Minimal Achievable Sufficient Statistic Learning

Milan Cvitkovic [1] **Günther Koliander** [2]

Abstract

We introduce Minimal Achievable Sufficient Statistic (MASS) Learning, a training method for machine learning models that attempts to produce minimal sufficient statistics with respect to a class of functions (e.g. deep networks) being optimized over. In deriving MASS Learning, we also introduce Conserved Differential Information (CDI), an information-theoretic quantity that — unlike standard mutual information — can be usefully applied to deterministically-dependent continuous random variables like the input and output of a deep network. In a series of experiments, we show that deep networks trained with MASS Learning achieve competitive performance on supervised learning and uncertainty quantification benchmarks.

1. Introduction

The *representation learning* approach to machine learning focuses on finding a representation Z of an input random variable X that is useful for predicting a random variable Y (Goodfellow et al., 2016).

What makes a representation Z "useful" is much debated, but a common assertion is that Z should be a *minimal sufficient statistic* of X for Y(Adragni, Kofi P. & Cook, R. Dennis, 2009; Shamir et al., 2010; James et al., 2017; Achille & Soatto, 2018b). That is:

1. Z should be a *statistic* of X. This means $Z = f(X)$ for some function f.
2. Z should be *sufficient* for Y. This means $p(X|Z, Y) = p(X|Z)$.
3. Given that Z is a sufficient statistic, it should be *minimal* with respect to X. This means for any measurable, non-invertible function g, $g(Z)$ is no longer sufficient for Y.[1]

In other words: a minimal sufficient statistic is a random variable Z that tells you everything about Y you could ever care about, but if you do any irreversible processing to Z, you are guaranteed to lose some information about Y.

Minimal sufficient statistics have a long history in the field of statistics (Lehmann & Scheffe, 1950; Dynkin, 1951). But the minimality condition (3, above) is perhaps too strong to be useful in machine learning, since it is a statement about *any* function g, rather than about functions in a practical hypothesis class like the class of deep neural networks.

Instead, in this work we consider *minimal achievable sufficient statistics*: sufficient statistics that are minimal among some particular set of functions.

Definition 1 (Minimal Achievable Sufficient Statistic)**.** Let $Z = f(X)$ be a sufficient statistic of X for Y. Z is *minimal achievable* with respect to a set of functions $\mathcal{F}$ if $f \in \mathcal{F}$ and for any Lipschitz continuous, non-invertible function g where $g \circ f \in \mathcal{F}$, $g(Z)$ is no longer sufficient for Y.

Contributions:

- We introduce Conserved Differential Information (CDI), an information-theoretic quantity that, unlike mutual information, is meaningful for deterministically-dependent continuous random variables, such as the input and output of a deep network.
- We introduce Minimal Achievable Sufficient Statistic Learning (MASS Learning), a training objective based on CDI for finding minimal achievable sufficient statistics.
- We provide empirical evidence that models trained by MASS Learning achieve competitive performance on supervised learning and uncertainty quantification benchmarks.

[1]Department of Computing and Mathematical Sciences, California Institute of Technology, Pasadena, California, USA [2]Acoustics Research Institute, Austrian Academy of Sciences, Vienna, Austria. Correspondence to: Milan Cvitkovic <mcvitkov@caltech.edu>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

[1]This is not the most common phrasing of statistical minimality, but we feel it is more understandable. For the equivalence of this phrasing and the standard definition see Supplementary Material 7.1.

2. Conserved Differential Information

Before we present MASS Learning, we need to introduce Conserved Differential Information (CDI), on which MASS Learning is based.

CDI is an information-theoretic quantity that addresses an oft-cited issue in machine learning (Bell & Sejnowski, 1995; Amjad & Geiger, 2018; Saxe et al., 2018; Nash et al., 2018; Goldfeld et al., 2018), which is that for a continuous random variable X and a continuous, non-constant function f, the mutual information $I(X, f(X))$ is infinite. (See Supplementary Material 7.2 for details.) This makes $I(X, f(X))$ unsuitable for use in a learning objective when f is, for example, a standard deep network.

The infinitude of $I(X, f(X))$ has been circumvented in prior works by two strategies. One is discretize X and $f(X)$ (Tishby & Zaslavsky, 2015; Shwartz-Ziv & Tishby, 2017), though this is controversial (Saxe et al., 2018). Another is to use a random variable Z with distribution $p(Z|X)$ as the representation of X rather than using $f(X)$ itself as the representation (Alemi et al., 2017; Kolchinsky et al., 2017; Achille & Soatto, 2018b). In this latter approach, $p(Z|X)$ is usually implemented by adding noise to a deep network that takes X as input.

These are both reasonable strategies for avoiding the infinitude of $I(X, f(X))$. But another approach would be to derive a new information-theoretic quantity that is better suited to this situation. To that end we present Conserved Differential Information:

Definition 2. For a continuous random variable X taking values in $\mathbb{R}^d$ and a Lipschitz continuous function $f\colon \mathbb{R}^d \to \mathbb{R}^r$, the **Conserved Differential Information** (CDI) is

$$\boxed{C(X, f(X)) := H(f(X)) - \mathbb{E}_X\left[\log\left(J_f(X)\right)\right]} \quad (1)$$

where H denotes the differential entropy

$$H(Z) = -\int p(z) \log p(z)\, \mathrm{d}z$$

and J_f is the Jacobian determinant of f

$$J_f(x) = \sqrt{\det\left(\frac{\partial f(x)}{\partial x^{\mathrm{T}}}\left(\frac{\partial f(x)}{\partial x^{\mathrm{T}}}\right)^{\mathrm{T}}\right)}$$

with $\frac{\partial f(x)}{\partial x^{\mathrm{T}}} \in \mathbb{R}^{r\times d}$ the Jacobian matrix of f at x.

Readers familiar with normalizing flows (Rezende & Mohamed, 2015) or Real NVP (Dinh et al., 2017) will note that the Jacobian determinant used in those methods is a special case of the Jacobian determinant in the definition of CDI. This is because normalizing flows and Real NVP are based on the change of variables formula for invertible mappings, while CDI is based in part on the more general change of variables formula for non-invertible mappings. More details on this connection are given in Supplementary Material 7.3. The mathematical motivation for CDI based on the recent work of Koliander et al. (2016) is provided in Supplementary Material 7.4. Figure 1 gives a visual example of what CDI measures about a function.

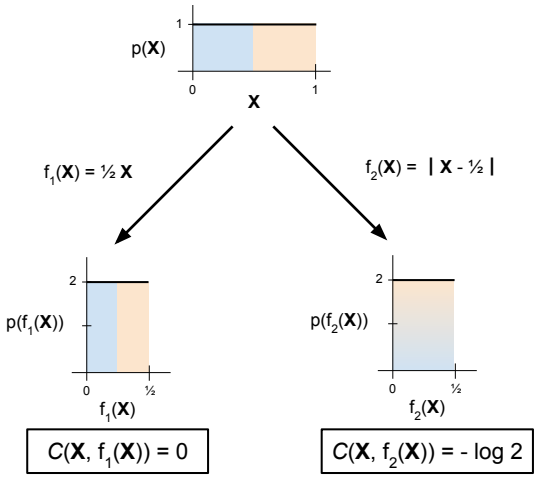

Figure 1. CDI of two functions f_1 and f_2 of the random variable X. Even though the random variables $f_1(X)$ and $f_2(X)$ have the same distribution, $C(X, f_1(X))$ is different from $C(X, f_2(X))$. This is because f_1 is an invertible function, while f_2 is not. CDI quantifies, roughly speaking, "how non-invertible" f_2 is.

The conserved differential information $C(X, f(X))$ between continuous, deterministically-dependent random variables behaves much like mutual information does between discrete random variables. For example, when f is invertible, $C(X, f(X)) = H(X)$, just like with the mutual information between discrete random variables. Most importantly for our purposes, though, $C(X, f(X))$ obeys the following data processing inequality:

Theorem 1 (CDI Data Processing Inequality)**.** *For Lipschitz continuous functions f and g with the same output space,*

$$C\left(X, f(X)\right) \geq C\left(X, g(f(X))\right)$$

with equality if and only if g is invertible almost everywhere.

The proof is in Supplementary Material 7.5.

3. MASS Learning

With CDI and its data processing inequality in hand, we can give the following optimization-based characterization of minimal achievable sufficient statistics:

Theorem 2. *Let X be a continuous random variable, Y be a discrete random variable, and $\mathcal{F}$ be any set of Lipschitz*

continuous functions with a common output space (e.g., different parameter settings of a deep network). If

$$f \in \arg\min_{S \in \mathcal{F}} C(X, S(X))$$
$$s.t.\ \ I(S(X), Y) = \max_{S'} I(S'(X), Y)$$

then $f(X)$ is a minimal achievable sufficient statistic of X for Y with respect to $\mathcal{F}$.

Proof. First note the following lemma (Cover & Thomas, 2006).

Lemma 1. *$Z = f(X)$ is a sufficient statistic for a discrete random variable Y if and only if $I(Z, Y) = \max_{S'} I(S'(X), Y)$.*

Lemma 1 guarantees that any f satisfying the conditions in Theorem 2 is sufficient. Suppose such an f was not minimal achievable. Then by Definition 1 there would exist a non-invertible, Lipschitz continuous g such that $g(f(X))$ was sufficient. But by Theorem 1, it would then also be the case that $C(X, g(f(X))) < C(X, f(X))$, which would contradict f minimizing $C(X, S(X))$. □

We can turn Theorem 2 into a learning objective over functions f by relaxing the strict constraint into a Lagrangian formulation with Lagrange multiplier $1/\beta$ for $\beta > 0$:

$$C(X, f(X)) - \frac{1}{\beta} I(f(X), Y)$$

The larger the value of β, the more our objective will encourage minimality over sufficiency. We can then simplify this formulation using the identity $I(f(X), Y) = H(Y) - H(Y|f(X))$, which gives us the following optimization objective:

$$\boxed{\begin{aligned}\mathcal{L}_{MASS}(f) := H(Y|f(X)) + \beta H(f(X)) \\ - \beta \mathbb{E}_X[\log J_f(X)].\end{aligned}} \tag{2}$$

We refer to minimizing this objective as **MASS Learning**.

3.1. Practical implementation

In practice, we are interested in using MASS Learning to train a deep network f_θ with parameters θ using a finite dataset $\{(x_i, y_i)\}_{i=1}^N$ of N datapoints sampled from the joint distribution $p(x, y)$ of X and Y. To do this, we introduce a parameterized variational approximation $q_\phi(f_\theta(x)|y) \approx p(f_\theta(x)|y)$. Using q_ϕ, we minimize the following empirical upper bound to $\mathcal{L}_{MASS}$:

$$\begin{aligned}\mathcal{L}_{MASS} \leq \widehat{\mathcal{L}}_{MASS}(\theta, \phi) := \frac{1}{N} \sum_{i=1}^{N} &- \log q_\phi(y_i|f_\theta(x_i)) \\ &- \beta \log q_\phi(f_\theta(x_i)) \\ &- \beta \log J_{f_\theta}(x_i),\end{aligned}$$

where the quantity $q_\phi(f_\theta(x_i))$ is computed as $\sum_y q_\phi(f_\theta(x_i)|y)p(y)$ and the quantity $q_\phi(y_i|f_\theta(x_i))$ is computed with Bayes rule as $\frac{q_\phi(f_\theta(x_i)|y_i)p(y_i)}{\sum_y q_\phi(f_\theta(x_i)|y)p(y)}$. When Y is discrete and takes on finitely many values, as in classification problems, and when we choose a variational distribution q_ϕ that is differentiable with respect to ϕ (e.g. a multivariate Gaussian), then we can minimize $\widehat{\mathcal{L}}_{MASS}(\theta, \phi)$ using stochastic gradient descent (SGD).

To perform classification using our trained network, we use the learned variational distribution q_ϕ and Bayes rule:

$$p(y_i|x_i) \approx p(y_i|f_\theta(x_i)) \approx \frac{q_\phi(f_\theta(x_i)|y_i)p(y_i)}{\sum_y q_\phi(f_\theta(x_i)|y)p(y)}.$$

Computing the J_{f_θ} term in $\widehat{\mathcal{L}}_{MASS}$ for every sample in an SGD minibatch is too expensive to be practical. For $f_\theta \colon \mathbb{R}^d \to \mathbb{R}^r$, doing so would require on the order of r times more operations than in standard training of deep networks by, since computing the J_{f_θ} term involves computing the full Jacobian matrix of the network, which, in our implementation, involves performing r backpropagations. Thus to make training tractable, we use a subsampling strategy: we estimate the J_{f_θ} term using only a $1/r$ fraction of the datapoints in a minibatch. In practice, we have found this subsampling strategy to not noticeably alter the numerical value of the J_{f_θ} term during training.

Subsampling for the J_{f_θ} term results in a significant training speedup, but it must nevertheless be emphasized that, even with subsampling, our implementation of MASS Learning is roughly eight times as slow as standard deep network training. (Unless $\beta = 0$, in which case the speed is the same.) This is by far the most significant drawback of (our implementation of) MASS Learning. There are many easier-to-compute upper bounds or estimates of J_{f_θ} that one could use to make MASS Learning faster, and one could also potentially find non-invertible network architectures which admit more efficiently computable Jacobians, but we do not explore these options in this work.

4. Related Work

4.1. Connection to the Information Bottleneck

The well-studied Information Bottleneck learning method (Tishby et al., 2000; Tishby & Zaslavsky, 2015; Strouse & Schwab, 2015; Alemi et al., 2017; Saxe et al., 2018; Amjad & Geiger, 2018; Goldfeld et al., 2018; Kolchinsky et al., 2019; Achille & Soatto, 2018b;a) is based on minimizing the Information Bottleneck Lagrangian

$$\mathcal{L}_{IB}(Z) := \beta I(X, Z) - I(Y, Z)$$

for $\beta > 0$, where Z is the representation whose conditional distribution $p(Z|X)$ one is trying to learn.

