Deep Variational Information Bottleneck

Kayla Bennett

University of Arizona

February 26, 2024

Paper Contributions

- Variational approximation to Information Bottleneck objective function
- Apply Variational IB method to neural networks
- Objective is robust to adversarial attack

Background: Mutual Information

- Measures mutual dependence between variables
- Always non-negative
- Invariant to reparameterizations
- $I(X;Y) = D_{KL}(P(X,Y)||P(X) \otimes P(Y))$
- In words: KL divergence between marginal distribution and product of distributions

Background: Information Bottleneck Method

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

- Learn informative encoding Z about target Y while ignoring input data X
- Intractable for most models
  - Exceptions: All variables are discrete, or all are jointly gaussian
- $\beta > 0$ hyperparameter trades between objectives
- High β fails to learn anything
- Experiments further explore β

Method: Overview

- Use variational approximations to compute lower-bound of IB objective
- Use local reparameterization trick & Monte Carlo sampling to estimate the gradient
- Optimize using SGD and train a deep neural network

Factoring the Joint

- Assume factorization:

$$p(X, Y, Z) = p(Z|X, Y)p(Y|X)p(X) = p(Z|X)p(Y|X)p(X)$$

- Markov chain: $Y \leftrightarrow X \leftrightarrow Z$
- Representation Z can't depend directly on Y

First Term of IB Objective

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

$p(y|z)$ from Markov chain:

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

Approximating the First Term

- $p(y|z)$ is intractable
- define variational approximation $q(y|z)$
- $q(y|z)$ is a second neural network with its own parameters

Lower Bound on $I(Z, Y)$ using $q(y|z)$

KL Divergence is always nonnegative:

$$KL[p(Y|Z), q(Y|Z)] \geq 0 \rightarrow \int dy\, p(y|z) \log p(y|z) \geq \int dy\, p(y|z) \log q(y|z)$$

$$I(Z, Y) \geq \int dy\, dz\, p(y, z) \log q(y|z) + H(Y)$$

Ignore $H(Y)$ since it doesn't depend on the optimization.

Lower Bound on $I(Z, Y)$ continued

Rewrite using Markov Chain assumption:

$$p(y,z) = \int dx\, p(x,y,z) = \int dx\, p(x)p(y|x)p(z|x)$$

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

Above lower bound only requires samples from our data and stochastic encoder, as well as tractable $q(y|z)$!

Approximating $\beta I(Z, X)$

$$I(Z,X) = \int dz\, dx\, p(x,z) \log \tfrac{p(z|x)}{p(z)}$$

$$= \int dz\, dx\, p(x,z) \log p(z|x) - \int dz\, p(z) \log p(z)$$

- Computing marginal $p(z)$ could be difficult
- Define variational approximation $r(z)$
- Use nonnegativity of KL to get an upper bound:

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

Combining Terms and Approximating Empirically

$$I(Z,Y) - \beta I(Z,X)$$

$$\geq \int dx\, dy\, dz\, p(x)p(y|x)p(z|x) \log q(y|z)$$

$$-\beta \int dx\, dz\, p(x)p(z|x) \log \frac{p(z|x)}{r(z)}$$

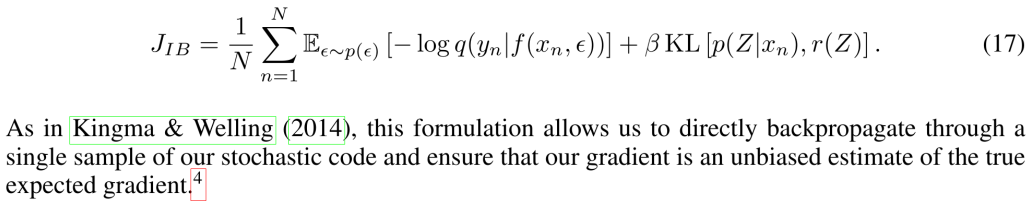
$$= L$$

- Approximate $p(x,y)$ empirically:

$$p(x,y) = \frac{1}{N}\sum_{n=1}^{N} \delta_{x_n}(x)\delta_{y_n}(y)$$

Finally the Objective Function

- $p(z|x) = N(z|f_e^{\mu}(x), f_e^{\Sigma}(x))$
- MLP encoder outputs K-dimensional μ and σ
- Apply Reparameterization trick: $p(z|x)dz = p(\epsilon)d\epsilon$ and $z = f(x, \epsilon)$
- Final objective function to minimize:



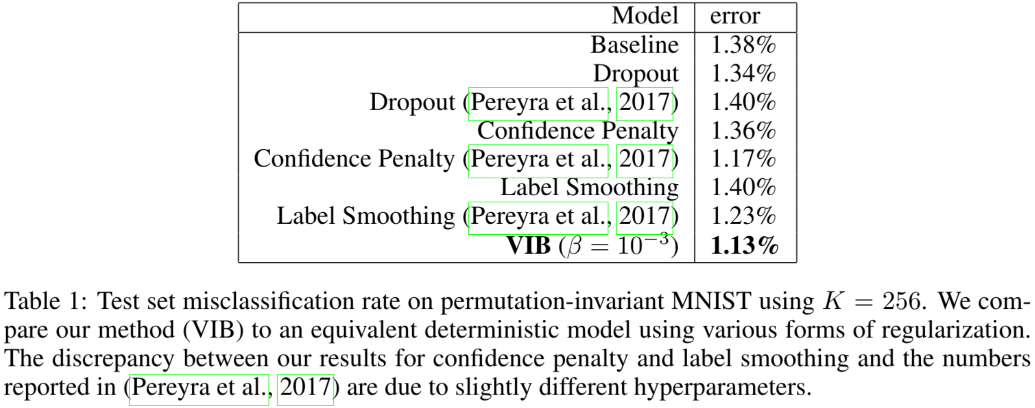
$$J_{IB} = \frac{1}{N} \sum_{n=1}^{N} \mathbb{E}_{\epsilon \sim p(\epsilon)} \left[-\log q(y_n | f(x_n, \epsilon)) \right] + \beta \, \mathrm{KL} \left[p(Z|x_n), r(Z) \right]. \tag{17}$$

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]

MNIST Experiments

- Use existing MLP model and compare with other regularization methods
- Stochastic encoder: $p(z|x) = N(z|f_e^{\mu}(x), f_e^{\Sigma}(x))$
- f_e: MLP w/ layers: $784 - 1024 - 1024 - 2K$
- Final layer: K means and K standard deviations
- Try bottleneck sizes $K = 256$ and $K = 2$
- Decoder $q(y|z)$ is a logistic regression model
- $r(z)$ is a K-dimensional spherical gaussian

MNIST: Comparison



Model	error
Baseline	1.38%
Dropout	1.34%
Dropout (Pereyra et al., 2017)	1.40%
Confidence Penalty	1.36%
Confidence Penalty (Pereyra et al., 2017)	1.17%
Label Smoothing	1.40%
Label Smoothing (Pereyra et al., 2017)	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., 2017) are due to slightly different hyperparameters.

MNIST: 2D Embedding

- worse results than K=256 embedding, but same trends
- Mostly interesting for visualizations

MNIST: Error and MI statistics

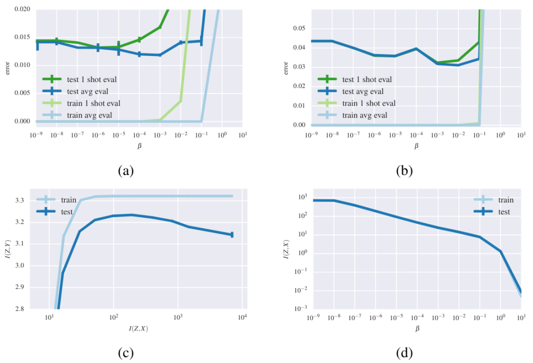
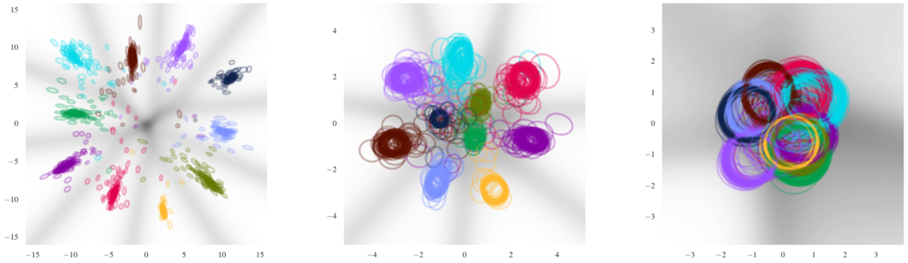


