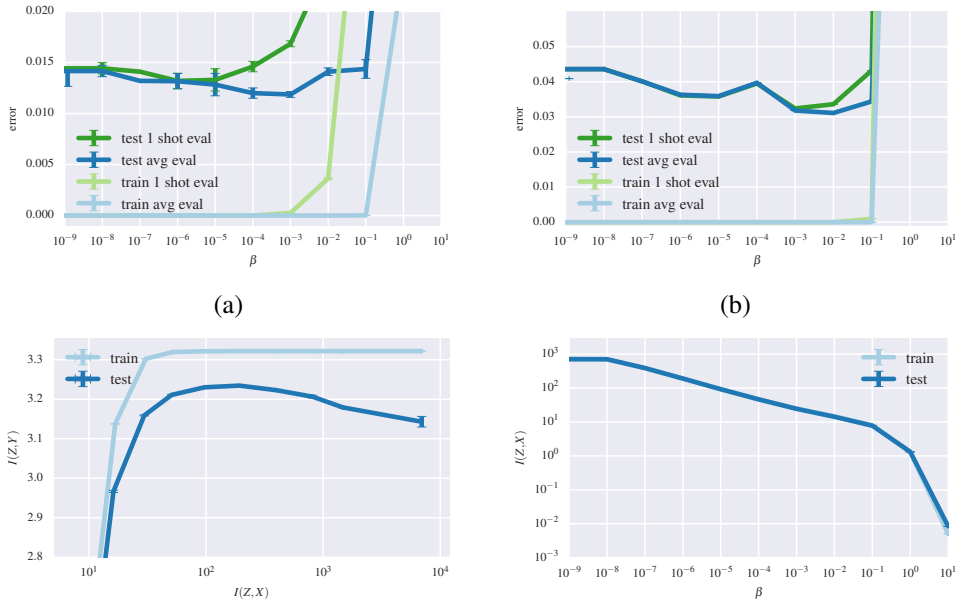


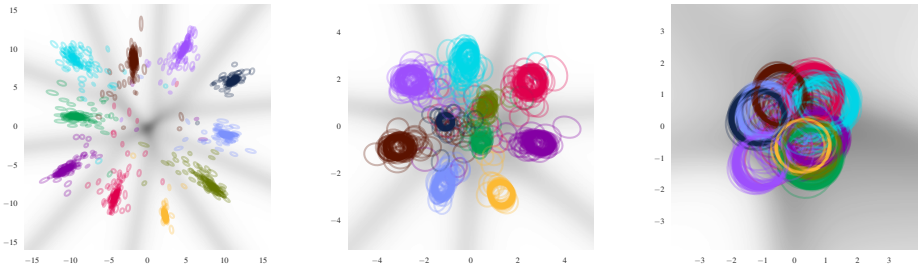
Figure 1: Results of VIB model on MNIST. (a) Error rate vs β for $K = 256$ on train and test set. "1 shot eval" means a single posterior sample of z, "avg eval" means 12 Monte Carlo samples. The spike in the error rate at $\beta \sim 10^{-2}$ corresponds to a model that is too highly regularized. Plotted values are the average over 5 independent training runs at each β. Error bars show the standard deviation in the results. (b) Same as (a), but for $K = 2$. Performance is much worse, since we pass through a very narrow bottleneck. (c) $I(Z, Y)$ vs $I(Z, X)$ as we vary β for $K = 256$. We see that increasing $I(Z, X)$ helps training set performance, but can result in overfitting. (d) $I(Z, X)$ vs β for $K = 256$. We see that for a good value of β, such as 10^{-2}, we only need to store about 10 bits of information about the input.

4.1.2 TWO DIMENSIONAL EMBEDDING

To better understand the behavior of our method, we refit our model to MNIST using a $K = 2$ dimensional bottleneck, but using a full covariance Gaussian. (The neural net predicts the mean and the Cholesky decomposition of the covariance matrix.) Figure 1(b) shows that, not surprisingly, the classification performance is worse (note the different scaled axes), but the overall trends are the

same as in the $K = 256$ dimensional case. The IB curve (not shown) also has a similar shape to before, except now the gap between training and testing is even larger.

Figure 2 provides a visualization of what the network is doing. We plot the posteriors $p(z|x)$ as a 2d Gaussian ellipse (representing the 95% confidence region) for 1000 images from the test set. Colors correspond to the true class labels. In the background of each plot is the entropy of the variational classifier $q(y|z)$ evaluated at that point.



(a) $\beta = 10^{-3}$, $\text{err}_{\text{mc}} = 3.18\%$, $\text{err}_1 = 3.24\%$ (b) $\beta = 10^{-1}$, $\text{err}_{\text{mc}} = 3.44\%$, $\text{err}_1 = 4.32\%$ (c) $\beta = 10^{0}$, $\text{err}_{\text{mc}} = 33.82\%$, $\text{err}_1 = 62.81\%$.

Figure 2: Visualizing embeddings of 1000 test images in two dimensions. We plot the 95% confidence interval of the Gaussian embedding $p(z|x) = \mathcal{N}(\mu, \Sigma)$ as an ellipse. The images are colored according to their true class label. The background greyscale image denotes the entropy of the variational classifier evaluated at each two dimensional location. As β becomes larger, we forget more about the input and the embeddings start to overlap to such a degree that the classes become indistinguishable. We also report the test error using a single sample, err_1, and using 12 Monte Carlo samples, err_{mc}. For "good" values of β, a single sample suffices.

We see several interesting properties. First, as β increases (so we pass less information through), the embedding covariances increase in relation to the distance between samples, and the classes start to overlap. Second, once β passes a critical value, the encoding "collapses", and essentially all the class information is lost. Third, there is a fair amount of uncertainty in the class preditions ($q(y|z)$) in the areas between the class embeddings. Fourth, for intermediate values of β (say 10^{-1} in Figure 2(b)), predictive performance is still good, even though there is a lot of uncertainty about where any individual image will map to in comparison to other images in the same class. This means it would be difficult for an outside agent to infer which particular instance the model is representing, a property which we will explore more in the following sections.

4.2 Behavior on adversarial examples

Szegedy et al. (2013) was the first work to show that deep neural networks (and other kinds of classifiers) can be easily "fooled" into making mistakes by changing their inputs by imperceptibly small amounts. In this section, we will show how training with the VIB objective makes models significantly more robust to such adversarial examples.