The $\mathcal{L}_{IB}$ learning objective can be motivated based on pure information-theoretic elegance. But some works like (Shamir et al., 2010) also point out the connection between the $\mathcal{L}_{IB}$ objective and minimal sufficient statistics, which is based on the following theorem:

Theorem 3. *Let X be a discrete random variable drawn according to a distribution $p(X|Y)$ determined by the discrete random variable Y. Let $\mathcal{F}$ be the set of deterministic functions of X to any target space. Then $f(X)$ is a minimal sufficient statistic of X for Y if and only if*

$$f \in \arg\min_{S \in \mathcal{F}} I(X, S(X))$$
$$s.t.\ \ I(S(X), Y) = \max_{S' \in \mathcal{F}} I(S'(X), Y).$$

The $\mathcal{L}_{IB}$ objective can then be thought of as a Lagrangian relaxation of the optimization problem in this theorem.

Theorem 3 only holds for discrete random variables. For continuous X it holds only in the reverse direction, so minimizing $\mathcal{L}_{IB}$ for continuous X has no formal connection to finding minimal sufficient statistics, not to mention minimal achievable sufficient statistics. See Supplementary Material 7.6 for details.

Nevertheless, the optimization problems in Theorem 2 and Theorem 3 are extremely similar, relying as they both do on Lemma 1 for their proofs. And the idea of relaxing the optimization problem in Theorem 2 into a Lagrangian formulation to get $\mathcal{L}_{MASS}$ is directly inspired by the Information Bottleneck. So while MASS Learning and Information Bottleneck learning entail different network architectures and loss functions, there is an Information Bottleneck flavor to MASS Learning.

4.2. Jacobian Regularization

The presence of the J_{f_θ} term in $\widehat{\mathcal{L}}_{MASS}$ is reminiscent of the contrastive autoencoder (Rifai et al., 2011) and Jacobian Regularization literature (Sokolic et al., 2017; Ross & Doshi-Velez, 2018; Varga et al., 2017; Novak et al., 2018; Jakubovitz & Giryes, 2018). Both these literatures suggest that minimizing $\mathbb{E}_X[\|D_f(X)\|_F]$, where $D_f(x) = \frac{\partial f(x)}{\partial x^{\mathrm{T}}} \in \mathbb{R}^{r \times d}$ is the Jacobian matrix, seems to improve generalization and adversarial robustness.

This may seem paradoxical at first, since by applying the AM-GM inequality to the eigenvalues of $D_f(x)D_f(x)^{\mathrm{T}}$ we have

$$\begin{aligned}
\mathbb{E}_X[\|D_f(X)\|_F^{2r}] &= \mathbb{E}_X[\mathrm{Tr}(D_f(X)D_f(X)^{\mathrm{T}})^r] \\
&\geq \mathbb{E}_X[r^r \det(D_f(X)D_f(X)^{\mathrm{T}})] \\
&= \mathbb{E}_X[r^r J_f(X)^2] \\
&\geq \log \mathbb{E}_X[r^r J_f(X)^2] \\
&\geq 2\mathbb{E}_X[\log J_f(X)] + r \log r
\end{aligned}$$

and $\mathbb{E}_X[\log J_f(X)]$ is being *maximized* by $\widehat{\mathcal{L}}_{MASS}$. So $\widehat{\mathcal{L}}_{MASS}$ might seem to be optimizing for worse generalization according to the Jacobian regularization literature. However, the entropy term in $\widehat{\mathcal{L}}_{MASS}$ strongly encourages minimizing $\mathbb{E}_X[\|D_f(X)\|_F]$. So overall $\widehat{\mathcal{L}}_{MASS}$ seems to be seeking the right balance of sensitivity (dependent on the value of β) in the network to its inputs, which is precisely in alignment with what the Jacobian regularization literature suggests.

5. Experiments

In this section we compare MASS Learning to other approaches for training deep networks. Code to reproduce all experiments is available online.[2] Full details on all experiments is in Supplementary Material 7.7.

We use the abbreviation "SoftmaxCE" to refer to the standard approach of training deep networks for classification problems by minimizing the softmax cross entropy loss

$$\widehat{\mathcal{L}}_{SoftmaxCE}(\theta) := -\frac{1}{N}\sum_{i=1}^{N}\left(\log \texttt{softmax}(f_\theta(x_i))_{y_i}\right)$$

where $\texttt{softmax}(f_\theta(x_i))_{y_i}$ is the y_ith element of the softmax function applied to the outputs $f_\theta(x_i)$ of the network's last linear layer. As usual, $\texttt{softmax}(f_\theta(x_i))_{y_i}$ is taken to be the network's estimate of $p(y_i|x_i)$.

We also compare against the Variational Information Bottleneck method (Alemi et al., 2017) for representation learning, which we abbreviate as "VIB".

We use two networks in our experiments. "SmallMLP" is a feedforward network with two fully-connected layers of 400 and 200 hidden units, respectively, both with `elu` nonlinearities (Clevert et al., 2015). "ResNet20" is the 20-layer residual network of He et al. (2016).

We performed all experiments on the CIFAR-10 dataset (Krizhevsky, 2009) and implemented all experiments using PyTorch (Paszke et al., 2017).

5.1. Classification Accuracy and Regularization

We first confirm that networks trained by MASS Learning can make accurate predictions in supervised learning tasks. We compare the classification accuracy of networks trained on varying amounts of data to see the extent to which MASS Learning regularizes networks.

Classification accuracies for the SmallMLP network are shown in Table 1, and for the ResNet20 network in Table 2. For the SmallMLP network, MASS Learning performs slightly worse than SoftmaxCE and VIB training. For the

[2] https://github.com/mwcvitkovic/MASS-Learning

Table 1. Test-set classification accuracy (percent) on CIFAR-10 dataset using the SmallMLP network trained by various methods. Full experiment details are in Supplementary Material 7.7. Values are the mean classification accuracy over 4 training runs with different random seeds, plus or minus the standard deviation. Emboldened accuracies are those for which the maximum observed mean accuracy in the column was within one standard deviation. WD is weight decay; D is dropout.

METHOD	TRAINING SET SIZE		
	2500	10,000	40,000
SoftmaxCE	34.2 ± 0.8	$\mathbf{44.6 \pm 0.6}$	52.7 ± 0.4
SoftmaxCE, WD	23.9 ± 0.9	36.4 ± 0.9	48.1 ± 0.1
SoftmaxCE, D	33.7 ± 1.1	44.1 ± 0.6	53.7 ± 0.3
VIB, β=1e−1	32.2 ± 0.6	40.6 ± 0.4	46.1 ± 0.5
VIB, β=1e−2	34.6 ± 0.4	43.8 ± 0.8	51.9 ± 0.8
VIB, β=1e−3	$\mathbf{35.6 \pm 0.5}$	$\mathbf{44.6 \pm 0.6}$	51.8 ± 0.8
VIB, β=1e−1, D	29.0 ± 0.6	40.1 ± 0.5	49.5 ± 0.5
VIB, β=1e−2, D	32.5 ± 0.9	43.9 ± 0.3	53.6 ± 0.3
VIB, β=1e−3, D	34.5 ± 1.0	$\mathbf{44.4 \pm 0.4}$	$\mathbf{54.3 \pm 0.2}$
MASS, β=1e−2	29.6 ± 0.4	39.9 ± 1.2	46.3 ± 1.2
MASS, β=1e−3	32.7 ± 0.8	41.5 ± 0.7	47.8 ± 0.8
MASS, β=1e−4	34.0 ± 0.3	41.5 ± 1.1	47.9 ± 0.8
MASS, β=0	34.1 ± 0.6	42.0 ± 0.6	48.2 ± 0.9
MASS, β=1e−2, D	29.3 ± 1.2	41.7 ± 0.4	52.0 ± 0.6
MASS, β=1e−3, D	31.5 ± 0.6	43.7 ± 0.2	53.1 ± 0.4
MASS, β=1e−4, D	32.7 ± 0.8	43.4 ± 0.5	53.2 ± 0.1
MASS, β=0, D	32.2 ± 1.1	43.9 ± 0.4	52.7 ± 0.0

larger ResNet20 network, MASS Learning performs equivalently to the other methods. It is notable that with the ResNet20 network VIB and MASS Learning both perform well when $\beta = 0$, and neither perform significantly better than SoftmaxCE. This may be because the hyperparameters used in training the ResNet20 network, which were taken directly from the original paper (He et al., 2016), are specifically tuned for SoftmaxCE training and are more sensitive to the specifics of the network architecture than to the loss function.

5.2. Uncertainty Quantification

We also evaluate the ability of networks trained by MASS Learning to properly quantify their uncertainty about their predictions. We assess uncertainty quantification in two ways: using proper scoring rules (Lakshminarayanan et al., 2017), which are scalar measures of how well a network's predictive distribution $p(y|f_\theta(x))$ is calibrated, and by assessing performance on an out-of-distribution (OOD) detection task.

Tables 3 through 8 show the uncertainty quantification performance of networks according to two proper scoring rules: the Negative Log Likelihood (NLL) and the Brier Score. The entropy and test accuracy of the predictive distributions are also given, for reference.

Table 2. Test-set classification accuracy (percent) on CIFAR-10 dataset using the ResNet20 network trained by various methods. No data augmentation was used — full details in Supplementary Material 7.7. Values are the mean classification accuracy over 4 training runs with different random seeds, plus or minus the standard deviation. Emboldened accuracies are those for which the maximum observed mean accuracy in the column was within one standard deviation.

METHOD	TRAINING SET SIZE		
	2500	10,000	40,000
SoftmaxCE	$\mathbf{50.0 \pm 0.7}$	$\mathbf{67.5 \pm 0.8}$	$\mathbf{81.7 \pm 0.3}$
VIB, β=1e−3	$\mathbf{49.5 \pm 1.1}$	$\mathbf{66.9 \pm 1.0}$	81.0 ± 0.3
VIB, β=1e−4	49.4 ± 1.0	66.4 ± 0.5	81.2 ± 0.4
VIB, β=1e−5	$\mathbf{50.0 \pm 1.1}$	$\mathbf{67.9 \pm 0.8}$	80.9 ± 0.5
VIB, β=0	$\mathbf{50.6 \pm 0.8}$	$\mathbf{67.1 \pm 1.0}$	$\mathbf{81.5 \pm 0.2}$
MASS, β=1e−3	38.2 ± 0.7	59.6 ± 0.8	75.8 ± 0.5
MASS, β=1e−4	$\mathbf{49.9 \pm 1.0}$	66.6 ± 0.4	80.6 ± 0.5
MASS, β=1e−5	$\mathbf{50.1 \pm 0.5}$	$\mathbf{67.4 \pm 1.0}$	$\mathbf{81.6 \pm 0.4}$
MASS, β=0	$\mathbf{50.2 \pm 1.0}$	$\mathbf{67.4 \pm 0.3}$	$\mathbf{81.5 \pm 0.2}$

For the SmallMLP network in Tables 3, 4, and 5, VIB provides the best combination of high accuracy and low NLL and Brier score across all sizes of training set, despite SoftmaxCE with weight decay achieving the best scoring rule values. For the larger ResNet20 network in Tables 6 and 7, MASS Learning provides the best combination of accuracy and proper scoring rule performance, though its performance falters when trained on only 2,500 datapoints in Table and 8. These ResNet20 UQ results also show the trend that MASS Learning with larger β leads to better calibrated network predictions. Thus, as measured by proper scoring rules, MASS Learning can significantly improve the calibration of a network's predictions while maintaining the same accuracy.

Tables 9 through 14 show metrics for performance on an OOD detection task where the network predicts not just the class of the input image, but whether the image is from its training distribution (CIFAR-10 images) or from another distribution (SVHN images (Netzer et al., 2011)). Following Hendrycks & Gimpel (2017) and Alemi et al. (2018), the metrics we report for this task are the Area under the ROC curve (AUROC) and Average Precision score (APR). APR depends on whether the network is tasked with identifying in-distribution or out-of-distribution images; we report values for both cases as APR In and APR Out, respectively.

There are different detection methods that networks can use to identify OOD inputs. One way, applicable to all training methods, is to use the entropy of the predictive distribution $p(y|f_\theta(x))$: larger entropy suggests the input is OOD. For networks trained by MASS Learning, the variational distribution $q_\phi(f_\theta(x)|y)$ is a natural OOD detector: a small value of $\max_i q_\phi(f_\theta(x)|y_i)$ suggests the input is OOD. For networks trained by SoftmaxCE, a distribution $q_\phi(f_\theta(x)|y)$

can be learned by MLE on the training set and used to detect OOD inputs in the same way.

For both the SmallMLP network in Tables 9, 10, and 11 and the ResNet20 network in Tables 12, 13, and 14, MASS Learning performs comparably or better than SoftmaxCE and VIB. However, one should note that MASS Learning with $\beta = 0$ gives performance not significantly different to MASS Learning with $\beta \neq 0$ on these OOD tasks, which suggests that the good performance of MASS Learning may be due to its use of a variational distribution to produce predictions, rather than to the overall MASS Learning training scheme.

5.3. Does MASS Learning finally solve the mystery of why stochastic gradient descent with the cross entropy loss works so well in deep learning?

We do not believe so. Figure 2 shows how the values of the three terms in $\widehat{\mathcal{L}}_{MASS}$ change as the SmallMLP network trains on the CIFAR-10 dataset using either the SoftmaxCE training or MASS Learning. Despite achieving similar accuracies, the SoftmaxCE training method does not seem to be implicitly performing MASS Learning, based on the differing values of the entropy (orange) and Jacobian (green) terms between the two methods as training progresses.

6. Discussion

MASS Learning is a new approach to representation learning that performs well on classification accuracy, regularization, and uncertainty quantification benchmarks, despite not being directly formulated for any of these tasks. It shows particularly strong performance in improving uncertainty quantification.

There are several potential ways to improve MASS Learning. Starting at the lowest level: it is likely that we did not manage to minimize $\widehat{\mathcal{L}}_{MASS}$ anywhere close to the extent possible in our experiments, given the minimal hyperparameter tuning we performed. In particular, we noticed that the initialization of the variational distribution played a large role in performance, but we were not able to fully explore it.