Figure 1: Results of VIB model on MNIST. (a) Error rate vs β for $K = 256$ on train and test set. "1 shot eval" means a single posterior sample of z, "avg eval" means 12 Monte Carlo samples. The spike in the error rate at $\beta \sim 10^{-2}$ corresponds to a model that is too highly regularized. Plotted 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.

MNIST: 2D Embedding Visualization



(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.

Adversary ML Models

- Train model to add noise to a sample and change class prediction
- Could be targeted or untargeted
- Authors evaluate robustness to two adversary models:
  - L_2 Optimizer
  - Fast Gradient Sign - Evaluate gradient and take one step of size ϵ

MNIST: Adversarial Experiment

- Perturb 10 zeroes to classify as ones
- use L_2 adversary from Carlini & Wagner (2016)

MNIST: Adversarial Results

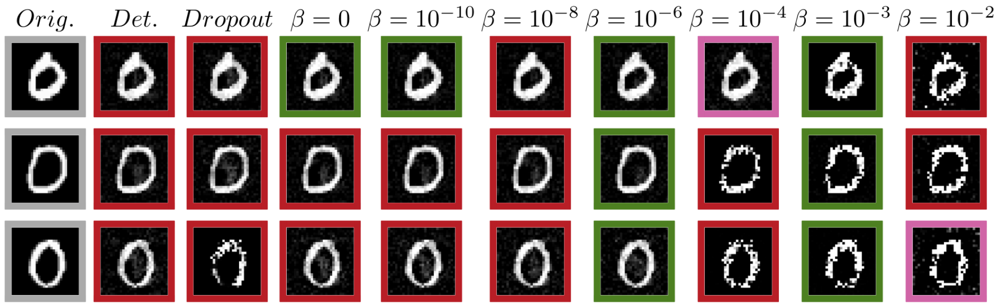
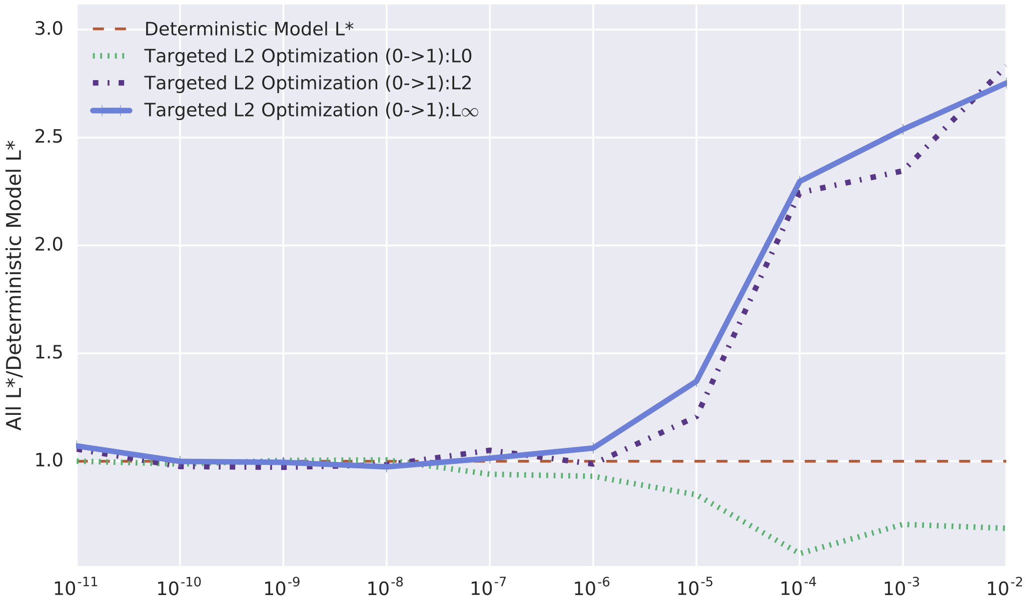
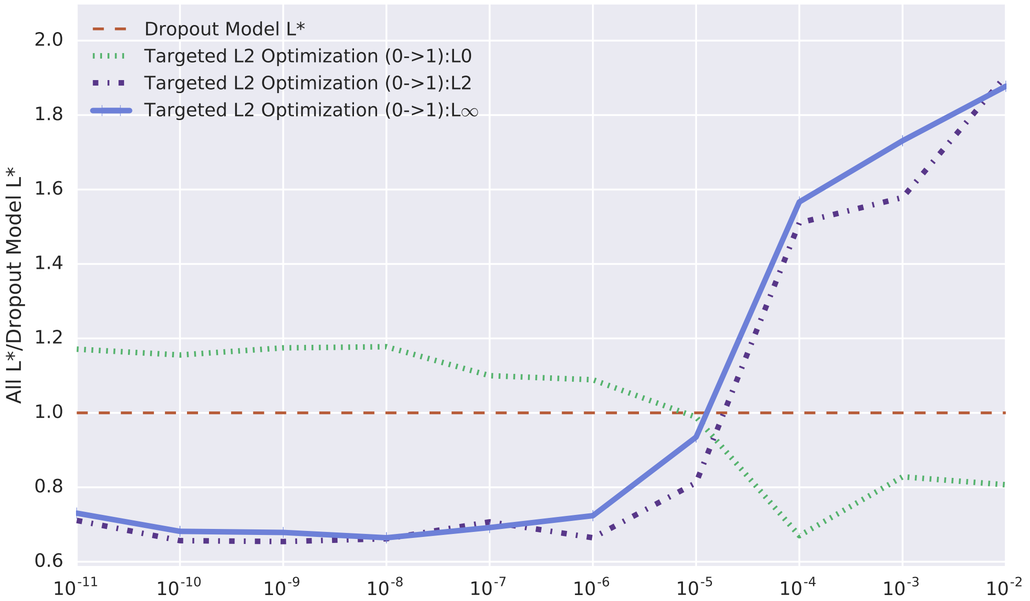