4.2.1 Types of Adversaries

Since the initial work by Szegedy et al. (2013) and Goodfellow et al. (2014), many different adversaries have been proposed. Most attacks fall into three broad categories: optimization-based attacks (Szegedy et al., 2013; Carlini & Wagner, 2016; Moosavi-Dezfooli et al., 2016; Papernot et al., 2015; Robinson & Graham, 2015; Sabour et al., 2016), which directly run an optimizer such as L-BFGS or ADAM (Kingma & Ba, 2015) on image pixels to find a minimal perturbation that changes the model's classification; single-step gradient-based attacks (Goodfellow et al., 2014; Kurakin et al., 2016; Huang et al., 2015), which choose a gradient direction of the image pixels at some loss and then take a single step in that direction; and iterative gradient-based attacks (Kurakin et al., 2016),

which take multiple small steps along the gradient direction of the image pixels at some loss, recomputing the gradient direction after each step.[6]

Many adversaries can be formalized as either untargeted or targeted variants. An untargeted adversary can be defined as $A(X, M) \rightarrow X'$, where $A(.)$ is the adversarial function, X is the input image, X' is the adversarial example, and M is the target model. A is considered successful if $M(X) \neq M(X')$. Recently, Moosavi-Dezfooli et al. (2016) showed how to create a "universal" adversarial perturbation δ that can be added to any image X in order to make $M(X + \delta) \neq M(X)$ for a particular target model.

A targeted adversary can be defined as $A(X, M, l) \rightarrow X'$, where l is an additional target label, and A is only considered successful if $M(X') = l$.[7] Targeted attacks usually require larger magnitude perturbations, since the adversary cannot just "nudge" the input across the nearest decision boundary, but instead must force it into a desired decision region.

In this work, we focus on the Fast Gradient Sign (FGS) method proposed in Goodfellow et al. (2014) and the L_2 optimization method proposed in Carlini & Wagner (2016). FGS is a standard baseline attack that takes a single step in the gradient direction to generate the adversarial example. As originally described, FGS generates untargeted adversarial examples. On MNIST, Goodfellow et al. (2014) reported that FGS could generate adversarial examples that fooled a maxout network approximately 90% of the time with $\epsilon = 0.25$, where ϵ is the magnitude of the perturbation at each pixel. The L_2 optimization method has been shown to generate adversarial examples with smaller perturbations than any other method published to date, which were capable of fooling the target network 100% of the time. We consider both targeted attacks and untargeted attacks for the L_2 optimization method.[8]

4.2.2 Adversarial Robustness

There are multiple definitions of adversarial robustness in the literature. The most basic, which we shall use, is accuracy on adversarially perturbed versions of the test set, called adversarial examples.

It is also important to have a measure of the magnitude of the adversarial perturbation. Since adversaries are defined relative to human perception, the ideal measure would explicitly correspond to how easily a human observer would notice the perturbation. In lieu of such a measure, it is common to compute the size of the perturbation using L_0, L_1, L_2, and L_∞ norms (Szegedy et al., 2013; Goodfellow et al., 2014; Carlini & Wagner, 2016; Sabour et al., 2016). In particular, the L_0 norm measures the number of perturbed pixels, the L_2 norm measures the Euclidean distance between X and X', and the L_∞ norm measures the largest single change to any pixel.

4.2.3 Experimental Setup

We used the same model architectures as in Section 4.1, using a $K = 256$ bottleneck. The architectures included a deterministic (base) model trained by MLE; a deterministic model trained with dropout (the dropout rate was chosen on the validation set); and a stochastic model trained with VIB for various values of β.

For the VIB models, we use 12 posterior samples of Z to compute the class label distribution $p(y|x)$. This helps ensure that the adversaries can get a consistent gradient when constructing the perturbation, and that they can get a consistent evaluation when checking if the perturbation was successful

[6] There are also other adversaries that don't fall as cleanly into those categories, such as "fooling images" from Nguyen et al. (2014), which remove the human perceptual constraint, generating regular geometric patterns or noise patterns that networks confidently classify as natural images; and the idea of generating adversaries by stochastic search for images near the decision boundary of multiple networks from Baluja et al. (2015).

[7] Sabour et al. (2016) proposes a variant of the targeted attack, $A(X_S, M, X_T, k) \rightarrow X'_S$, where X_S is the source image, X_T is a target image, and k is a target layer in the model M. A produces X'_S by minimizing the difference in activations of M at layer k between X_T and X'_S. The end result of this attack for a classification network is still that $M(X'_S)$ yields a target label implicitly specified by X_T in a successful attack.

[8] Carlini & Wagner (2016) shared their code with us, which allowed us to perform the attack with exactly the same parameters they used for their paper, including the maximum number of iterations and maximum C value (see their paper for details).

(i.e., it reduces the chance that the adversary "gets lucky" in its perturbation due to an untypical sample). We also ran the VIB models in "mean mode", where the σs are forced to be 0. This had no noticeable impact on the results, so all reported results are for stochastic evaluation with 12 samples.

4.2.4 MNIST RESULTS AND DISCUSSION

We selected the first 10 zeros in the MNIST test set, and use the L_2 optimization adversary of Carlini & Wagner (2016) to try to perturb those zeros into ones.[9] Some sample results are shown in Figure 3. We see that the deterministic models are easily fooled by making small perturbations, but for the VIB models with reasonably large β, the adversary often fails to find an attack (indicated by the green borders) within the permitted number of iterations. Furthermore, when an attack is succesful, it needs to be much larger for the VIB models. To quantify this, Figure 4 plots the magnitude of the perturbation (relative to that of the deterministic and dropout models) needed for a successful attack as a function of β. As β increases, the L_0 norm of the perturbation decreases, but both L_2 and L_∞ norms increase, indicating that the adversary is being forced to put larger modifications into fewer pixels while searching for an adversarial perturbation.

Figure 5 plots the accuracy on FGS adversarial examples of the first 1000 images from the MNIST test set as a function of β. Each point in the plot corresponds to 3 separate executions of three different models trained with the same value of β. All models tested achieve over 98.4% accuracy on the unperturbed MNIST test set, so there is no appreciable measurement distortion due to underlying model accuracy.

Figure 6 plots the accuracy on L_2 optimization adversarial examples of the first 1000 images from the MNIST test set as a function of β. The same sets of three models per β were tested three times, as with the FGS adversarial examples.

We generated both untargeted and targeted adversarial examples for Figure 6. For targeting, we generate a random target label different from the source label in order to avoid biasing the results with unevenly explored source/target pairs. We see that for a reasonably broad range of β values, the VIB models have significantly better accuracy on the adversarial examples than the deterministic models, which have an accuracy of 0% (the L_2 optimization attack is very effective on traditional model architectures).