Moving a level higher, it may be that we are effectively minimized $\widehat{\mathcal{L}}_{MASS}$, but that $\widehat{\mathcal{L}}_{MASS}$ is not a useful empirical approximation or upper bound to $\mathcal{L}_{MASS}$. This could be due to an insufficiently expressive variational distribution, or simply that the quantities in $\widehat{\mathcal{L}}_{MASS}$ require more data to approximate well than our datasets contained.

At higher levels still, it may be the case that the Lagrangian formulation of Theorem 2 as $\mathcal{L}_{MASS}$ is impractical for finding minimal achievable sufficient statistics. Or it may be that the difference between minimal and minimal achievable sufficient statistics is relevant for performance on machine learning tasks. Or it may simply be that framing machine learning as a problem of finding minimal sufficient statistics is not productive.

Finally, while we again note that more work is needed to reduce the computational cost of our implementation of MASS Learning, we believe the concept of MASS learning, and the concepts of minimal achievability and Conserved Differential Information we introduce along with it, are beneficial to the theoretical understanding of representation learning.

Acknowledgements

We would like to thank Georg Pichler, Thomas Vidick, Alex Alemi, Alessandro Achille, and Joseph Marino for useful discussions.

References

Achille, A. and Soatto, S. Emergence of invariance and disentanglement in deep representations. In *2018 Information Theory and Applications Workshop (ITA)*, pp. 1–9, 2018a.

Achille, A. and Soatto, S. Information dropout: Learning optimal representations through noisy computation. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 2897–2905, 2018b.

Adragni, Kofi P. and Cook, R. Dennis. Sufficient dimension reduction and prediction in regression. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 367(1906):4385–4405, November 2009. doi: 10.1098/rsta.2009.0110. URL https://royalsocietypublishing.org/doi/full/10.1098/rsta.2009.0110.

Alemi, A. A., Fischer, I., Dillon, J. V., and Murphy, K. Deep variational information bottleneck. *International Conference on Learning Representations*, abs/1612.00410, 2017. URL https://arxiv.org/abs/1612.00410.

Alemi, A. A., Fischer, I., and Dillon, J. V. Uncertainty in the Variational Information Bottleneck. *arXiv:1807.00906 [cs, stat]*, July 2018. URL http://arxiv.org/abs/1807.00906. arXiv: 1807.00906.

Amjad, R. A. and Geiger, B. C. Learning representations for neural network-based classification using the information bottleneck principle. *IEEE transactions on pattern analysis and machine intelligence*, 2018.

Bell, A. J. and Sejnowski, T. J. An information-maximization approach to blind separation and blind

Table 3. Uncertainty quantification metrics (proper scoring rules) on CIFAR-10 using the SmallMLP network trained on 40,000 datapoints. Test Accuracy and Entropy of the network's predictive distribution are given for reference. Full experiment details are in Supplementary Material 7.7. Values are the mean over 4 training runs with different random seeds, plus or minus the standard deviation. Emboldened values are those for which the minimum observed mean value in the column was within one standard deviation. WD is weight decay; D is dropout. Lower values are better.

Method	Test Accuracy	Entropy	NLL	Brier Score
SoftmaxCE	52.7 ± 0.4	0.211 ± 0.003	4.56 ± 0.07	0.0840 ± 0.0005
SoftmaxCE, WD	48.1 ± 0.1	1.500 ± 0.009	$\mathbf{1.47 \pm 0.01}$	$\mathbf{0.0660 \pm 0.0003}$
SoftmaxCE, D	53.7 ± 0.3	0.606 ± 0.005	1.79 ± 0.02	0.0681 ± 0.0005
VIB, β=1e$-$1	46.1 ± 0.5	0.258 ± 0.005	5.35 ± 0.15	0.0944 ± 0.0009
VIB, β=1e$-$2	51.9 ± 0.8	0.193 ± 0.004	5.03 ± 0.19	0.0861 ± 0.0015
VIB, β=1e$-$3	51.8 ± 0.8	0.174 ± 0.003	5.49 ± 0.20	0.0866 ± 0.0015
VIB, β=1e$-$1, D	49.5 ± 0.5	0.957 ± 0.005	1.62 ± 0.01	$\mathbf{0.0660 \pm 0.0003}$
VIB, β=1e$-$2, D	53.6 ± 0.3	0.672 ± 0.014	1.69 ± 0.01	0.0668 ± 0.0006
VIB, β=1e$-$3, D	54.3 ± 0.2	0.617 ± 0.007	1.75 ± 0.02	0.0677 ± 0.0005
MASS, β=1e$-$2	46.3 ± 1.2	0.203 ± 0.005	6.89 ± 0.16	0.0968 ± 0.0024
MASS, β=1e$-$3	47.8 ± 0.8	0.207 ± 0.004	5.89 ± 0.21	0.0935 ± 0.0017
MASS, β=1e$-$4	47.9 ± 0.8	0.212 ± 0.003	5.71 ± 0.16	0.0934 ± 0.0017
MASS, β=0	48.2 ± 0.9	0.208 ± 0.004	5.74 ± 0.20	0.0927 ± 0.0017
MASS, β=1e$-$2, D	52.0 ± 0.6	0.690 ± 0.013	1.85 ± 0.03	0.0694 ± 0.0005
MASS, β=1e$-$3, D	53.1 ± 0.4	0.649 ± 0.010	1.82 ± 0.04	0.0684 ± 0.0007
MASS, β=1e$-$4, D	53.2 ± 0.1	0.664 ± 0.020	1.79 ± 0.02	0.0680 ± 0.0002
MASS, β=0, D	52.7 ± 0.0	0.662 ± 0.003	1.82 ± 0.02	0.0690 ± 0.0003

Table 4. Uncertainty quantification metrics (proper scoring rules) on CIFAR-10 using the SmallMLP network trained on 10,000 datapoints. Test Accuracy and Entropy of the network's predictive distribution are given for reference. Full experiment details are in Supplementary Material 7.7. Values are the mean over 4 training runs with different random seeds, plus or minus the standard deviation. Emboldened values are those for which the minimum observed mean value in the column was within one standard deviation. WD is weight decay; D is dropout. Lower values are better.

Method	Test Accuracy	Entropy	NLL	Brier Score
SoftmaxCE	44.6 ± 0.6	0.250 ± 0.004	5.33 ± 0.06	0.0974 ± 0.0011
SoftmaxCE, WD	36.4 ± 0.9	0.897 ± 0.033	$\mathbf{2.44 \pm 0.11}$	$\mathbf{0.0905 \pm 0.0019}$
SoftmaxCE, D	44.1 ± 0.6	0.379 ± 0.007	3.76 ± 0.04	0.0935 ± 0.0012
VIB, β=1e$-$1	40.6 ± 0.4	0.339 ± 0.011	4.86 ± 0.23	0.1017 ± 0.0016
VIB, β=1e$-$2	43.8 ± 0.8	0.274 ± 0.004	4.83 ± 0.16	0.0983 ± 0.0017
VIB, β=1e$-$3	44.6 ± 0.6	0.241 ± 0.004	5.50 ± 0.11	0.0983 ± 0.0005
VIB, β=1e$-$1, D	40.1 ± 0.5	0.541 ± 0.015	3.22 ± 0.09	0.0945 ± 0.0012
VIB, β=1e$-$2, D	43.9 ± 0.3	0.413 ± 0.009	3.43 ± 0.09	0.0927 ± 0.0011
VIB, β=1e$-$3, D	44.4 ± 0.4	0.389 ± 0.004	3.61 ± 0.06	0.0927 ± 0.0004
MASS, β=1e$-$2	39.9 ± 1.2	0.172 ± 0.008	10.06 ± 0.37	0.1109 ± 0.0020
MASS, β=1e$-$3	41.5 ± 0.7	0.197 ± 0.005	8.03 ± 0.28	0.1069 ± 0.0016
MASS, β=1e$-$4	41.5 ± 1.1	0.208 ± 0.008	7.55 ± 0.44	0.1054 ± 0.0023
MASS, β=0	42.0 ± 0.6	0.215 ± 0.009	7.21 ± 0.28	0.1043 ± 0.0015
MASS, β=1e$-$2, D	41.7 ± 0.4	0.399 ± 0.017	4.21 ± 0.17	0.0974 ± 0.0013
MASS, β=1e$-$3, D	43.7 ± 0.2	0.412 ± 0.010	3.71 ± 0.07	0.0930 ± 0.0006
MASS, β=1e$-$4, D	43.4 ± 0.5	0.435 ± 0.011	3.50 ± 0.05	0.0923 ± 0.0005
MASS, β=0, D	43.9 ± 0.4	0.447 ± 0.009	3.40 ± 0.03	$\mathbf{0.0913 \pm 0.0008}$

Table 5. Uncertainty quantification metrics (proper scoring rules) on CIFAR-10 using the SmallMLP network trained on 2,500 datapoints. Test Accuracy and Entropy of the network's predictive distribution are given for reference. Full experiment details are in Supplementary Material 7.7. Values are the mean over 4 training runs with different random seeds, plus or minus the standard deviation. Emboldened values are those for which the minimum observed mean value in the column was within one standard deviation. WD is weight decay; D is dropout. Lower values are better.

Method	Test Accuracy	Entropy	NLL	Brier Score
SoftmaxCE	34.2 ± 0.8	0.236 ± 0.025	8.14 ± 0.84	0.1199 ± 0.0024
SoftmaxCE, WD	23.9 ± 0.9	0.954 ± 0.017	$\mathbf{3.41 \pm 0.07}$	$\mathbf{0.1114 \pm 0.0013}$
SoftmaxCE, D	33.7 ± 1.1	0.203 ± 0.006	9.68 ± 0.06	0.1219 ± 0.0013
VIB, β=1e$-$1	32.2 ± 0.6	0.247 ± 0.007	8.33 ± 0.50	0.1219 ± 0.0013
VIB, β=1e$-$2	34.6 ± 0.4	0.249 ± 0.004	7.36 ± 0.18	0.1175 ± 0.0005
VIB, β=1e$-$3	35.6 ± 0.5	0.217 ± 0.008	8.03 ± 0.37	0.1175 ± 0.0012
VIB, β=1e$-$1, D	29.0 ± 0.6	0.383 ± 0.011	6.32 ± 0.16	0.1219 ± 0.0010
VIB, β=1e$-$2, D	32.5 ± 0.9	0.260 ± 0.006	7.41 ± 0.25	0.1211 ± 0.0019
VIB, β=1e$-$3, D	34.5 ± 1.0	0.200 ± 0.002	9.44 ± 0.16	0.1203 ± 0.0020
MASS, β=1e$-$2	29.6 ± 0.4	0.047 ± 0.002	57.13 ± 1.60	0.1381 ± 0.0007
MASS, β=1e$-$3	32.7 ± 0.8	0.048 ± 0.004	46.40 ± 3.81	0.1322 ± 0.0018
MASS, β=1e$-$4	34.0 ± 0.3	0.052 ± 0.002	39.10 ± 1.96	0.1293 ± 0.0009
MASS, β=0	34.1 ± 0.6	0.061 ± 0.003	33.60 ± 1.34	0.1285 ± 0.0012
MASS, β=1e$-$2, D	29.3 ± 1.2	0.118 ± 0.008	20.51 ± 0.83	0.1349 ± 0.0018
MASS, β=1e$-$3, D	31.5 ± 0.6	0.145 ± 0.004	15.65 ± 0.71	0.1289 ± 0.0010
MASS, β=1e$-$4, D	32.7 ± 0.8	0.185 ± 0.010	11.21 ± 0.66	0.1245 ± 0.0011
MASS, β=0, D	32.2 ± 1.1	0.217 ± 0.008	9.70 ± 0.29	0.1236 ± 0.0021

Table 6. Uncertainty quantification metrics (proper scoring rules) on CIFAR-10 using the ResNet20 network trained on 40,000 datapoints. Test Accuracy and Entropy of the network's predictive distribution are given for reference. Full experiment details are in Supplementary Material 7.7. Values are the mean over 4 training runs with different random seeds, plus or minus the standard deviation. Emboldened values are those for which the minimum observed mean value in the column was within one standard deviation. Lower values are better.

Method	Test Accuracy	Entropy	NLL	Brier Score
SoftmaxCE	81.7 ± 0.3	0.087 ± 0.002	1.45 ± 0.04	$\mathbf{0.0324 \pm 0.0005}$
VIB, β=1e$-$3	81.0 ± 0.3	0.089 ± 0.003	1.51 ± 0.04	0.0334 ± 0.0005
VIB, β=1e$-$4	81.2 ± 0.4	0.092 ± 0.002	1.46 ± 0.05	0.0331 ± 0.0007
VIB, β=1e$-$5	80.9 ± 0.5	0.087 ± 0.005	1.58 ± 0.08	0.0339 ± 0.0008
VIB, β=0	81.5 ± 0.2	0.079 ± 0.001	1.70 ± 0.06	0.0331 ± 0.0007
MASS, β=1e$-$3	75.8 ± 0.5	0.139 ± 0.003	1.66 ± 0.07	0.0417 ± 0.0011
MASS, β=1e$-$4	80.6 ± 0.5	0.109 ± 0.002	$\mathbf{1.33 \pm 0.02}$	0.0337 ± 0.0008
MASS, β=1e$-$5	81.6 ± 0.4	0.095 ± 0.003	$\mathbf{1.36 \pm 0.03}$	$\mathbf{0.0320 \pm 0.0005}$
MASS, β=0	81.5 ± 0.2	0.092 ± 0.000	1.43 ± 0.04	0.0325 ± 0.0004

Table 7. Uncertainty quantification metrics (proper scoring rules) on CIFAR-10 using the ResNet20 network trained on 10,000 datapoints. Test Accuracy and Entropy of the network's predictive distribution are given for reference. Full experiment details are in Supplementary Material 7.7. Values are the mean over 4 training runs with different random seeds, plus or minus the standard deviation. Emboldened values are those for which the minimum observed mean value in the column was within one standard deviation. Lower values are better.