Figure 3: The adversary is trying to force each 0 to be classified as a 1. Successful attacks have a red background. Unsuccessful attacks have a green background. In the case that the label is changed to an incorrect label different from the target label (i.e., the classifier outputs something other 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.

L2 Perturbation vs Beta, Compare with Deterministic



L2 Perturbation vs Beta, Compare with Dropout



Accuracy vs Beta, L2 Adversary

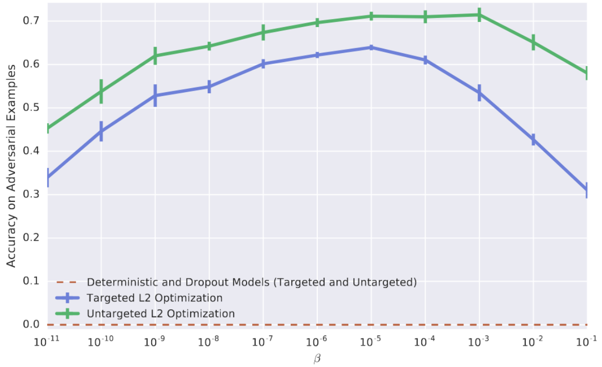
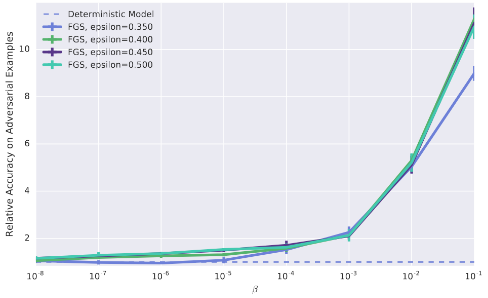
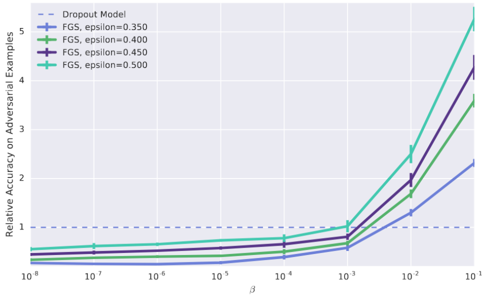


Figure 6: Classification accuracy (from 0 to 1) on L_2 adversarial examples (of all classes) as a function of β. The blue line is for targeted attacks, and the green line is for untargeted attacks (which are easier to resist). In this case, $\beta = 10^{-11}$ has performance indistinguishable from $\beta = 0$. 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.