Figure 6 also reveals a surprising level of adversarial robustness even when $\beta \to 0$. This can be explained by the theoretical framework of Fawzi et al. (2016). Their work proves that quadratic classifiers (e.g., $x^\mathsf{T} Ax$, symmetric A) have a greater capacity for adversarial robustness than linear classifiers. As we show in Appendix C, our Gaussian/softmax encoder/decoder is approximately quadratic for all $\beta < \infty$.

4.2.5 IMAGENET RESULTS AND DISCUSSION

VIB improved classification accuracy and adversarial robustness for toy datasets like MNIST. We now investigate if VIB offers similar advantages for ImageNet, a more challenging natural image classification. Recall that ImageNet has approximately 1M images spanning 1K classes. We pre-process images such that they are 299x299 pixels.

Architecture

We make use of publicly available, pretrained checkpoints[10] of Inception Resnet V2 (Szegedy et al., 2016) on ImageNet (Deng et al., 2009). The checkpoint obtains 80.4% classification accuracy on the ImageNet validation set. Using the checkpoint, we transformed the original training set by applying the pretrained network to each image and extracting the representation at the penultimate layer. This new image representation has 1536 dimensions. The higher layers of the network continue to classify this representation with 80.4% accuracy; conditioned on this extraction the classification

[9] We chose this pair of labels since intuitively zeros and ones are the digits that are least similar in terms of human perception, so if the adversary can change a zero into a one without much human-noticeable perturbation, it is unlikely that the model has learned a representation similar to what humans learn.

[10] Available at the Tensorflow Models repository in the Slim directory: `https://github.com/tensorflow/models/tree/master/slim`

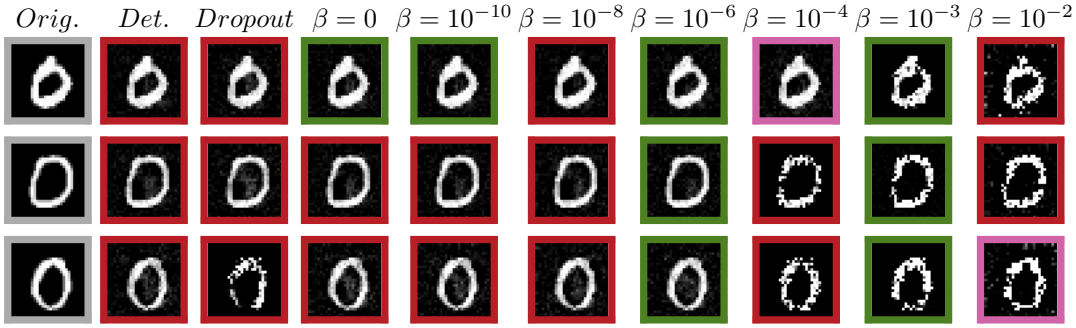
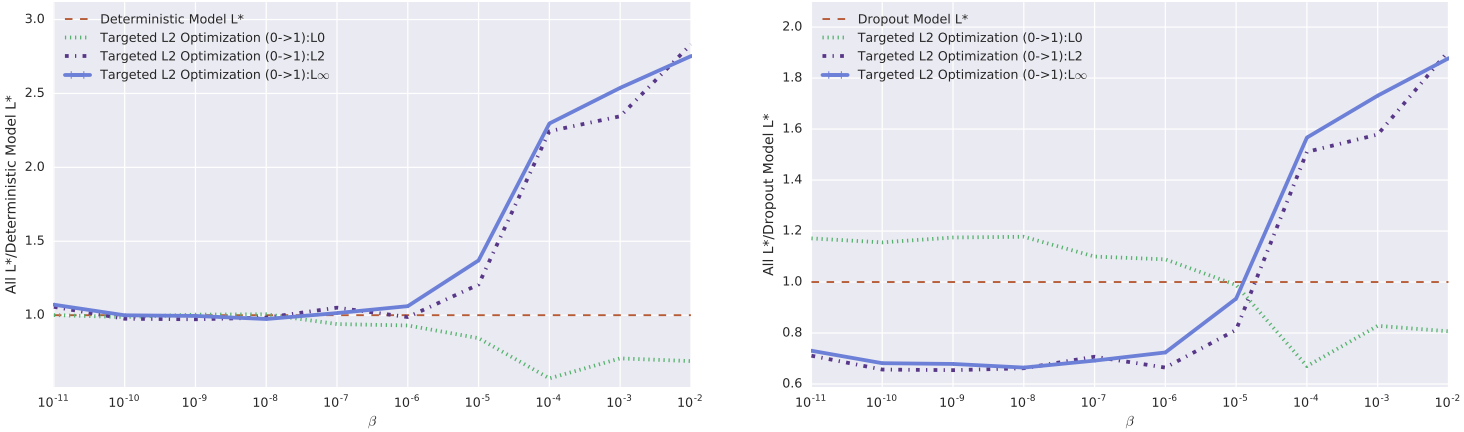
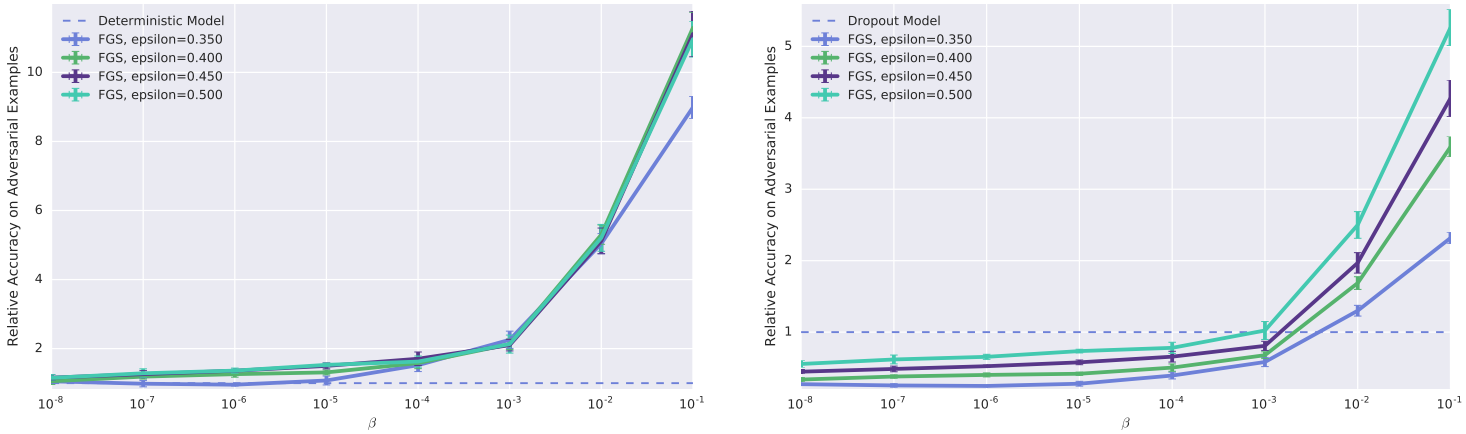


Figure 3: The adversary is trying to force each 0 to be classified as a 1. Successful attacks have a red background. Unsuccessful attacks have a green background. In the case that the label is changed to an incorrect label different from the target label (i.e., the classifier outputs something other than 0 or 1), the background is purple. The first column is the original image. The second column is adversarial examples targeting our deterministic baseline model. The third column is adversarial examples targeting our dropout model. The remaining columns are adversarial examples targeting our VIB models for different β.