Method	Test Accuracy	Entropy	NLL	Brier Score
SoftmaxCE	67.5 ± 0.8	0.195 ± 0.011	$\mathbf{2.19 \pm 0.06}$	$\mathbf{0.0557 \pm 0.0012}$
VIB, β=1e$-$3	66.9 ± 1.0	0.193 ± 0.008	$\mathbf{2.26 \pm 0.13}$	0.0570 ± 0.0017
VIB, β=1e$-$4	66.4 ± 0.5	0.197 ± 0.009	2.30 ± 0.02	0.0577 ± 0.0007
VIB, β=1e$-$5	67.9 ± 0.8	0.166 ± 0.010	2.49 ± 0.13	$\mathbf{0.0561 \pm 0.0011}$
VIB, β=0	67.1 ± 1.0	0.162 ± 0.009	2.64 ± 0.11	0.0578 ± 0.0016
MASS, β=1e$-$3	59.6 ± 0.8	0.252 ± 0.007	2.61 ± 0.11	0.0688 ± 0.0014
MASS, β=1e$-$4	66.6 ± 0.4	0.209 ± 0.009	$\mathbf{2.18 \pm 0.05}$	0.0570 ± 0.0005
MASS, β=1e$-$5	67.4 ± 1.0	0.192 ± 0.007	$\mathbf{2.22 \pm 0.07}$	$\mathbf{0.0561 \pm 0.0017}$
MASS, β=0	67.4 ± 0.3	0.189 ± 0.004	2.30 ± 0.08	$\mathbf{0.0562 \pm 0.0007}$

Table 8. Uncertainty quantification metrics (proper scoring rules) on CIFAR-10 using the ResNet20 network trained on 2,500 datapoints. Test Accuracy and Entropy of the network's predictive distribution are given for reference. Full experiment details are in Supplementary Material 7.7. Values are the mean over 4 training runs with different random seeds, plus or minus the standard deviation. Emboldened values are those for which the minimum observed mean value in the column was within one standard deviation. Lower values are better.

Method	Test Accuracy	Entropy	NLL	Brier Score
SoftmaxCE	50.0 ± 0.7	0.349 ± 0.005	$\mathbf{2.98 \pm 0.06}$	$\mathbf{0.0833 \pm 0.0012}$
VIB, β=1e−3	49.5 ± 1.1	0.363 ± 0.005	3.10 ± 0.11	$\mathbf{0.0836 \pm 0.0020}$
VIB, β=1e−4	49.4 ± 1.0	0.372 ± 0.016	$\mathbf{3.02 \pm 0.10}$	$\mathbf{0.0833 \pm 0.0016}$
VIB, β=1e−5	50.0 ± 1.1	0.306 ± 0.021	3.48 ± 0.15	0.0849 ± 0.0013
VIB, β=0	50.6 ± 0.8	0.271 ± 0.019	3.80 ± 0.15	0.0850 ± 0.0007
MASS, β=1e−3	38.2 ± 0.7	0.469 ± 0.012	3.75 ± 0.08	0.1010 ± 0.0017
MASS, β=1e−4	49.9 ± 1.0	0.344 ± 0.001	3.24 ± 0.08	$\mathbf{0.0837 \pm 0.0017}$
MASS, β=1e−5	50.1 ± 0.5	0.277 ± 0.008	3.81 ± 0.11	0.0859 ± 0.0005
MASS, β=0	50.2 ± 1.0	0.265 ± 0.009	3.96 ± 0.15	0.0861 ± 0.0020

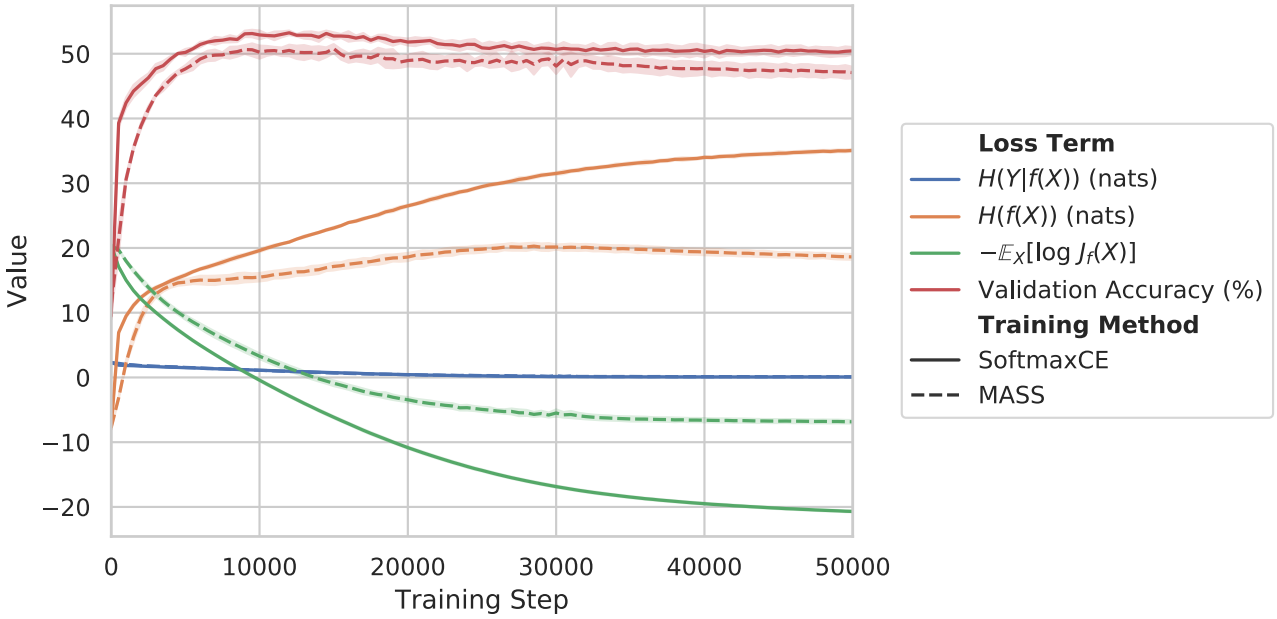

Figure 2. Estimated value of each term in the MASS Learning loss function, $\mathcal{L}_{MASS}(f) = H(Y|f(X)) + \beta H(f(X)) - \beta \mathbb{E}_X[\log J_f(X)]$, during training of the SmallMLP network on the CIFAR-10 dataset. The MASS training was performed with $\beta = 0.001$, though the plotted values are for the terms without being multiplied by the β coefficients. The values of these terms for SoftmaxCE training are estimated using a distribution $q_\phi(f_\theta(x)|y)$, with the distribution parameters ϕ being estimated at each training step by MLE over the training data.

deconvolution. *Neural Computation*, 7(6):1129–1159, November 1995. ISSN 0899-7667.

Clevert, D.-A., Unterthiner, T., and Hochreiter, S. Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). *arXiv:1511.07289 [cs]*, November 2015. URL http://arxiv.org/abs/1511.07289. arXiv: 1511.07289.

Cover, T. M. and Thomas, J. A. *Elements of Information Theory 2nd Edition*. Wiley-Interscience, Hoboken, NJ, 2 edition edition, July 2006. ISBN 978-0-471-24195-9.

Dinh, L., Krueger, D., and Bengio, Y. NICE: Non-linear Independent Components Estimation. *arXiv:1410.8516 [cs]*, October 2014. URL http://arxiv.org/abs/1410.8516. arXiv: 1410.8516.

Dinh, L., Sohl-Dickstein, J., and Bengio, S. Density estimation using Real NVP. *International Conference on Learning Representations*, 2017. URL https://arxiv.org/abs/1605.08803.

Dynkin, E. B. Necessary and sufficient statistics for afamily of probability distributions. *Uspekhi Mat. Nauk*, 6(1): 68–90, 1951. URL http://www.mathnet.ru/php/archive.phtml?wshow=paper&jrnid=rm&paperid=6820&option_lang=eng.

Federer, H. *Geometric Measure Theory*. Springer, New York, NY, 1969.

Goldfeld, Z., Berg, E. v. d., Greenewald, K., Melnyk, I., Nguyen, N., Kingsbury, B., and Polyanskiy, Y. Estimating Information Flow in Neural Networks. *arXiv:1810.05728 [cs, stat]*, October 2018. URL http://arxiv.org/abs/1810.05728. arXiv: 1810.05728.

Goodfellow, I., Bengio, Y., and Courville, A. *Deep Learning*. MIT Press, 2016.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

Hendrycks, D. and Gimpel, K. A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks. *International Conference on Learning Representations*, 2017. URL https://arxiv.org/abs/1610.02136v3.

Jakubovitz, D. and Giryes, R. Improving dnn robustness to adversarial attacks using jacobian regularization. *ECCV*, 2018. URL https://arxiv.org/abs/1803.08680.

James, R. G., Mahoney, J. R., and Crutchfield, J. P. Trimming the Independent Fat: Sufficient Statistics, Mutual Information, and Predictability from Effective Channel States. *Physical Review E*, 95(6), June 2017. ISSN 2470-0045, 2470-0053. doi: 10.1103/PhysRevE.95.060102. URL http://arxiv.org/abs/1702.01831. arXiv: 1702.01831.

Kingma, D. and Ba, J. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations*, December 2015. URL http://arxiv.org/abs/1412.6980.

Kolchinsky, A., Tracey, B. D., and Wolpert, D. H. Nonlinear Information Bottleneck. *arXiv:1705.02436 [cs, math, stat]*, May 2017. URL http://arxiv.org/abs/1705.02436. arXiv: 1705.02436.

Kolchinsky, A., Tracey, B. D., and Van Kuyk, S. Caveats for information bottleneck in deterministic scenarios. *International Conference on Learning Representations*, 2019. URL http://arxiv.org/abs/1808.07593. arXiv: 1808.07593.

Koliander, G., Pichler, G., Riegler, E., and Hlawatsch, F. Entropy and Source Coding for Integer-Dimensional Singular Random Variables. *IEEE Transactions on Information Theory*, 62(11):6124–6154, November 2016. ISSN 0018-9448, 1557-9654. doi: 10.1109/TIT.2016.2604248. URL http://arxiv.org/abs/1505.03337. arXiv: 1505.03337.

Krantz, S. G. and Parks, H. R. *Geometric Integration Theory*. Birkhuser, Basel, Switzerland, 2009.

Krizhevsky, A. Learning multiple layers of features from tiny images. 2009.

Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. *NIPS*, 2017.

Lehmann, E. L. and Scheffe, H. Completeness, Similar Regions, and Unbiased Estimation: Part I. *Sankhy: The Indian Journal of Statistics (1933-1960)*, 10(4):305–340, 1950. ISSN 0036-4452. URL https://www.jstor.org/stable/25048038.

Nash, C., Kushman, N., and Williams, C. K. I. Inverting Supervised Representations with Autoregressive Neural Density Models. June 2018. URL https://arxiv.org/abs/1806.00400.

Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. Reading digits in natural images with unsupervised feature learning. 2011.

Novak, R., Bahri, Y., Abolafia, D. A., Pennington, J., and Sohl-Dickstein, J. Sensitivity and Generalization in Neural Networks: an Empirical Study. *International Conference on Learning Representations*, 2018. URL http://arxiv.org/abs/1802.08760. arXiv: 1802.08760.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in PyTorch. In *NIPS-W*, 2017.

Rezende, D. J. and Mohamed, S. Variational inference with normalizing flows. *International Conference on Machine Learning*, 2015. URL https://arxiv.org/abs/1505.05770.

Rifai, S., Mesnil, G., Vincent, P., Muller, X., Bengio, Y., Dauphin, Y., and Glorot, X. Higher Order Contractive Auto-Encoder. In *Machine Learning and Knowledge Discovery in Databases*, Lecture Notes in Computer Science, pp. 645–660. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-23783-6.

Ross, A. S. and Doshi-Velez, F. Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients. *AAAI*, 2018. URL https://arxiv.org/abs/1711.09404.

Saxe, A. M., Bansal, Y., Dapello, J., Advani, M., Kolchinsky, A., Tracey, B. D., and Cox, D. D. On the Information Bottleneck Theory of Deep Learning. *International Conference on Learning Representations*, February 2018. URL https://openreview.net/forum?id=ry_WPG-A-.

Shamir, O., Sabato, S., and Tishby, N. Learning and generalization with the information bottleneck. *Theoretical Computer Science*, 411(29):2696–2711, June 2010. ISSN 0304-3975. doi: 10.1016/j.tcs.2010.04.006. URL http://www.sciencedirect.com/science/article/pii/S030439751000201X.

Shwartz-Ziv, R. and Tishby, N. Opening the Black Box of Deep Neural Networks via Information. *arXiv:1703.00810 [cs]*, March 2017. URL http://arxiv.org/abs/1703.00810. arXiv: 1703.00810.

Sokolic, J., Giryes, R., Sapiro, G., and Rodrigues, M. R. D. Robust Large Margin Deep Neural Networks. *IEEE Transactions on Signal Processing*, 65(16):4265–4280, August 2017. ISSN 1053-587X, 1941-0476. doi: 10.1109/TSP.2017.2708039. URL http://arxiv.org/abs/1605.08254. arXiv: 1605.08254.

Strouse, D. and Schwab, D. J. The deterministic information bottleneck. *Neural Computation*, 29:1611–1630, 2015. URL https://arxiv.org/abs/1604.00268.

Tishby, N. and Zaslavsky, N. Deep Learning and the Information Bottleneck Principle. *2015 IEEE Information Theory Workshop (ITW)*, pp. 1–5, March 2015. URL https://arxiv.org/abs/1503.02406.

Tishby, N., Pereira, F. C., and Bialek, W. The information bottleneck method. *arXiv:physics/0004057*, April 2000. URL http://arxiv.org/abs/physics/0004057. arXiv: physics/0004057.

Varga, D., Csiszrik, A., and Zombori, Z. Gradient Regularization Improves Accuracy of Discriminative Models. *arXiv:1712.09936 [cs]*, December 2017. URL http://arxiv.org/abs/1712.09936. arXiv: 1712.09936.

Table 9. Out-of-distribution detection metrics for SmallMLP network trained on 40,000 CIFAR-10 images, with SVHN as the out-of-distribution examples. Full experiment details are in Supplementary Material 7.7. Values are the mean over 4 training runs with different random seeds, plus or minus the standard deviation. Emboldened values are those for which the maximum observed mean value in the column was within one standard deviation. WD is weight decay; D is dropout. Higher values are better.