Accuracy vs Beta, FGS Adversary



(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.

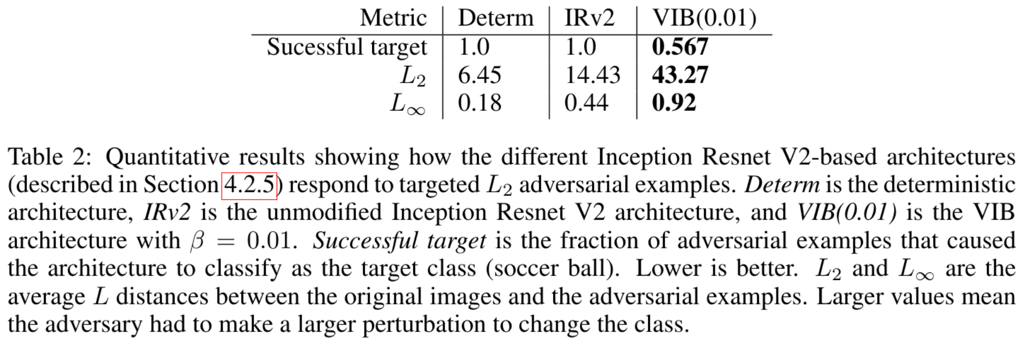
ImageNet: Adversarial Experiment

- 1M images, 1K classes
- Use pretrained Inception Resnet V2 at 80.4% accuracy
- apply the pretrained model to each image and extract latent representation
- resulting representation is 1536 dimensions
- input these to model mostly similar to MNIST experiments

ImageNet: Classification results

- All results lower than 80.4% baseline
- Likely need more training time or better hyperparameters
- Best accuracy: 80.12% with $\beta = 0.01$
- $\beta = 0.01 \rightarrow$ roughly 45 bits in $I(X, Z)$
- $\beta = 0 \rightarrow$ 10,000 bits, but only 78.87% accuracy

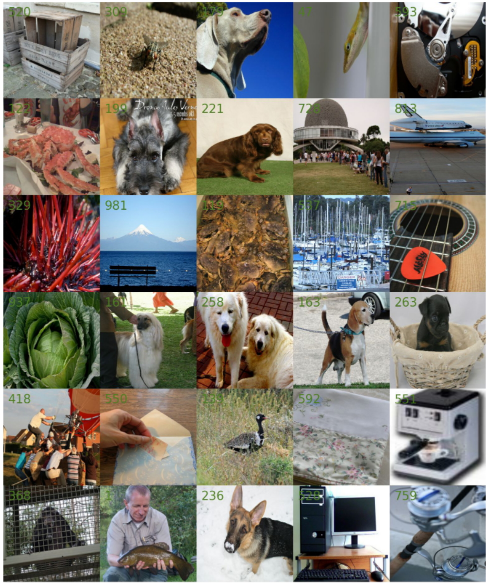
ImageNet: Adversarial Results



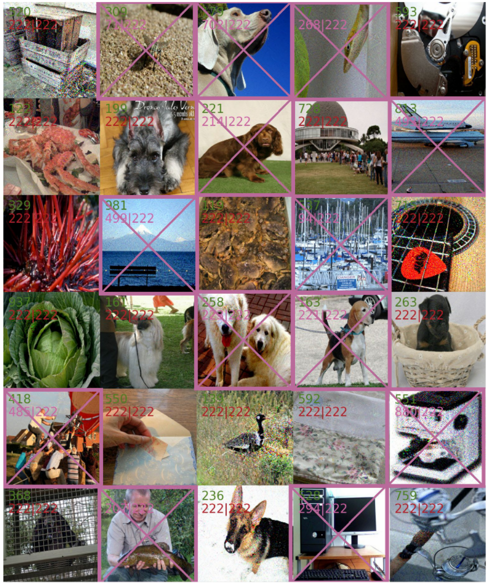
Metric	Determ	IRv2	VIB(0.01)
Sucessful target	1.0	1.0	**0.567**
L_2	6.45	14.43	**43.27**
L_∞	0.18	0.44	**0.92**

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.