(a) (b)

Figure 4: (a) Relative magnitude of the adversarial perturbation, measured using L_0, L_2, and L_∞ norms, for the images in Figure 3 as a function of β. (We normalize all values by the corresponding norm of the perturbation against the base model.) As β increases, L_0 decreases, but both L_2 and L_∞ increase, indicating that the adversary is being forced to put larger modifications into fewer pixels while searching for an adversarial perturbation. (b) Same as (a), but with the dropout model as the baseline. Dropout is more robust to the adversarial perturbations than the base deterministic model, but still performs much worse than the VIB model as β increases.



(a) (b)

Figure 5: Classification accuracy of VIB classifiers, divided by accuracy of baseline classifiers, on FGS-generated adversarial examples as a function of β. Higher is better, and the baseline is always at 1.0. For the FGS adversarial examples, when $\beta = 0$ (not shown), the VIB model's performance is almost identical to when $\beta = 10^{-8}$. (a) FGS accuracy normalized by the base deterministic model performance. The base deterministic model's accuracy on the adversarial examples ranges from about 1% when $\epsilon = 0.5$ to about 5% when $\epsilon = 0.35$. (b) Same as (a), but with the dropout model as the baseline. The dropout model is more robust than the base model, but less robust than VIB, particularly for stronger adversaries (i.e., larger values of ϵ). The dropout model's accuracy on the adversarial examples ranges from about 5% when $\epsilon = 0.5$ to about 16% when $\epsilon = 0.35$. As in the other results, relative performance is more dramatic as β increases, which seems to indicate that the VIB models are learning to ignore more of the perturbations caused by the FGS method, even though they were not trained on any adversarial examples.

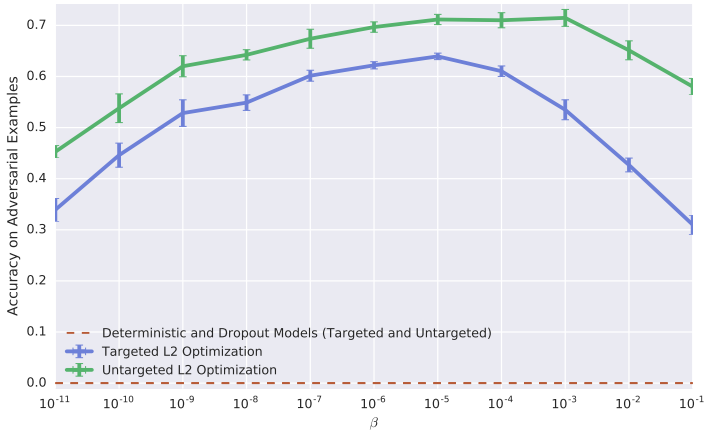
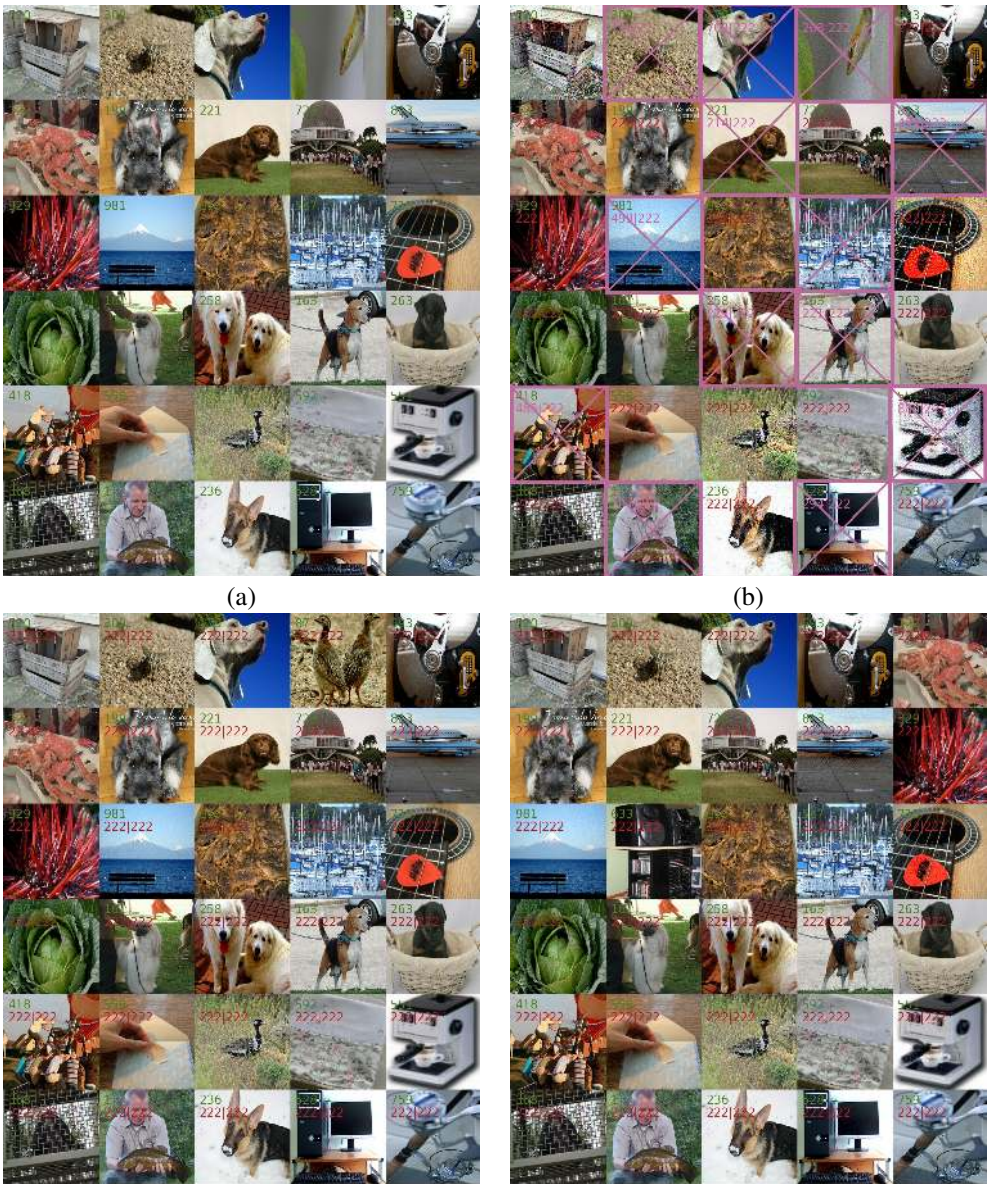