Training Method	Test Accuracy	Detection Method	AUROC	APR In	APR Out
SoftmaxCE	52.7 ± 0.4	Entropy	0.65 ± 0.01	0.68 ± 0.01	0.61 ± 0.01
		$\max_i q_\phi(f_\theta(x)\|y_i)$	0.38 ± 0.01	0.42 ± 0.01	0.43 ± 0.01
SoftmaxCE, WD	48.1 ± 0.1	Entropy	0.65 ± 0.01	0.69 ± 0.01	0.59 ± 0.01
		$\max_i q_\phi(f_\theta(x)\|y_i)$	0.43 ± 0.01	0.43 ± 0.01	0.48 ± 0.02
SoftmaxCE, D	53.7 ± 0.3	Entropy	0.71 ± 0.01	$\mathbf{0.75 \pm 0.01}$	0.65 ± 0.01
		$\max_i q_\phi(f_\theta(x)\|y_i)$	0.33 ± 0.00	0.39 ± 0.00	0.40 ± 0.00
VIB, β=1e$-$1	46.1 ± 0.5	Entropy	0.62 ± 0.01	0.66 ± 0.01	0.57 ± 0.01
		Rate	0.47 ± 0.02	0.49 ± 0.01	0.46 ± 0.01
VIB, β=1e$-$2	51.9 ± 0.8	Entropy	0.64 ± 0.01	0.67 ± 0.01	0.59 ± 0.01
		Rate	0.58 ± 0.03	0.59 ± 0.02	0.55 ± 0.02
VIB, β=1e$-$3	51.8 ± 0.8	Entropy	0.65 ± 0.00	0.67 ± 0.01	0.61 ± 0.00
		Rate	0.52 ± 0.03	0.54 ± 0.03	0.50 ± 0.03
VIB, β=1e$-$1, D	49.5 ± 0.5	Entropy	0.68 ± 0.01	$\mathbf{0.74 \pm 0.01}$	0.60 ± 0.01
		Rate	0.34 ± 0.01	0.40 ± 0.01	0.39 ± 0.00
VIB, β=1e$-$2, D	53.6 ± 0.3	Entropy	0.69 ± 0.02	0.73 ± 0.01	0.62 ± 0.02
		Rate	0.50 ± 0.03	0.51 ± 0.02	0.51 ± 0.03
VIB, β=1e$-$3, D	54.3 ± 0.2	Entropy	0.69 ± 0.01	0.73 ± 0.01	0.62 ± 0.01
		Rate	0.45 ± 0.01	0.45 ± 0.01	0.49 ± 0.01
MASS, β=1e$-$2	46.3 ± 1.2	Entropy	0.64 ± 0.01	0.67 ± 0.01	0.61 ± 0.01
		$\max_i q_\phi(f_\theta(x)\|y_i)$	0.51 ± 0.03	0.56 ± 0.05	0.49 ± 0.01
MASS, β=1e$-$3	47.8 ± 0.8	Entropy	0.63 ± 0.02	0.65 ± 0.02	0.60 ± 0.02
		$\max_i q_\phi(f_\theta(x)\|y_i)$	0.63 ± 0.07	0.64 ± 0.08	0.60 ± 0.05
MASS, β=1e$-$4	47.9 ± 0.8	Entropy	0.63 ± 0.02	0.65 ± 0.02	0.60 ± 0.02
		$\max_i q_\phi(f_\theta(x)\|y_i)$	0.57 ± 0.06	0.58 ± 0.05	0.56 ± 0.05
MASS, β=0	48.2 ± 0.9	Entropy	0.63 ± 0.02	0.65 ± 0.02	0.59 ± 0.02
		$\max_i q_\phi(f_\theta(x)\|y_i)$	0.58 ± 0.06	0.58 ± 0.05	0.56 ± 0.05
MASS, β=1e$-$2, D	52.0 ± 0.6	Entropy	$\mathbf{0.73 \pm 0.01}$	$\mathbf{0.75 \pm 0.01}$	$\mathbf{0.67 \pm 0.01}$
		$\max_i q_\phi(f_\theta(x)\|y_i)$	0.65 ± 0.06	0.70 ± 0.06	0.58 ± 0.05
MASS, β=1e$-$3, D	53.1 ± 0.4	Entropy	$\mathbf{0.71 \pm 0.02}$	0.73 ± 0.01	0.64 ± 0.02
		$\max_i q_\phi(f_\theta(x)\|y_i)$	0.64 ± 0.10	0.66 ± 0.10	0.60 ± 0.09
MASS, β=1e$-$4, D	53.2 ± 0.1	Entropy	$\mathbf{0.73 \pm 0.01}$	$\mathbf{0.75 \pm 0.01}$	$\mathbf{0.67 \pm 0.01}$
		$\max_i q_\phi(f_\theta(x)\|y_i)$	0.65 ± 0.09	0.65 ± 0.08	0.61 ± 0.08
MASS, β=0, D	52.7 ± 0.0	Entropy	$\mathbf{0.71 \pm 0.02}$	$\mathbf{0.74 \pm 0.01}$	$\mathbf{0.65 \pm 0.02}$
		$\max_i q_\phi(f_\theta(x)\|y_i)$	0.63 ± 0.09	0.65 ± 0.08	0.59 ± 0.09

Table 10. Out-of-distribution detection metrics for SmallMLP network trained on 10,000 CIFAR-10 images, with SVHN as the out-of-distribution examples. Full experiment details are in Supplementary Material 7.7. Values are the mean over 4 training runs with different random seeds, plus or minus the standard deviation. Emboldened values are those for which the maximum observed mean value in the column was within one standard deviation. WD is weight decay; D is dropout. Higher values are better.

Training Method	Test Accuracy	Detection Method	AUROC	APR In	APR Out
SoftmaxCE	44.6 ± 0.6	Entropy	0.62 ± 0.00	0.64 ± 0.01	0.59 ± 0.00
		$\max_i q_\phi(f_\theta(x)\|y_i)$	0.36 ± 0.01	0.40 ± 0.01	0.42 ± 0.00
SoftmaxCE, WD	36.4 ± 0.9	Entropy	0.62 ± 0.02	0.62 ± 0.02	0.60 ± 0.02
		$\max_i q_\phi(f_\theta(x)\|y_i)$	0.30 ± 0.01	0.37 ± 0.00	0.39 ± 0.01
SoftmaxCE, D	44.1 ± 0.6	Entropy	0.66 ± 0.01	$\mathbf{0.69 \pm 0.01}$	0.62 ± 0.01
		$\max_i q_\phi(f_\theta(x)\|y_i)$	0.29 ± 0.01	0.37 ± 0.00	0.38 ± 0.00
VIB, β=1e$-$1	40.6 ± 0.4	Entropy	0.60 ± 0.01	0.64 ± 0.01	0.56 ± 0.01
		Rate	0.50 ± 0.02	0.52 ± 0.02	0.48 ± 0.01
VIB, β=1e$-$2	43.8 ± 0.8	Entropy	0.62 ± 0.00	0.64 ± 0.01	0.59 ± 0.01
		Rate	0.55 ± 0.03	0.57 ± 0.02	0.53 ± 0.02
VIB, β=1e$-$3	44.6 ± 0.6	Entropy	0.62 ± 0.01	0.64 ± 0.01	0.59 ± 0.01
		Rate	0.49 ± 0.04	0.52 ± 0.04	0.48 ± 0.03
VIB, β=1e$-$1, D	40.1 ± 0.5	Entropy	0.62 ± 0.00	0.65 ± 0.01	0.57 ± 0.00
		Rate	0.49 ± 0.02	0.51 ± 0.02	0.48 ± 0.01
VIB, β=1e$-$2, D	43.9 ± 0.3	Entropy	$\mathbf{0.67 \pm 0.01}$	$\mathbf{0.69 \pm 0.01}$	0.62 ± 0.00
		Rate	0.60 ± 0.02	0.61 ± 0.02	0.56 ± 0.01
VIB, β=1e$-$3, D	44.4 ± 0.4	Entropy	$\mathbf{0.67 \pm 0.01}$	$\mathbf{0.69 \pm 0.01}$	0.63 ± 0.01
		Rate	0.50 ± 0.03	0.53 ± 0.03	0.49 ± 0.02
MASS, β=1e$-$2	39.9 ± 1.2	Entropy	0.63 ± 0.02	0.64 ± 0.02	0.60 ± 0.01
		$\max_i q_\phi(f_\theta(x)\|y_i)$	0.54 ± 0.03	0.58 ± 0.04	0.50 ± 0.02
MASS, β=1e$-$3	41.5 ± 0.7	Entropy	0.61 ± 0.02	0.62 ± 0.02	0.59 ± 0.01
		$\max_i q_\phi(f_\theta(x)\|y_i)$	0.59 ± 0.07	0.60 ± 0.06	0.56 ± 0.06
MASS, β=1e$-$4	41.5 ± 1.1	Entropy	0.60 ± 0.00	0.61 ± 0.01	0.58 ± 0.00
		$\max_i q_\phi(f_\theta(x)\|y_i)$	0.55 ± 0.05	0.56 ± 0.04	0.53 ± 0.04
MASS, β=0	42.0 ± 0.6	Entropy	0.60 ± 0.02	0.61 ± 0.02	0.57 ± 0.01
		$\max_i q_\phi(f_\theta(x)\|y_i)$	0.55 ± 0.06	0.57 ± 0.04	0.54 ± 0.05
MASS, β=1e$-$2, D	41.7 ± 0.4	Entropy	$\mathbf{0.67 \pm 0.01}$	$\mathbf{0.68 \pm 0.01}$	$\mathbf{0.63 \pm 0.01}$
		$\max_i q_\phi(f_\theta(x)\|y_i)$	0.63 ± 0.04	$\mathbf{0.65 \pm 0.04}$	0.57 ± 0.04
MASS, β=1e$-$3, D	43.7 ± 0.2	Entropy	$\mathbf{0.67 \pm 0.01}$	$\mathbf{0.68 \pm 0.01}$	$\mathbf{0.63 \pm 0.01}$
		$\max_i q_\phi(f_\theta(x)\|y_i)$	$\mathbf{0.66 \pm 0.05}$	0.66 ± 0.04	$\mathbf{0.61 \pm 0.06}$
MASS, β=1e$-$4, D	43.4 ± 0.5	Entropy	$\mathbf{0.68 \pm 0.01}$	$\mathbf{0.69 \pm 0.01}$	$\mathbf{0.64 \pm 0.02}$
		$\max_i q_\phi(f_\theta(x)\|y_i)$	$\mathbf{0.64 \pm 0.07}$	$\mathbf{0.65 \pm 0.05}$	$\mathbf{0.59 \pm 0.08}$
MASS, β=0, D	43.9 ± 0.4	Entropy	$\mathbf{0.68 \pm 0.00}$	$\mathbf{0.69 \pm 0.01}$	$\mathbf{0.64 \pm 0.00}$
		$\max_i q_\phi(f_\theta(x)\|y_i)$	$\mathbf{0.65 \pm 0.04}$	$\mathbf{0.66 \pm 0.03}$	$\mathbf{0.60 \pm 0.06}$

Table 11. Out-of-distribution detection metrics for SmallMLP network trained on 2,500 CIFAR-10 images, with SVHN as the out-of-distribution examples. Full experiment details are in Supplementary Material 7.7. Values are the mean over 4 training runs with different random seeds, plus or minus the standard deviation. Emboldened values are those for which the maximum observed mean value in the column was within one standard deviation. WD is weight decay; D is dropout. Higher values are better.

Training Method	Test Accuracy	Detection Method	AUROC	APR In	APR Out
SoftmaxCE	34.2 ± 0.8	Entropy	0.61 ± 0.01	0.62 ± 0.01	0.59 ± 0.01
		$\max_i q_\phi(f_\theta(x)\|y_i)$	0.30 ± 0.02	0.38 ± 0.01	0.39 ± 0.01
SoftmaxCE, WD	23.9 ± 0.9	Entropy	$\mathbf{0.70 \pm 0.03}$	$\mathbf{0.67 \pm 0.03}$	$\mathbf{0.71 \pm 0.04}$
		$\max_i q_\phi(f_\theta(x)\|y_i)$	0.23 ± 0.02	0.36 ± 0.01	0.36 ± 0.01
SoftmaxCE, D	33.7 ± 1.1	Entropy	0.60 ± 0.01	0.62 ± 0.01	0.58 ± 0.01
		$\max_i q_\phi(f_\theta(x)\|y_i)$	0.27 ± 0.01	0.37 ± 0.00	0.37 ± 0.00
VIB, β=1e$-$1	32.2 ± 0.6	Entropy	0.58 ± 0.01	0.60 ± 0.02	0.56 ± 0.01
		Rate	0.52 ± 0.02	0.54 ± 0.02	0.49 ± 0.02
VIB, β=1e$-$2	34.6 ± 0.4	Entropy	0.60 ± 0.01	0.62 ± 0.01	0.57 ± 0.01
		Rate	0.52 ± 0.04	0.55 ± 0.04	0.48 ± 0.03
VIB, β=1e$-$3	35.6 ± 0.5	Entropy	0.59 ± 0.01	0.60 ± 0.01	0.56 ± 0.01
		Rate	0.50 ± 0.04	0.53 ± 0.03	0.48 ± 0.03
VIB, β=1e$-$1, D	29.0 ± 0.6	Entropy	0.57 ± 0.01	0.60 ± 0.01	0.53 ± 0.01
		Rate	0.45 ± 0.02	0.48 ± 0.02	0.46 ± 0.01
VIB, β=1e$-$2, D	32.5 ± 0.9	Entropy	0.62 ± 0.01	0.63 ± 0.02	0.59 ± 0.01
		Rate	0.53 ± 0.05	0.56 ± 0.04	0.52 ± 0.04
VIB, β=1e$-$3, D	34.5 ± 1.0	Entropy	0.63 ± 0.01	0.64 ± 0.02	0.60 ± 0.01
		Rate	0.56 ± 0.05	0.57 ± 0.03	0.54 ± 0.05
MASS, β=1e$-$2	29.6 ± 0.4	Entropy	0.59 ± 0.01	0.61 ± 0.01	0.56 ± 0.01
		$\max_i q_\phi(f_\theta(x)\|y_i)$	0.43 ± 0.03	0.48 ± 0.03	0.43 ± 0.01
MASS, β=1e$-$3	32.7 ± 0.8	Entropy	0.57 ± 0.01	0.59 ± 0.02	0.55 ± 0.01
		$\max_i q_\phi(f_\theta(x)\|y_i)$	0.57 ± 0.04	0.59 ± 0.04	0.54 ± 0.03
MASS, β=1e$-$4	34.0 ± 0.3	Entropy	0.57 ± 0.01	0.57 ± 0.01	0.55 ± 0.01
		$\max_i q_\phi(f_\theta(x)\|y_i)$	0.59 ± 0.03	0.58 ± 0.03	0.57 ± 0.03
MASS, β=0	34.1 ± 0.6	Entropy	0.57 ± 0.01	0.58 ± 0.01	0.55 ± 0.00
		$\max_i q_\phi(f_\theta(x)\|y_i)$	0.61 ± 0.03	0.59 ± 0.04	0.59 ± 0.04
MASS, β=1e$-$2, D	29.3 ± 1.2	Entropy	0.62 ± 0.02	$\mathbf{0.64 \pm 0.03}$	0.59 ± 0.02
		$\max_i q_\phi(f_\theta(x)\|y_i)$	0.50 ± 0.05	0.54 ± 0.05	0.47 ± 0.03
MASS, β=1e$-$3, D	31.5 ± 0.6	Entropy	0.61 ± 0.02	0.62 ± 0.03	0.58 ± 0.01
		$\max_i q_\phi(f_\theta(x)\|y_i)$	0.62 ± 0.04	$\mathbf{0.63 \pm 0.04}$	0.58 ± 0.04
MASS, β=1e$-$4, D	32.7 ± 0.8	Entropy	0.61 ± 0.02	0.61 ± 0.03	0.59 ± 0.01
		$\max_i q_\phi(f_\theta(x)\|y_i)$	0.65 ± 0.04	$\mathbf{0.63 \pm 0.04}$	0.62 ± 0.05
MASS, β=0, D	32.2 ± 1.1	Entropy	0.63 ± 0.01	0.64 ± 0.02	0.61 ± 0.01
		$\max_i q_\phi(f_\theta(x)\|y_i)$	$\mathbf{0.65 \pm 0.05}$	$\mathbf{0.64 \pm 0.05}$	0.62 ± 0.06