ImageNet Perturbations: Clean & VIB

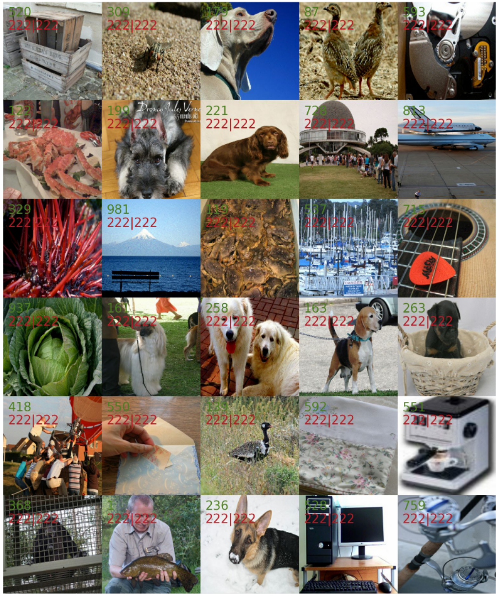


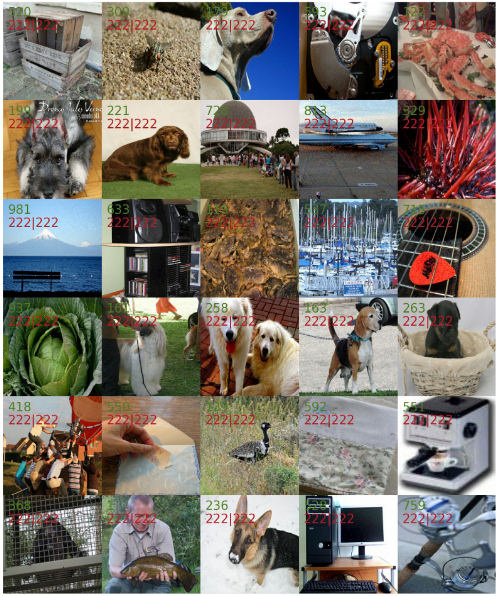
(a)



(b)

ImageNet Perturbations: Deterministic & Unmodified ResNet





ImageNet: Magnitude of Perturbations

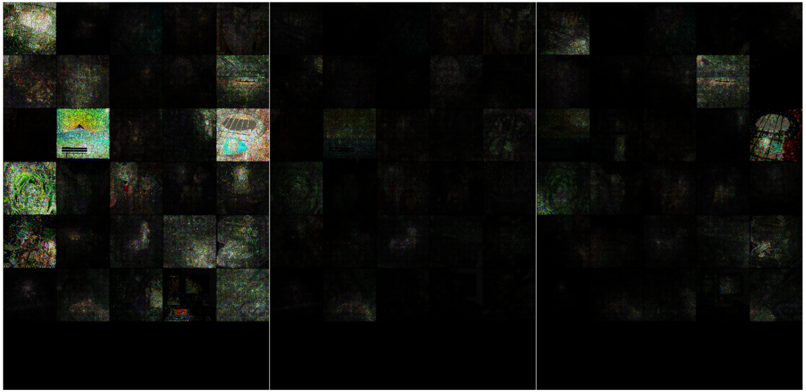


Figure 8: Shown are the absolute differences between the original and final perturbed images for all three networks. The left block shows the perturbations created while targeting the VIB network. The middle block shows the perturbations needed for the deterministic baseline using precomputed whitened 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.

Final Thoughts

- I'm reasonably convinced this is a good regularizer
- Some examples support adversarial robustness, others not so much
- Learning β is probably nontrivial

Questions?

References

Alemi, A. A., Fischer, I., Dillon, J. V., & Murphy, K. (2016). Deep variational information bottleneck. arXiv preprint arXiv:1612.00410.