Figure 6: Classification accuracy (from 0 to 1) on L_2 adversarial examples (of all classes) as a function of β. The blue line is for targeted attacks, and the green line is for untargeted attacks (which are easier to resist). In this case, $\beta = 10^{-11}$ has performance indistinguishable from $\beta = 0$. The deterministic model and dropout model both have a classification accuracy of 0% in both the targeted and untargeted attack scenarios, indicated by the horizontal red dashed line at the bottom of the plot. This is the same accuracy on adversarial examples from this adversary reported in Carlini & Wagner (2016) on a convolutional network trained on MNIST.



(a) (b)

(c) (d)

Figure 7: The results of our ImageNet targeted L_2 optimization attack. In all cases we target a new label of 222 ("soccer ball"). Figure (a) shows the 30 images from the first 40 images in the ImageNet validation set that the VIB network classifies correctly. The class label is shown in green on each image. The predicted label and targeted label are shown in red. Figure (b) shows adversarial examples of the same images generated by attacking our VIB network with $\beta = 0.01$. While all of the attacks change the classification of the image, in 13 out of 30 examples the attack fails to hit the intended target class ("soccer ball"). Pink crosses denote cases where the attack failed to force the model to misclassify the image as a soccer ball. Figure (c) shows the same result but for our deterministic baseline operating on the whitened precomputed features. The attack always succeeds. Figure (d) is the same but for the original full Inception ResNet V2 network without modification. The attack always succeeds. There are slight variations in the set of adversarial examples shown for each network because we limited the adversarial search to correctly classified images. In the case of the deterministic baseline and original Inception ResNet V2 network, the perturbations are hardly noticable in the perturbed images, but in many instances, the perturbations for the VIB network can be percieved.

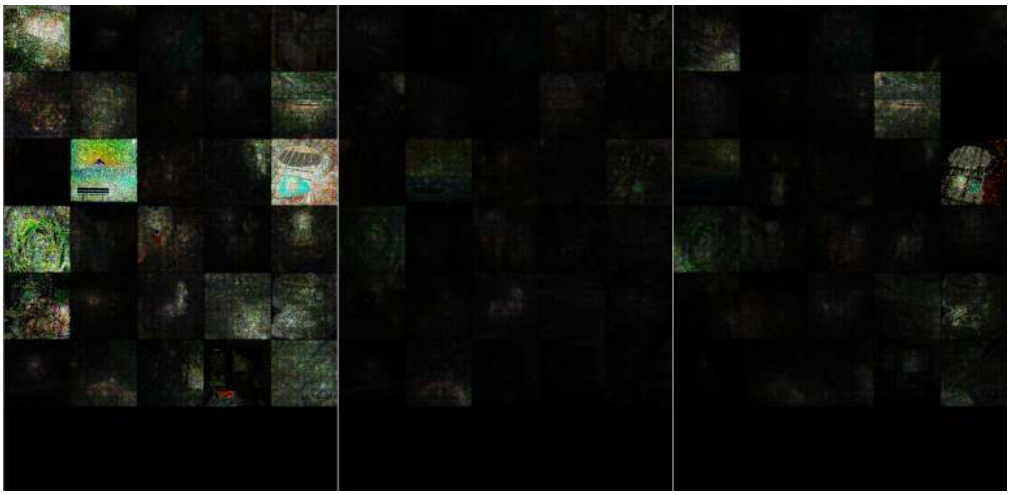


Figure 8: Shown are the absolute differences between the original and final perturbed images for all three networks. The left block shows the perturbations created while targeting the VIB network. The middle block shows the perturbations needed for the deterministic baseline using precomputed whitened features. The right block shows the perturbations created for the unmodified Inception ResNet V2 network. The contrast has been increased by the same amount in all three columns to emphasize the difference in the magnitude of the perturbations. The VIB network required much larger perturbations to confuse the classifier, and even then did not achieve the targeted class in 13 of those cases.

model is simply logistic regression. To further speed training, we whitened the 1536 dimensional representation.

Under this transformation, the experiment regime is identical to the permutation invariant MNIST task. We therefore used a similar model architecture. Inputs are passed through two fully connected layers, each with 1024 units. Next, data is fed to a stochastic encoding layer; this layer is characterized by a spherical Gaussian with 1024 learned means and standard deviations. The output of the stochastic layer is fed to the variational classifier–itself a logistic regression, for simplicity. All other hyperparameters and training choices are identical to those used in MNIST, more details in Appendix A.

Classification

We see the same favorable VIB classification performance in ImageNet as in MNIST. By varying β, the estimated mutual information between encoding and image ($I(Z, X)$) varies as well. At large values of β accuracy suffers, but at intermediate values we obtain improved performance over both a deterministic baseline and a $\beta = 0$ regime. In all cases our accuracy is somewhat lower than the original 80.4% accuracy. This may be a consequence of inadequate training time or suboptimal hyperparameters.

Overall the best accuracy we achieved was using $\beta = 0.01$. Under this setting we saw an accuracy of 80.12%–nearly the same as the state-of-the-art unmodified network– but with substantially smaller information footprint, only $I(X, Z) \sim 45$ bits. This is a surprisingly small amount of information; $\beta = 0$ implies over 10,000 bits yet only reaches an accuracy of 78.87%. The deterministic baseline, which was the same network, but without the VIB loss and a 1024 fully connected linear layer instead of the stochastic embedding similarly only achieved 78.75% accuracy. We stress that regressions from the achievable 80.4% are likely due to suboptimal hyperparameters settings or inadequate training.