Table 12. Out-of-distribution detection metrics for ResNet20 network trained on 40,000 CIFAR-10 images, with SVHN as the out-of-distribution examples. Full experiment details are in Supplementary Material 7.7. Values are the mean over 4 training runs with different random seeds, plus or minus the standard deviation. Emboldened values are those for which the maximum observed mean value in the column was within one standard deviation. Higher values are better.

Training Method	Test Accuracy	Detection Method	AUROC	APR In	APR Out
SoftmaxCE	81.7 ± 0.3	Entropy	$\mathbf{0.77 \pm 0.02}$	0.81 ± 0.02	0.70 ± 0.02
		$\max_i q_\phi(f_\theta(x)\|y_i)$	0.59 ± 0.03	0.62 ± 0.03	0.55 ± 0.02
VIB, β=1e$-$3	81.0 ± 0.3	Entropy	0.74 ± 0.02	0.79 ± 0.02	0.67 ± 0.02
		Rate	0.55 ± 0.04	0.57 ± 0.05	0.51 ± 0.03
VIB, β=1e$-$4	81.2 ± 0.4	Entropy	0.73 ± 0.02	0.76 ± 0.03	0.66 ± 0.02
		Rate	0.50 ± 0.02	0.54 ± 0.02	0.48 ± 0.01
VIB, β=1e$-$5	80.9 ± 0.5	Entropy	0.75 ± 0.02	0.80 ± 0.02	0.67 ± 0.02
		Rate	0.18 ± 0.05	0.34 ± 0.01	0.34 ± 0.01
VIB, β=0	81.5 ± 0.2	Entropy	$\mathbf{0.79 \pm 0.02}$	$\mathbf{0.84 \pm 0.02}$	$\mathbf{0.73 \pm 0.04}$
		Rate	0.11 ± 0.03	0.32 ± 0.01	0.32 ± 0.01
MASS, β=1e$-$3	75.8 ± 0.5	Entropy	0.74 ± 0.03	0.77 ± 0.03	0.69 ± 0.03
		$\max_i q_\phi(f_\theta(x)\|y_i)$	0.37 ± 0.04	0.43 ± 0.02	0.42 ± 0.02
MASS, β=1e$-$4	80.6 ± 0.5	Entropy	$\mathbf{0.76 \pm 0.04}$	$\mathbf{0.80 \pm 0.04}$	$\mathbf{0.70 \pm 0.05}$
		$\max_i q_\phi(f_\theta(x)\|y_i)$	0.48 ± 0.06	0.53 ± 0.05	0.47 ± 0.04
MASS, β=1e$-$5	81.6 ± 0.4	Entropy	0.77 ± 0.01	$\mathbf{0.82 \pm 0.01}$	$\mathbf{0.71 \pm 0.02}$
		$\max_i q_\phi(f_\theta(x)\|y_i)$	0.54 ± 0.03	0.58 ± 0.03	0.51 ± 0.02
MASS, β=0	81.5 ± 0.2	Entropy	$\mathbf{0.79 \pm 0.03}$	$\mathbf{0.83 \pm 0.02}$	$\mathbf{0.73 \pm 0.03}$
		$\max_i q_\phi(f_\theta(x)\|y_i)$	0.49 ± 0.04	0.54 ± 0.04	0.47 ± 0.02

Table 13. Out-of-distribution detection metrics for ResNet20 network trained on 10,000 CIFAR-10 images, with SVHN as the out-of-distribution examples. Full experiment details are in Supplementary Material 7.7. Values are the mean over 4 training runs with different random seeds, plus or minus the standard deviation. Emboldened values are those for which the maximum observed mean value in the column was within one standard deviation. Higher values are better.

Training Method	Test Accuracy	Detection Method	AUROC	APR In	APR Out
SoftmaxCE	67.5 ± 0.8	Entropy	0.64 ± 0.02	0.68 ± 0.02	0.58 ± 0.02
		$\max_i q_\phi(f_\theta(x)\|y_i)$	0.59 ± 0.03	0.61 ± 0.03	0.57 ± 0.04
VIB, β=1e$-$3	66.9 ± 1.0	Entropy	0.59 ± 0.02	0.63 ± 0.04	0.54 ± 0.02
		Rate	$\mathbf{0.72 \pm 0.05}$	$\mathbf{0.73 \pm 0.05}$	$\mathbf{0.67 \pm 0.05}$
VIB, β=1e$-$4	66.4 ± 0.5	Entropy	0.59 ± 0.01	0.63 ± 0.02	0.54 ± 0.01
		Rate	0.59 ± 0.07	0.60 ± 0.07	0.56 ± 0.06
VIB, β=1e$-$5	67.9 ± 0.8	Entropy	0.61 ± 0.03	0.65 ± 0.04	0.56 ± 0.03
		Rate	0.39 ± 0.07	0.42 ± 0.03	0.43 ± 0.04
VIB, β=0	67.1 ± 1.0	Entropy	0.64 ± 0.01	0.68 ± 0.01	0.58 ± 0.01
		Rate	0.32 ± 0.03	0.39 ± 0.01	0.39 ± 0.01
MASS, β=1e$-$3	59.6 ± 0.8	Entropy	0.59 ± 0.02	0.62 ± 0.03	0.56 ± 0.02
		$\max_i q_\phi(f_\theta(x)\|y_i)$	0.49 ± 0.07	0.46 ± 0.06	0.48 ± 0.08
MASS, β=1e$-$4	66.6 ± 0.4	Entropy	0.62 ± 0.02	0.67 ± 0.02	0.56 ± 0.03
		$\max_i q_\phi(f_\theta(x)\|y_i)$	0.61 ± 0.05	0.61 ± 0.05	0.60 ± 0.05
MASS, β=1e$-$5	67.4 ± 1.0	Entropy	0.64 ± 0.02	0.69 ± 0.03	0.58 ± 0.01
		$\max_i q_\phi(f_\theta(x)\|y_i)$	0.61 ± 0.08	0.61 ± 0.06	$\mathbf{0.61 \pm 0.09}$
MASS, β=0	67.4 ± 0.3	Entropy	0.64 ± 0.01	0.68 ± 0.02	0.58 ± 0.01
		$\max_i q_\phi(f_\theta(x)\|y_i)$	0.55 ± 0.05	0.56 ± 0.04	0.54 ± 0.05

Table 14. Out-of-distribution detection metrics for ResNet20 network trained on 2,500 CIFAR-10 images, with SVHN as the out-of-distribution examples. Full experiment details are in Supplementary Material 7.7. Values are the mean over 4 training runs with different random seeds, plus or minus the standard deviation. Emboldened values are those for which the maximum observed mean value in the column was within one standard deviation. Higher values are better.

Training Method	Test Accuracy	Detection Method	AUROC	APR In	APR Out
SoftmaxCE	50.0 ± 0.7	Entropy	0.51 ± 0.01	0.52 ± 0.02	0.49 ± 0.01
		$\max_i q_\phi(f_\theta(x)\|y_i)$	0.63 ± 0.04	0.62 ± 0.03	0.63 ± 0.04
VIB, β=1e$-$3	49.5 ± 1.1	Entropy	0.48 ± 0.05	0.50 ± 0.05	0.47 ± 0.03
		Rate	$\mathbf{0.68 \pm 0.07}$	$\mathbf{0.68 \pm 0.05}$	$\mathbf{0.66 \pm 0.08}$
VIB, β=1e$-$4	49.4 ± 1.0	Entropy	0.47 ± 0.05	0.50 ± 0.05	0.47 ± 0.03
		Rate	$\mathbf{0.66 \pm 0.09}$	$\mathbf{0.65 \pm 0.08}$	$\mathbf{0.66 \pm 0.09}$
VIB, β=1e$-$5	50.0 ± 1.1	Entropy	0.48 ± 0.05	0.49 ± 0.05	0.48 ± 0.03
		Rate	0.59 ± 0.10	0.55 ± 0.08	0.61 ± 0.09
VIB, β=0	50.6 ± 0.8	Entropy	0.51 ± 0.07	0.54 ± 0.08	0.50 ± 0.06
		Rate	$\mathbf{0.52 \pm 0.20}$	0.53 ± 0.15	0.56 ± 0.17
MASS, β=1e$-$3	38.2 ± 0.7	Entropy	0.48 ± 0.04	0.50 ± 0.04	0.47 ± 0.03
		$\max_i q_\phi(f_\theta(x)\|y_i)$	0.54 ± 0.11	0.48 ± 0.06	0.51 ± 0.08
MASS, β=1e$-$4	49.9 ± 1.0	Entropy	0.49 ± 0.04	0.51 ± 0.05	0.48 ± 0.03
		$\max_i q_\phi(f_\theta(x)\|y_i)$	$\mathbf{0.72 \pm 0.08}$	$\mathbf{0.71 \pm 0.08}$	$\mathbf{0.73 \pm 0.08}$
MASS, β=1e$-$5	50.1 ± 0.5	Entropy	0.50 ± 0.06	0.51 ± 0.06	0.49 ± 0.04
		$\max_i q_\phi(f_\theta(x)\|y_i)$	$\mathbf{0.69 \pm 0.10}$	$\mathbf{0.68 \pm 0.10}$	$\mathbf{0.70 \pm 0.10}$
MASS, β=0	50.2 ± 1.0	Entropy	0.51 ± 0.06	0.53 ± 0.06	0.50 ± 0.04
		$\max_i q_\phi(f_\theta(x)\|y_i)$	$\mathbf{0.69 \pm 0.07}$	$\mathbf{0.68 \pm 0.07}$	$\mathbf{0.68 \pm 0.07}$

7. Supplementary Material

7.1. Standard Definition of Minimal Sufficient Statistics

The most common phrasing of the definition of *minimal sufficient statistic* is:

Definition 3 (Minimal Sufficient Statistic)**.** A sufficient statistic $f(X)$ for Y is *minimal* if for any other sufficient statistic $h(X)$ there exists a measurable function g such that $f = g \circ h$ almost everywhere.

Some references do not explicitly mention the "measurability" and "almost everywhere" conditions on g, but since we are in the probabilistic setting it is this definition of $f = g \circ h$ that is meaningful.

Our preferred phrasing of the definition of *minimal sufficient statistic*, which we use in our Introduction, is:

Definition 4 (Minimal Sufficient Statistic)**.** A sufficient statistic $f(X)$ for Y is *minimal* if for any measurable function g, $g(f(X))$ is no longer sufficient for Y unless g is invertible almost everywhere (i.e. there exist a measurable function g^{-1} and a set $\mathcal{A}$ such that $g^{-1}(g(x)) = x$ for all $x \in \mathcal{A}$ and the event $\{X \in \mathcal{A}^c\}$ has probability zero).

The equivalence of Definition 3 and Definition 4 is given by the following lemma:

Lemma 2. *Assume that there exists a minimal sufficient statistic $h(X)$ for Y by Definition 3. Then a sufficient statistic $f(X)$ is minimal in the sense of Definition 3 if and only if it is minimal in the sense of Definition 4.*

Proof. We first assume that $f(X)$ is minimal in the sense of Definition 3. Let g be any measurable function such that $g(f(X))$ is sufficient for Y. By the minimality (Def. 3) of f there must exist a measurable function $\tilde{g}$ such that $\tilde{g}(g(f(x))) = f(x)$ almost everywhere. This proves that f is minimal in the sense of Definition 4.