Considering a continuum of β and a deterministic baseline, the best classification accuracy was achieved with a $\beta = 0.01 \in (0, 1)$. In other words, VIB offered accuracy benefit yet using a mere ~ 45 bits of information from each image.

Adversarial Robustness

We next show that the VIB-trained network improves resistance to adversarial attack.

We focus on the Carlini targeted L_2 attack (see Section 4.2.1). We show results for the VIB-trained network and a deterministic baseline (both on top of precomputed features), as well as for the original pretrained Inception ResNet V2 network itself. The VIB network is more robust to the targeted L_2 optimization attack in both magnitude of perturbation and frequency of successful attack.

Figure 7 shows some example images which were all misclassified as "soccer balls" by the deterministic models; by contrast, with the VIB model, only 17 out of 30 of the attacks succeeded in being mislabeled as the target label.[11] We find that the VIB model can resist about 43.3% of the attacks, but the deterministic models always fail (i.e., always misclassify into the targeted label).

Figure 8 shows the absolute pixel differences between the perturbed and unperturbed images for the examples in Figure 7. We see that the VIB network requires much larger perturbations in order to fool the classifier, as quantified in Table 2.

Metric	Determ	IRv2	VIB(0.01)
Sucessful target	1.0	1.0	**0.567**
L_2	6.45	14.43	**43.27**
L_∞	0.18	0.44	**0.92**

Table 2: Quantitative results showing how the different Inception Resnet V2-based architectures (described in Section 4.2.5) respond to targeted L_2 adversarial examples. *Determ* is the deterministic architecture, *IRv2* is the unmodified Inception Resnet V2 architecture, and *VIB(0.01)* is the VIB architecture with $\beta = 0.01$. *Successful target* is the fraction of adversarial examples that caused the architecture to classify as the target class (soccer ball). Lower is better. L_2 and L_∞ are the average L distances between the original images and the adversarial examples. Larger values mean the adversary had to make a larger perturbation to change the class.

5 Future Directions

There are many possible directions for future work, including: putting the VIB objective at multiple or every layer of a network; testing on real images; using richer parametric marginal approximations, rather than assuming $r(z) = \mathcal{N}(0, I)$; exploring the connections to differential privacy (see e.g., Wang et al. (2016a); Cuff & Yu (2016)); and investigating open universe classification problems (see e.g., Bendale & Boult (2015)). In addition, we would like to explore applications to sequence prediction, where X denotes the past of the sequence and Y the future, while Z is the current representation of the network. This form of the information bottleneck is known as predictive information (Bialek et al., 2001; Palmer et al., 2015).

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

David Barber Felix Agakov. The IM algorithm: a variational approach to information maximization. In *NIPS*, volume 16, 2004.

Shumeet Baluja, Michele Covell, and Rahul Sukthankar. The virtues of peer pressure: A simple method for discovering high-value mistakes. In *Intl. Conf. Computer Analysis of Images and Patterns*, 2015.

Abhijit Bendale and Terrance Boult. Towards open world recognition. In *CVPR*, 2015.

[11] The attacks still often cause the VIB model to misclassify the image, but not to the targeted label. This is a form of "partial" robustness, in that an attacker will have a harder time hitting the target class, but can still disrupt correct function of the network.

William Bialek, Ilya Nemenman, and Naftali Tishby. Predictability, complexity, and learning. *Neural computation*, 13(11):2409–2463, 2001.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. In *ICML*, 2015.

Ryan P. Browne and Paul D. McNicholas. Multivariate sharp quadratic bounds via Σ-strong convexity and the fenchel connection. *Electronic Journal of Statistics*, 9, 2015.

Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. *Arxiv*, 2016.

Matthew Chalk, Olivier Marre, and Gasper Tkacik. Relevant sparse codes with variational information bottleneck. In *NIPS*, 2016.

G. Chechik, A Globersonand N. Tishby, and Y. Weiss. Information bottleneck for gaussian variables. *J. of Machine Learning Research*, 6:165188, 2005.

Paul Cuff and Lanqing Yu. Differential privacy as a mutual information constraint. In *ACM Conference on Computer and Communications Security (CCS)*, 2016.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 248–255. IEEE, 2009.

Alhussein Fawzi, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. Robustness of classifiers: from adversarial to random noise. In *NIPS*, 2016.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *AI/Statistics*, volume 9, pp. 249–256, 2010.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *ICLR*, 2015.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-VAE: Learning basic visual concepts with a constrained variational framework. In *ICLR*, 2017. URL https://openreview.net/pdf?id=Sy2fzU9gl.

Ruitong Huang, Bing Xu, Dale Schuurmans, and Csaba Szepesvári. Learning with a strong adversary. *CoRR*, abs/1511.03034, 2015.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. In *ICLR*, 2014.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *ICLR Workshop*, 2017. URL https://openreview.net/pdf?id=S1OufnIlx.

Christos Louizos, Kevin Swersky, Yujia Li, Max Welling, and Richard Zemel. The variational fair autoencoder. In *ICLR*, 2016. URL http://arxiv.org/abs/1511.00830.

David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

Shakir Mohamed and Danilo Jimenez Rezende. Variational information maximisation for intrinsically motivated reinforcement learning. In *NIPS*, pp. 2125–2133, 2015.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. *Arxiv*, 2016.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *CVPR*, 2016.

Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *CVPR*, 2015. URL http://arxiv.org/abs/1412.1897.

Stephanie E Palmer, Olivier Marre, Michael J Berry, and William Bialek. Predictive information in a sensory population. *PNAS*, 112(22):6908–6913, 2015.

Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *Proceedings of the 1st IEEE European Symposium on Security and Privacy*, 2015.

Gabriel Pereyra, George Tuckery, Jan Chorowski, and Lukasz Kaiser. Regularizing neural networks by penalizing confident output predictions. In *ICLR Workshop*, 2017. URL https://openreview.net/pdf?id=HyhbYrGYe.

Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 30(4):838–855, 1992.

Leigh Robinson and Benjamin Graham. Confusing deep convolution networks by relabelling. *arXiv preprint 1510.06925*, 2015.

Sara Sabour, Yanshuai Cao, Fartash Faghri, and David J Fleet. Adversarial manipulation of deep representations. In *ICLR*, 2016.

Noam Slonim, Gurinder Singh Atwal, Gašper Tkačik, and William Bialek. Information-based clustering. *PNAS*, 102(51):18297–18302, 2005.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *ICLR*, 2014. URL http://arxiv.org/abs/1312.6199.

Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. *arXiv preprint arXiv:1602.07261*, 2016.

N Tishby and N Zaslavsky. Deep learning and the information bottleneck principle. In *IEEE Information Theory Workshop*, pp. 1–5, April 2015a.

N. Tishby, F.C. Pereira, and W. Biale. The information bottleneck method. In *The 37th annual Allerton Conf. on Communication, Control, and Computing*, pp. 368–377, 1999.

Naftali Tishby and Noga Zaslavsky. Deep learning and the information bottleneck principle. In *Information Theory Workshop (ITW), 2015 IEEE*, pp. 1–5. IEEE, 2015b.

Weina Wang, Lei Ying, and Junshan Zhang. On the relation between identifiability, differential privacy and Mutual-Information privacy. *IEEE Trans. Inf. Theory*, 62:5018–5029, 2016a.

Weiran Wang, Honglak Lee, and Karen Livescu. Deep variational canonical correlation analysis. *arXiv [cs.LG]*, 11 October 2016b. URL https://arxiv.org/abs/1610.03454.

A HYPERPARAMETERS AND ARCHITECTURE DETAILS FOR EXPERIMENTS

All of the networks for this paper were trained using TensorFlow (Abadi et al., 2016). All weights were initialized using the default TensorFlow Xavier initialization scheme (Glorot & Bengio, 2010) using the averaging fan scaling factor on uniform noise. All biases were initialized to zero. The Adam optimizer (Kingma & Ba, 2015) was used with initial learning rate of 10^{-4}, $(\beta_1 = 0.5, \beta_2 = 0.999)$ and exponential decay, decaying the learning rate by a factor of 0.97 every 2 epochs. The networks were all trained for 200 epochs total. For the MNIST experiments, a batch size of 100 was used, and the full 60,000 training and validation set was used for training, and the 10,000 test images for test results. The input images were scaled to have values between -1 and 1 before fed to the network.

All runs maintained an exponential weighted average of the parameters during the training run; these averaged parameters were used at test time. This is in the style of Polyak averaging Polyak & Juditsky (1992), with a decay constant of 0.999. Our estimate of mutual informations were measured in bits. For the VIB experiments in all sections, no other form of regularization was used.

For the 256 dimensional gaussian embeddings of Section 4.1.1, a linear layer of size 512 was used to create the 256 mean values and standard deviations for the embedding. The standard deviations were made to be positive by a softplus transformation with a bias of -5.0 to have them initially be small.

$$\sigma = \log\left(1 + \exp(x - 5.0)\right) \tag{19}$$

For the 1024 dimensional Imagenet embeddings of Section 4.2.5, a sigma bias of 0.57 was used to keep the initial standard deviations near 1 originally, and a batch size of 200 was used.

For the 2 dimensional gaussian embeddings of Section 4.1.2, a linear layer was used with 2+4 = 6 outputs, the first two of which were used for the means, and the other 4 were reshaped to a 2×2 matrix, the center was transformed according to a softplus with a bias of -5.0, and the off diagonal components were multiplied by 10^{-2}, while the upper triangular element was dropped to form the Cholesky decomposition of the covariance matrix.

B CONNECTION TO VARIATIONAL AUTOENCODERS

We can also consider unsupervised versions of the information bottleneck objective. Consider the objective:

$$\max I(Z, X) - \beta I(Z, i), \tag{20}$$

similar to the information theoretic objective for clustering introduced in Slonim et al. (2005).

Here the aim is to take our data X and maximize the mutual information contained in some encoding Z, while restricting how much information we allow our representation to contain about the identity of each data element in our sample (i). We will form a bound much like we did in the main text. For the first term, we form a variational decoder $q(x|z)$ and take a bound:

$$\begin{aligned} I(Z, X) &= \int dx\, dz\, p(x, z) \log \frac{p(x|z)}{p(x)} && (21)\\ &= H(x) + \int dz\, p(x) \int dx\, p(x|z) \log p(x|z) && (22)\\ &\geq \int dz\, p(x) \int dx\, p(x|z) \log q(x|z) && (23)\\ &= \int dx\, p(x) \int dz\, p(x|z) \log q(x|z). && (24) \end{aligned}$$

Here we have dropped the entropy in our data $H(X)$ because it is out of our control and we have used the nonnegativity of the Kullbach-Leibler divergence to replace our intractable $p(x|z)$ with a variational decoder $q(x|z)$.

Turning our attention to the second term, note that:

$$p(z|i) = \int dx\, p(z|x)p(x|i) = \int dx\, p(z|x)\delta(x - x_i) = p(z|x_i), \tag{25}$$

and that we will take $p(i) = \frac{1}{N}$.

So that we can bound our second term from above

$$I(Z, i) = \sum_i \int dz\, p(z|i)p(i) \log \frac{p(z|i)}{p(z)} \tag{26}$$

$$= \frac{1}{N} \sum_i \int dz\, p(z|x_i) \log \frac{p(z|x_i)}{p(z)} \tag{27}$$

$$\leq \frac{1}{N} \sum_i \int dz\, p(z|x_i) \log \frac{p(z|x_i)}{r(z)}, \tag{28}$$

Where we have replaced the intractable marginal $p(z)$ with a variational marginal $r(z)$.

Putting these two bounds together we have that our unsupervised information bottleneck objective takes the form

$$I(Z, X) - \beta I(Z, i) \leq \int dx\, p(x) \int dz\, p(z|x) \log q(x|z) - \beta \frac{1}{N} \sum_i \mathrm{KL}[p(Z|x_i), r(Z)]. \tag{29}$$

And this takes the form of a variational autoencoder (Kingma & Welling, 2014), except with the second KL divergence term having an arbitrary weight β.

It is interesting that while this objective takes the same mathematical form as that of a Variational Autoencoder, the interpretation of the objective is very different. In the VAE, the model starts life as a generative model with a defined prior $p(z)$ and stochastic decoder $p(x|z)$ as part of the model, and the encoder $q(z|x)$ is created to serve as a variational approximation to the true posterior $p(z|x) = p(x|z)p(z)/p(x)$. In the VIB approach, the model is originally just the stochastic encoder $p(z|x)$, and the decoder $q(x|z)$ is the variational approximation to the true $p(x|z) = p(z|x)p(x)/p(z)$ and $r(z)$ is the variational approximation to the marginal $p(z) = \int dx\, p(x)p(z|x)$. This difference in interpretation makes natural suggestions for novel directions for improvement.