Now assume that $f(X)$ is minimal in the sense of Definition 4 and let $\tilde{f}(X)$ be another sufficient statistic. Because h is minimal (Def. 3), there exist g_1 such that $h = g_1 \circ \tilde{f}$ almost everywhere and g_2 such that $h = g_2 \circ f$ almost everywhere. Because f is minimal (Def. 4), g_2 must be one-to-one almost everywhere, i.e. there exists a $\tilde{g}_2$ such that $\tilde{g}_2 \circ h = \tilde{g}_2 \circ g_2 \circ f = f$ almost everywhere. In turn, we obtain that $\tilde{g}_2 \circ g_1 \circ \tilde{f} = f$ almost everywhere, and since $\tilde{f}$ was arbitrary this proves the minimality of f in the sense of Definition 3. □

7.2. The Mutual Information Between the Input and Output of a Deep Network is Infinite

Typically the mutual information between continuous random variables X and Y is given by

$$I(X,Y) = \int p(x,y) \log \frac{p(x,y)}{p(x)p(y)} \mathrm{d}x\mathrm{d}y,$$

but this quantity is only defined when the joint density $p(x,y)$ is integrable, which it is not in the case that $Y = f(X)$. (The technical term for $p(x,y)$ in this case is a "singular distribution".) Instead, to compute $I(X, f(X))$ we must refer to the "master definition" of mutual information (Cover & Thomas, 2006), which is

$$I(X,Y) = \sup_{\mathcal{P},\mathcal{Q}} I([X]_{\mathcal{P}}, [Y]_{\mathcal{Q}}), \tag{3}$$

where $\mathcal{P}$ and $\mathcal{Q}$ are finite partitions of the range of X and Y, respectively, and $[X]_{\mathcal{P}}$ is the random variable obtained by quantizing X using partition $\mathcal{P}$, and analogously for $[Y]_{\mathcal{Q}}$.

From this definition, we can prove the following Lemma:

Lemma 3. *If X and Y are continuous random variables, and there are open sets O_X and O_Y in the support of X and Y, respectively, such that $y = f(x)$ for $x \in O_X$ and $y \in O_Y$, then $I(X,Y) = \infty$.*

This includes all X and Y where $Y = f(X)$ for an f that is continuous somewhere on its domain, e.g., any deep network (considered as a function from an input vector to an output vector).

Proof. Suppose X and Y satisfy the conditions of the lemma. Let O_X and O_Y be open sets with $f(O_X) = O_Y$ and $\mathbb{P}[X \in O_X] =: \delta > 0$, which exist by the lemma's assumptions. Then let $\mathcal{P}^n_{O_Y}$ be a partition of O_Y into n disjoint sets. Because Y is continuous and hence does not have any atoms, we may assume that the probability of Y belonging to each element of $\mathcal{P}^n_{O_Y}$ is equal to the same nonzero value δ/n. Denote by $\mathcal{P}^n_{O_X}$ the partition of O_X into n disjoint sets, where

each set in $\mathcal{P}^n_{O_X}$ is the preimage of one of the sets in $\mathcal{P}^n_{O_Y}$. We can construct partitions of the whole domains of X and Y as $\mathcal{P}^n_{O_X} \cup O_X^c$ and $\mathcal{P}^n_{O_Y} \cup O_Y^c$, respectively. Using these partitions in (3), we obtain

$$\begin{aligned} I(X,Y) &\geq (1-\delta)\log(1-\delta) + \sum_{A \in [X]_{\mathcal{P}^n_{O_X}}} \mathbb{P}[X \in A, Y \in f(A)] \log \frac{\mathbb{P}[X \in A, Y \in f(A)]}{\mathbb{P}[X \in A]\mathbb{P}[Y \in f(A)]} \\ &= (1-\delta)\log(1-\delta) + n\frac{\delta}{n}\log\frac{\frac{\delta}{n}}{\frac{\delta}{n}\frac{\delta}{n}} \\ &= (1-\delta)\log(1-\delta) + \delta\log\frac{n}{\delta}. \end{aligned}$$

By letting n go to infinity, we can see that the supremum in Eq. 3 is infinity. □

7.3. The Change of Variables Formula for Non-invertible Mappings

The change of variables formula is widely used in machine learning and is key to recent results in density estimation and generative modeling like normalizing flows (Rezende & Mohamed, 2015), NICE (Dinh et al., 2014), and Real NVP (Dinh et al., 2017). But all uses of the change of variables formula in the machine learning literature that we are aware of use it with respect to bijective mappings between random variables, despite the formula also being applicable to non-invertible mappings between random variables. To address this gap, we offer the following brief tutorial.

The familiar form of the change of variables formula for a random variable X with density $p(x)$ and a bijective, differentiable function $f\colon \mathbb{R}^d \to \mathbb{R}^d$ is

$$\int_{\mathbb{R}^d} p(x) J_f(x)\,\mathrm{d}x = \int_{\mathbb{R}^d} p(f^{-1}(y))\,\mathrm{d}y. \tag{4}$$

where $J_f(x) = \left|\det \frac{\partial f(x)}{\partial x^{\mathrm{T}}}\right|$.

A slightly more general phrasing of Equation 4 is

$$\int_{f^{-1}(\mathcal{B})} g(x) J_f(x)\,\mathrm{d}x = \int_{\mathcal{B}} g(f^{-1}(y))\,\mathrm{d}y. \tag{5}$$

where $g\colon \mathbb{R}^d \to \mathbb{R}$ is any non-negative measurable function, and $\mathcal{B} \subseteq \mathbb{R}^d$ is any measurable subset of $\mathbb{R}^d$.

We can extend Equation 5 to work in the case that f is not invertible. To do this, we must address two issues. First, if f is not invertible, then $f^{-1}(y)$ is not a single point but rather a set. Second, if f is not invertible, then the Jacobian matrix $\frac{\partial f(x)}{\partial x^{\mathrm{T}}}$ may not be square, and thus has no well defined determinant. Both issues can be resolved and lead to the following change of variables theorem (Krantz & Parks, 2009), which is based on the so-called coarea formula (Federer, 1969).

Theorem 4. *Let $f\colon \mathbb{R}^d \to \mathbb{R}^r$ with $r \leq d$ be a differentiable function, $g\colon \mathbb{R}^d \to \mathbb{R}$ a non-negative measurable function, $\mathcal{B} \subseteq \mathbb{R}^d$ a measurable set, and $J_f(x) = \sqrt{\det\left(\frac{\partial f(x)}{\partial x^{\mathrm{T}}}\left(\frac{\partial f(x)}{\partial x^{\mathrm{T}}}\right)^{\mathrm{T}}\right)}$. Then*

$$\int_{f^{-1}(\mathcal{B})} g(x) J_f(x)\,\mathrm{d}x = \int_{\mathcal{B}} \int_{f^{-1}(y)} g(x)\,\mathrm{d}\mathscr{H}^{d-r}(x)\,\mathrm{d}y. \tag{6}$$

where $\mathscr{H}^{d-r}$ is the $(d-r)$-dimensional Hausdorff measure (one can think of this as a measure for lower-dimensional structures in high-dimensional space, e.g. the area of 2-dimensional surfaces in 3-dimensional space).[3]

We see in Theorem 4 that Equation 6 looks a lot like Equations 4 and 5, but with $f^{-1}(y)$ replaced by an integral over the set $f^{-1}(y)$, which for almost every y is a $(d-r)$-dimensional set. And if f in Equation 6 happens to be bijective, Equation 6 reduces to Equation 5.

[3] In what follows, we will sometimes replace g by g/J_f such that the Jacobian appears on the right-hand side. Furthermore, we will not only use non-negative g. This can be justified by splitting g into positive and negative parts provided that either part results in a finite integral.

We also see that the Jacobian determinant in Equation 5 was replaced by the so-called r-dimensional Jacobian

$$\sqrt{\det\left(\frac{\partial f(x)}{\partial x^{\mathrm{T}}}\left(\frac{\partial f(x)}{\partial x^{\mathrm{T}}}\right)^{\mathrm{T}}\right)}$$

in Equation 6. A word of caution is in order, as the r-dimensional Jacobian does not have the same nice properties for concatenated functions as does the Jacobian in the bijective case. In particular, we cannot calculate $J_{f_2\circ f_1}$ based on the values of J_{f_1} and J_{f_2} because the product $\frac{\partial f_2(x)}{\partial x^{\mathrm{T}}}\frac{\partial f_1(x)}{\partial x^{\mathrm{T}}}\left(\frac{\partial f_2(x)}{\partial x^{\mathrm{T}}}\frac{\partial f_1(x)}{\partial x^{\mathrm{T}}}\right)^{\mathrm{T}}$ does not decompose into a product of $\frac{\partial f_2(x)}{\partial x^{\mathrm{T}}}\left(\frac{\partial f_2(x)}{\partial x^{\mathrm{T}}}\right)^{\mathrm{T}}$ and $\frac{\partial f_1(x)}{\partial x^{\mathrm{T}}}\left(\frac{\partial f_1(x)}{\partial x^{\mathrm{T}}}\right)^{\mathrm{T}}$. In other words, the trick used in techniques like normalizing flows and NICE to compute determinants of deep networks for use in the change of variables formula by decomposing the network's Jacobian into the product of layerwise Jacobians does not work straightforwardly in the case of non-invertible mappings.

7.4. Motivation for Conserved Differential Information

First, we present an alternative definition of conditional entropy that is meaningful for singular distributions (e.g., the joint distribution $p(X, f(X))$ for a function f). More information on this definition can be found in Koliander et al. (2016).

7.4.1. Singular Conditional Entropy

Assume that the random variable X has a probability density function $p_X(x)$ on $\mathbb{R}^d$. For a given differentiable function $f\colon \mathbb{R}^d \to \mathbb{R}^r$ ($r \leq d$), we want to analyze the conditional differential entropy $H(X|f(X))$. Following Koliander et al. (2016), we define this quantity as:

$$H(X|f(X)) = -\int_{\mathbb{R}^r} p_{f(X)}(y) \int_{f^{-1}(y)} \theta^{d-r}_{\Pr\{X\in\cdot|f(X)=y\}}(x) \log\left(\theta^{d-r}_{\Pr\{X\in\cdot|f(X)=y\}}(x)\right) \mathrm{d}\mathscr{H}^{d-r}(x)\,\mathrm{d}y \tag{7}$$

where $\mathscr{H}^{d-r}$ denotes $(d-r)$-dimensional Hausdorff measure. The function $p_{f(X)}$ is the probability density function of the random variable $f(X)$. Although $\theta^{d-r}_{\Pr\{X\in\cdot|f(X)=y\}}$ can also be interpreted as a probability density, it is not the commonly used density with respect to Lebesgue measure (which does not exist for $X|f(X)=y$) but a density with respect to a lower-dimensional Hausdorff measure. We will analyze the two functions $p_{f(X)}$ and $\theta^{d-r}_{\Pr\{X\in\cdot|f(X)=y\}}$ in more detail. The density $p_{f(X)}$ is defined by the relation

$$\int_{f^{-1}(\mathcal{B})} p_X(x)\,\mathrm{d}x = \int_{\mathcal{B}} p_{f(X)}(y)\,\mathrm{d}y\,, \tag{8}$$

which has to hold for every measurable set $\mathcal{B}\subseteq\mathbb{R}^r$. Using the coarea formula (or the related change-of-variables theorem), we see that

$$\int_{f^{-1}(\mathcal{B})} p_X(x)\,\mathrm{d}x = \int_{\mathcal{B}}\int_{f^{-1}(y)} \frac{p_X(x)}{J_f(x)}\,\mathrm{d}\mathscr{H}^{d-r}(x)\,\mathrm{d}y\,, \tag{9}$$

where $J_f(x) = \sqrt{\det\left(\frac{\partial f(x)}{\partial x^{\mathrm{T}}}\left(\frac{\partial f(x)}{\partial x^{\mathrm{T}}}\right)^{\mathrm{T}}\right)}$ is the r-dimensional Jacobian determinant. Thus, we identified

$$p_{f(X)}(y) = \int_{f^{-1}(y)} \frac{p_X(x)}{J_f(x)}\,\mathrm{d}\mathscr{H}^{d-r}(x)\,. \tag{10}$$

The second function, namely $\theta^{d-r}_{\Pr\{X\in\cdot|f(X)=y\}}$, is the Radon-Nikodym derivative of the conditional probability $\Pr\{X\in\cdot|f(X)=y\}$ with respect to $\mathscr{H}^{d-r}$ restricted to the set where $X|f(X)=y$ has positive probability (in the end, this will be the set $f^{-1}(y)$). To understand this function, we have to know something about the conditional distribution of X given $f(X)$. Formally, a (regular) conditional probability $\Pr\{X\in\cdot|f(X)=y\}$ has to satisfy three conditions:

- $\Pr\{X\in\cdot|f(X)=y\}$ is a probability measure for each fixed $y\in\mathbb{R}^r$.

- $\Pr\{X \in \mathcal{A} | f(X) = \cdot\}$ is measurable for each fixed measurable set $\mathcal{A} \subseteq \mathbb{R}^d$.
- For measurable sets $\mathcal{A} \subseteq \mathbb{R}^d$ and $\mathcal{B} \subseteq \mathbb{R}^r$, we have

$$\Pr\{(X, f(X)) \in \mathcal{A} \times \mathcal{B}\} = \int_{\mathcal{B}} \Pr\{X \in \mathcal{A} | f(X) = y\} p_{f(X)}(y) \, \mathrm{d}y \,. \tag{11}$$

In our setting, (11) becomes

$$\int_{\mathcal{A} \cap f^{-1}(\mathcal{B})} p_X(x) \, \mathrm{d}x = \int_{\mathcal{B}} \Pr\{X \in \mathcal{A} | f(X) = y\} p_{f(X)}(y) \, \mathrm{d}y \,. \tag{12}$$

Choosing

$$\Pr\{X \in \mathcal{A} | f(X) = y\} = \frac{1}{p_{f(X)}(y)} \int_{\mathcal{A} \cap f^{-1}(y)} \frac{p_X(x)}{J_f(x)} \, \mathrm{d}\mathscr{H}^{d-r}(x) \,, \tag{13}$$

the right-hand side in (12) becomes

$$\begin{aligned} \int_{\mathcal{B}} \Pr\{X \in \mathcal{A} | f(X) = y\} p_{f(X)}(y) \, \mathrm{d}y &= \int_{\mathcal{B}} \int_{\mathcal{A} \cap f^{-1}(y)} \frac{p_X(x)}{J_f(x)} \, \mathrm{d}\mathscr{H}^{d-r}(x) \, \mathrm{d}y \\ &= \int_{\mathcal{A} \cap f^{-1}(\mathcal{B})} p_X(x) \, \mathrm{d}x \,, \end{aligned} \tag{14}$$

where the final equality is again an application of the coarea formula. Thus, we identified

$$\theta^{d-r}_{\Pr\{X \in \cdot | f(X) = y\}}(x) = \frac{p_X(x)}{J_f(x) \, p_{f(X)}(y)} \,. \tag{15}$$

Although things might seem complicated up to this point, they simplify significantly once we put everything together. In particular, inserting (15) into (7), we obtain

$$\begin{aligned} H(X|f(X)) &= -\int_{\mathbb{R}^r} p_{f(X)}(y) \int_{f^{-1}(y)} \frac{p_X(x)}{J_f(x) \, p_{f(X)}(y)} \log\left(\frac{p_X(x)}{J_f(x) \, p_{f(X)}(y)}\right) \mathrm{d}\mathscr{H}^{d-r}(x) \, \mathrm{d}y \\ &= -\int_{\mathbb{R}^r} \int_{f^{-1}(y)} \frac{p_X(x)}{J_f(x)} \log\left(\frac{p_X(x)}{J_f(x) \, p_{f(X)}(y)}\right) \mathrm{d}\mathscr{H}^{d-r}(x) \, \mathrm{d}y \\ &= -\int_{\mathbb{R}^d} p_X(x) \log\left(\frac{p_X(x)}{J_f(x) \, p_{f(X)}(f(x))}\right) \mathrm{d}x \qquad (16) \\ &= H(X) + \int_{\mathbb{R}^d} p_X(x) \log\big(J_f(x) p_{f(X)}(f(x))\big) \, \mathrm{d}x \\ &= H(X) + \int_{\mathbb{R}^d} p_X(x) \log\big(p_{f(X)}(f(x))\big) \, \mathrm{d}x + \int_{\mathbb{R}^d} p_X(x) \log\big(J_f(x)\big) \, \mathrm{d}x \\ &= H(X) + \int_{\mathbb{R}^r} \int_{f^{-1}(y)} \frac{p_X(x)}{J_f(x)} \log\big(p_{f(X)}(f(x))\big) \, \mathrm{d}\mathscr{H}^{d-r}(x) \, \mathrm{d}y + \mathbb{E}\big[\log\big(J_f(X)\big)\big] \qquad (17) \\ &= H(X) + \int_{\mathbb{R}^r} \int_{f^{-1}(y)} \frac{p_X(x)}{J_f(x)} \, \mathrm{d}\mathscr{H}^{d-r}(x) \log\big(p_{f(X)}(y)\big) \, \mathrm{d}y + \mathbb{E}\big[\log\big(J_f(X)\big)\big] \\ &= H(X) + \int_{\mathbb{R}^r} p_{f(X)}(y) \log\big(p_{f(X)}(y)\big) \, \mathrm{d}y + \mathbb{E}\big[\log\big(J_f(X)\big)\big] \\ &= H(X) - H(f(X)) + \mathbb{E}\big[\log\big(J_f(X)\big)\big] \qquad (18) \end{aligned}$$

where (16) and (17) hold by the coarea formula.

So, altogether we have that for a random variable X and a function f, the singular conditional entropy between X and $f(X)$ is

$$H(X|f(X)) = H(X) - H(f(X)) + \mathbb{E}\big[\log\big(J_f(X)\big)\big]. \tag{19}$$

This quantity can loosely be interpreted as being the difference in differential entropies between X and $f(X)$ but with an additional term that corrects for any "uninformative" scaling that f does.

7.4.2. CONSERVED DIFFERENTIAL INFORMATION

For random variables that are not related by a deterministic function, mutual information can be expanded as

$$I(X,Y) = H(X) - H(X|Y) \tag{20}$$

where $H(X)$ and $H(X|Y)$ are differential entropy and conditional differential entropy, respectively. As we would like to measure information between random variables that are deterministically dependent, we can mimic this behavior by defining for a Lipschitz continuous mapping f:

$$C(X, f(X)) := H(X) - H(X|f(X))\,. \tag{21}$$

By (18), this can be simplified to

$$C(X, f(X)) = H(f(X)) - \mathbb{E}\big[\log\big(J_f(X)\big)\big] \tag{22}$$

yielding our definition of CDI.

7.5. Proof of CDI Data Processing Inequality

CDI Data Processing Inequality (Theorem 1)

For Lipschitz continuous functions f and g with the same output space,

$$C(X, f(X)) \geq C(X, g(f(X))$$

with equality if and only if g is one-to-one almost everywhere.

Proof. We calculate the difference between $C(X, f(X))$ and $C(X, g(f(X)))$.

$$\begin{aligned}
&C(X, f(X)) - C(X, g(f(X))) && (23)\\
&\quad = H(f(X)) - \mathbb{E}_X\big[\log J_f(X)\big] - H(g(f(X))) + \mathbb{E}_X\big[\log J_{g\circ f}(X)\big]\\
&\quad = H(f(X)) - H(g(f(X))) + \mathbb{E}_X\left[\log \frac{J_g(f(X))J_f(X)}{J_f(X)}\right] && (24)\\
&\quad = -\mathbb{E}_X[\log p_{f(X)}(f(X))] + \mathbb{E}_X\left[\log\left(\sum_{z\in g^{-1}(g(f(X)))} \frac{p_{f(X)}(f(z))}{J_g(f(z))}\right)\right] + \mathbb{E}_X[\log J_g(f(X))] && (25)\\
&\quad = \mathbb{E}_X\left[\log\left(\frac{\sum_{z\in g^{-1}(g(f(X)))} \frac{p_{f(X)}(f(z))}{J_g(f(z))}}{\frac{p_{f(X)}(f(X))}{J_g(f(X))}}\right)\right] && (26)
\end{aligned}$$

where (24) holds because the Jacobian determinant $J_{g\circ f}$ can be decomposed as g has the same domain and codomain and (25) holds because the probability density function of $g(f(X))$ can be calculated as $p_{g(f(X))}(z) = \sum_{z\in g^{-1}(g(f(X)))} \frac{p_{f(X)}(f(z))}{J_g(f(z))}$ using a change of variables argument. The resulting term in (26) is clearly always nonnegative which proves the inequality.

To prove the equality statement, we first assume that (26) is zero. In this case, $\sum_{z\in g^{-1}(g(f(x)))} \frac{p_{f(X)}(f(z))}{J_g(f(z))} = \frac{p_{f(X)}(f(x))}{J_g(f(x))}$ almost everywhere. Of course, we also have that $p_{f(X)}(f(x)) > 0$ almost everywhere. Thus, there exists a set $\mathcal{A}$ of probability one such that $\sum_{z\in g^{-1}(g(f(x)))} \frac{p_{f(X)}(f(z))}{J_g(f(z))} = \frac{p_{f(X)}(f(x))}{J_g(f(x))}$ and $p_{f(X)}(f(x)) > 0$ for all $x \in \mathcal{A}$. In particular, the set $g^{-1}(g(f(x))) \cap \mathcal{A} = \{f(x)\}$ and hence g is one-to-one almost everywhere.

For the other direction, assume that there exists $\tilde{g}$ such that $\tilde{g}(g(f(x))) = f(x)$ almost everywhere. We can assume without loss of generality that $p_{f(X)}(f(x)) = 0$ for all x that do not satisfy this equation. Restricting the expectation in (26) to the values that satisfy $\tilde{g}(g(f(x))) = f(x)$ does not change the expectation and gives the value zero. □

7.6. Theorem 3 Only Holds in the Reverse Direction for Continuous X

The specific claim we are making is as follows:

Theorem 5. *Let X be a continuous random variable drawn according to a distribution $p(X|Y)$ determined by the discrete random variable Y. Let $\mathcal{F}$ be the set of measurable functions of X to any target space. If $f(X)$ is a minimal sufficient statistic of X for Y then*

$$f \in \arg\min_{S \in \mathcal{F}} I(X, S(X)) \\ s.t.\ I(S(X), Y) = \max_{S' \in \mathcal{F}} I(S'(X), Y). \tag{27}$$

However, there may exist a function f satisfying (27) such that $f(X)$ is not a minimal sufficient statistic.

Proof. First, we prove the forward direction. According to Lemma 1, $Z = f(X)$ is a sufficient statistic for Y if and only if $I(Z, Y) = I(X, Y) = \max_{S'} I(S'(X), Y)$. To show the minimality condition in (27) for a minimal sufficient statistic, assume that there exists $S(X)$ such that $I(S(X), Y) = \max_{S' \in \mathcal{F}} I(S'(X), Y)$ and $I(X, S(X)) < I(X, f(X))$. Because f is assumed to be a minimal sufficient statistic, there exists g such that $f(X) = g(S(X))$ and by the data-processing inequality $I(X, S(X)) \geq I(X, f(X))$, a contradiction.

Next, we give an example of a function satisfying (27) such that $f(X)$ is not a minimal sufficient statistic. The example is the case when $I(X, f(X))$ is not finite, as is the case when f is a deterministic function and X is continuous. (See Lemma 3.) In this case, $I(X, S(X))$ is infinite for all deterministic, sufficient statistics S. Thus the set $\arg\min_S I(X, S(X))$ contains not only the minimal sufficient statistics, but all deterministic sufficient statistics. As a concrete example, consider two i.i.d. normally-distributed random variables with mean μ: $X = (X_1, X_2) \sim \mathcal{N}(\mu, 1)$. $T(X) = \frac{X_1 + X_2}{2}$ is a minimal sufficient statistic for μ. $T'(X) = (\frac{X_1 + X_2}{2}, X_1 \cdot X_2)$ is a non-minimal sufficient statistic for μ. However, both statistics satisfy $T, T' \in \arg\min_{S \in \mathcal{F}} I(X, S(X))$ since $\min_{S \in \mathcal{F}} I(X, S(X)) = \infty$ under the constraint $I(S(X), Y) = \max_{S' \in \mathcal{F}} I(S'(X), Y)$. □

7.7. Experiment Details

Code to reproduce all experiments is available online at `https://github.com/mwcvitkovic/MASS-Learning`.

7.7.1. Data

In all experiments above, the models were trained on the CIFAR-10 dataset (Krizhevsky, 2009). In the out-of-distribution detection experiments, the SVHN dataset (Netzer et al., 2011) was used as the out-of-distribution dataset. All channels in all datapoints were normalized to have zero mean and unit variance across their dataset. No data augmentation was used in any experiments.

7.7.2. Networks

The SmallMLP network is a 2-hidden-layer, fully-connected network with `elu` nonlinearities (Clevert et al., 2015). The first hidden layer contains 400 hidden units; the second contains 200 hidden units. Batch norm was applied after the linear mapping and before the nonlinearity of each hidden layer. Dropout, when used, was applied after the nonlinearity of each hidden layer. When used in VIB and MASS, the representation $f_\theta(x)$ was in $\mathbb{R}^{15}$, with the VIB encoder outputting parameters for a fully-covariant Gaussian distribution in $\mathbb{R}^{15}$. The marginal distribution in VIB and each component of the variational distribution q_ϕ (one component for each possible output class) in MASS were both mixtures of 10 full-covariance, 15-dimensional multivariate Gaussians.

The ResNet20 network is the 20-layer residual net of He et al. (2016). We adapted our implementation from `https://github.com/akamaster/pytorch_resnet_cifar10`, to whose authors we are very grateful. When used in VIB and MASS, the representation $f_\theta(x)$ was in $\mathbb{R}^{20}$, with the VIB encoder outputting parameters for a diagonally-covariant Gaussian distribution in $\mathbb{R}^{20}$. The marginal distribution in VIB and each component of the the variational distribution q_ϕ (one component for each possible output class) in MASS were both mixtures of 10 full-covariance, 20-dimensional multivariate Gaussians.

In experiments where a distribution $q_\phi(f_\theta(x)|y)$ is used in conjunction with a function f_θ trained by SoftmaxCE, each component of $q_\phi(f_\theta(x)|y)$ was a mixture of 10 full-covariance, 10-dimensional multivariate Gaussians, the parameters ϕ of which were estimated by MLE on the training set.

7.7.3. TRAINING

The SmallMLP network in all experiments and with all training methods was trained using the Adam optimizer (Kingma & Ba, 2015) with a learning rate of 0.0005 for 100,000 steps of stochastic gradient descent, using minibatches of size 256. All quantities we report in this paper were fully-converged to stable values by 100,000 steps. When training VIB, 5 encoder samples per datapoint were used during training, and 10 during testing. When training MASS, the learning rate of the parameters of the variational distribution q_ϕ was set at $2.5\text{e}{-5}$ to aid numerical stability.

The ResNet20 network in all experiments and with all training methods was trained using SGD with an initial learning rate of 0.1, decayed by a multiplicative factor of 0.1 at epochs 100 and 150, a momentum factor of 0.9, and minibatches of size 128. These values were taken directly from the original paper (He et al., 2016). However, unlike the original paper, we did not use data augmentation in order to keep the comparison between different numbers of training points more rigorous. This, combined with the smaller number of training points used, accounts for the around 82% accuracy we observe on CIFAR-10 compared to the around 91% accuracy in the original paper. We trained the network for 70,000 steps of stochastic gradient descent. All quantities we report in this paper were fully-converged to stable values by 70,000 steps. When training VIB, 10 encoder samples per datapoint were used during training, and 20 during testing. When training MASS, the learning rate of the parameters of the variational distribution was the same as those of the network.

The values of β we chose for VIB and MASS were selected so that the largest β value used in each experiment was much larger in magnitude than the remaining terms in the VIB or MASS training loss, and the smallest β value used was much smaller than the remaining terms. We made this choice in the hope of clearly observing the effect of the β parameter and more fairly comparing SoftmaxCE, VIB, and MASS. But we note that a finer-tuning of the β parameter would likely result in better performance for both VIB and MASS. We also note that the reason we omit a $\beta = 0$ run for VIB with the SmallMLP network was that we could not prevent training from failing due to numerical instability with $\beta = 0$ with this network.