This precise setup, albeit with a different motivation was recently explored in Higgins et al. (2016), where they demonstrated that by changing the weight of the variational autoencoders regularization term, there were able to achieve latent representations that were more capable when it came ot zero-shot learning and understanding "objectness". In that work, they motivated their choice to change the relative weightings of the terms in the objective by appealing to notions in neuroscience. Here we demonstrate that appealing to the information bottleneck objective gives a principled motivation and could open the door to better understanding the optimal choice of β and more tools for accessing the importance and tradeoff of both terms.

Beyond the connection to existing variational autoencoder techniques, we note that the unsupervised information bottleneck objective suggests new directions to explore, including targetting the exact marginal $p(z)$ in the regularization term, as well as the opportunity to explore tighter bounds on the first $I(Z, X)$ term that may not require explicit variational reconstruction.

C Quadratic bounds for stochastic logistic regression decoder

Consider the special case when the bottleneck Z is a multivariate Normal, i.e., $z|x \sim N(\mu_x, \Sigma_x)$ where Σ_x is a $K \times K$ positive definite matrix. The parameters μ_x, Σ_x can be constructed from a deep neural network, e.g.,

$$\begin{aligned} \mu_x &= \gamma_{1:K}(x) \\ \mathrm{chol}(\Sigma_x) &= \mathrm{diag}(\log(1 + \exp(\gamma_{K+1:2K}))) + \mathrm{subtril}(\gamma_{2K+1:K(K+3)/2}), \end{aligned}$$

where $\gamma(x) \in \mathbb{R}^{K(K+3)/2}$ is the network output of input x.

Suppose that the prediction is a categorical distribution computed as $\mathcal{S}(Wz)$ where W is a $C \times K$ weight matrix and $\log \mathcal{S}(x) = x - \mathsf{lse}(x)$ is the log-soft-max function with $\mathsf{lse}(x) = \log \sum_{k=1}^{K} \exp(x_k)$ being the log-sum-exp function.

This setup (which is identical to our experiments) induces a classifier which is bounded by a quadratic function, which is interesting because the theoretical framework Fawzi et al. (2016) proves that quadratic classifiers have greater capacity for adversarial robustness than linear functions.

We now derive an approximate bound using second order Taylor series expansion (TSE). The bound can be made proper via Browne & McNicholas (2015). However, using the TSE is sufficient to sketch the derivation.

Jensen's inequality implies that the negative log-likelihood soft-max is upper bounded by:

$$\begin{aligned} -\log \mathsf{E}\left[\mathcal{S}(WZ)|\mu_x, \Sigma_x\right] &\leq -\mathsf{E}\left[\log \mathcal{S}(WZ)|\mu_x, \Sigma_x\right] \\ &= -W\mu_x + \mathsf{E}\left[\mathsf{lse}(WZ)|\mu_x, \Sigma_x\right] \\ &= -W\mu_x + \mathsf{E}\left[\mathsf{lse}(Z)|W\mu_x, W\Sigma_x\right]. \end{aligned}$$

The second order Taylor series expansion (TSE) of lse is given by,

$$\mathsf{lse}(x+\delta) \approx \mathsf{lse}(x) + \delta^{\mathsf{T}}\,\mathcal{S}(x) + \tfrac{1}{2}\delta^{\mathsf{T}}\,\left[\operatorname{diag}(\mathcal{S}(x)) - \mathcal{S}(x)\mathcal{S}(x)^{\mathsf{T}}\right]\delta.$$

Taking the expectation of the TSE at the mean yields,

$$\begin{aligned} \mathsf{E}_{N(0,W\Sigma_x W^{\mathsf{T}})}[\,\mathsf{lse}(W\mu_x+\delta)] &\approx \mathsf{lse}(W\mu_x) + \mathsf{E}_{N(0,W\Sigma_x W^{\mathsf{T}})}[\delta^{\mathsf{T}}]\mathcal{S}(W\mu_x) + \\ &\quad + \tfrac{1}{2}\,\mathsf{E}_{N(0,W\Sigma_x W^{\mathsf{T}})}[\delta^{\mathsf{T}}\,\left[\operatorname{diag}(\mathcal{S}(W\mu_x)) - \mathcal{S}(W\mu_x)\mathcal{S}(W\mu_x)^{\mathsf{T}}\right]\delta] \\ &= \mathsf{lse}(W\mu_x) + \tfrac{1}{2}\operatorname{tr}(W\Sigma_x W^{\mathsf{T}}\,\left[\operatorname{diag}(\mathcal{S}(W\mu_x)) - \mathcal{S}(W\mu_x)\mathcal{S}(W\mu_x)^{\mathsf{T}}\right]) \\ &= \mathsf{lse}(W\mu_x) + \tfrac{1}{2}\operatorname{tr}(W\Sigma_x W^{\mathsf{T}}\,\operatorname{diag}(\mathcal{S}(W\mu_x))) - \tfrac{1}{2}\mathcal{S}(W\mu_x)^{\mathsf{T}}\,W\Sigma_x W^{\mathsf{T}}\,\mathcal{S}(W\mu_x) \\ &= \mathsf{lse}(W\mu_x) + \tfrac{1}{2}\sqrt{\mathcal{S}(W\mu_x)}^{\mathsf{T}}\,W\Sigma_x W^{\mathsf{T}}\,\sqrt{\mathcal{S}(W\mu_x)} - \tfrac{1}{2}\mathcal{S}(W\mu_x)^{\mathsf{T}}\,W\Sigma_x W^{\mathsf{T}}\,\mathcal{S}(W\mu_x) \end{aligned}$$

The second-moment was calculated by noting,

$$\mathsf{E}[X^{\mathsf{T}}\,BX] = \mathsf{E}\operatorname{tr}(XX^{\mathsf{T}}\,B) = \operatorname{tr}(\mathsf{E}[XX^{\mathsf{T}}]B) = \operatorname{tr}(\Sigma B).$$

Putting this altogether, we conclude,

$$\mathsf{E}\left[\mathcal{S}(WZ)|\mu_x, \Sigma_x\right] \gtrapprox \mathcal{S}(W\mu_x)\exp\left(-\tfrac{1}{2}\sqrt{\mathcal{S}(W\mu_x)}^{\mathsf{T}}\,W\Sigma_x W^{\mathsf{T}}\,\sqrt{\mathcal{S}(W\mu_x)} + \tfrac{1}{2}\mathcal{S}(W\mu_x)^{\mathsf{T}}\,W\Sigma_x W^{\mathsf{T}}\,\mathcal{S}(W\mu_x)\right).$$

As indicated, rather than approximate the lse via TSE, we can make a sharp, quadratic upper bound via Browne & McNicholas (2015). However this merely changes the $\mathcal{S}(W\mu_x)$ scaling in the exponential; the result is still log-quadratic.