Understanding information leakage of distributed inference with deep neural networks: overview of information theoretic approach and initial results

Tiffany Tuor, Shiqiang Wang, Kin K. Leung, Bong Jun Ko

Tiffany Tuor, Shiqiang Wang, Kin K. Leung, Bong Jun Ko, "Understanding information leakage of distributed inference with deep neural networks: overview of information theoretic approach and initial results," Proc. SPIE 10635, Ground/Air Multisensor Interoperability, Integration, and Networking for Persistent ISR IX, 106350K (4 May 2018); doi: 10.1117/12.2306000

SPIE.

Event: SPIE Defense + Security, 2018, Orlando, Florida, United States

Understanding information leakage of distributed inference with deep neural networks: Overview of information theoretic approach and initial results

Tiffany Tuor[a], Shiqiang Wang[b], Kin K. Leung[a], and Bong Jun Ko[b]

[a]Imperial College London, UK
[b]IBM T. J. Watson Research Center, Yorktown Heights, NY, USA.

ABSTRACT

With the emergence of Internet of Things (IoT) and edge computing applications, data is often generated by sensors and end users at the network edge, and decisions are made using these collected data. Edge devices often require cloud services in order to perform intensive inference tasks. Consequently, the inference of deep neural network (DNN) model is often partitioned between the edge and the cloud. In this case, the edge device performs inference up to an intermediate layer of the DNN, and offloads the output features to the cloud for the inference of the remaining of the network. Partitioning a DNN can help to improve energy efficiency but also rises some privacy concerns. The cloud platform can recover part of the raw data using intermediate results of the inference task. Recently, studies have also quantified an information theoretic trade-off between compression and prediction in DNNs. In this paper, we conduct a simple experiment to understand to which extent is it possible to reconstruct the raw data given the output of an intermediate layer, in other words, to which extent do we leak private information when sending the output of an intermediate layer to the cloud. We also present an overview of mutual-information based studies of DNN, to help understand information leakage and some potential ways to make distributed inference more secure.

Keywords: Cloud/edge computing, deep neural networks, distributed inference, Internet of Things, information leakage, mutual information

1. INTRODUCTION

Internet of Things (IoT) devices and edge computing applications generate a myriad of data at the network edge. During inference task, decisions are made using these collected data. In some cases, input edge devices are only used for sensing data and all the data collected at the edge are transmitted to the cloud, and the inference is performed entirely on the cloud. This approach has the benefit to alleviate the computational demand for edge devices. However, it requires intensive transmission between the edge and the cloud and raises some privacy issues as one may not want to send his/her raw data directly to the cloud. In other cases, inference is performed entirely on the end (edge) devices and decisions are made directly at the edge. However, edge devices usually have limited memory and battery budgets which are often not sufficient to satisfy the large resource demand of a deep neural network[1] (DNN) inference task.

A good compromise between these two options is to partition the inference task of a DNN between the edge and the cloud*. The edge device performs inference up to an intermediate layer of the DNN, and offloads the output features to the cloud for the inference of the remaining of the DNN (See Figure 1). Partitioning a DNN can improve energy efficiency. However, the distributed nature raises some concerns regarding communications cost and privacy issues. Although raw data is not sent directly to the cloud, the cloud may be able to reconstruct part of the raw data using the output of an intermediate DNN layer. To prevent private information leakage, one would like to find a compressed representation of the input data while retaining as much information as possible about the target output. Recently, studies have focused on an information theoretic view of DNN, where a trade-off between compression and prediction has been investigated. In this view, each layer is quantified by

*We assume a system can train a single end-to-end DNN model, and partition it between end devices and the cloud. In other words, the training part can be done in a centralized place and both the edge device and the cloud have the trained model.

Ground/Air Multisensor Interoperability, Integration, and Networking for Persistent ISR IX, edited by Michael A. Kolodny, Dietrich M. Wiegmann, Tien Pham, Proc. of SPIE Vol. 10635, 106350K · © 2018 SPIE · CCC code: 0277-786X/18/$18 · doi: 10.1117/12.2306000

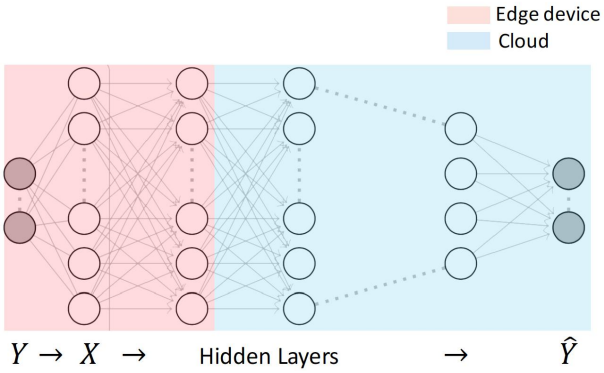


Figure 1: Distributed Inference

the amount of information it contains on the input variable and on the desired output. Understanding the mutual-information at different layers of the DNN can help us understand information leakage and potentially make distributed inference more secure.

In this paper, we give an overview of existing work on the information theoretic view of DNNs and discuss the privacy leakage of distributed inference. In Section 2, we start with a simple experiment to understand to which extent is it possible to reconstruct raw data given the outputs at different intermediate layers. In Section 3, we summarize some basic concepts of information theory. In Section 4, we present the information theoretic view of DNN and summarize the related work. Finally, in Section 5, we discuss how mutual-information based studies of DNN can help understand information leakage and make distributed inference more secure.

2. INITIAL EXPERIMENT: RECOVERING RAW DATA

We start with a simple experiment that simulates a distributed DNN inference task and evaluate how well the cloud can recover the raw data using the output of an intermediate layer. This experiment is similar to[2] but we consider a simpler dataset here for illustration.

The experiment asks the following question: given a representation of the original image (i.e., the output of a layer of the DNN when the original image is the input of the DNN), how well can we reconstruct the original image? Our goal is to approximate the inverse of an image representation, in other words we want to find the image whose representation best coincides with the given one.

Formally, we denote the representation of the original image x obtained at the output of layer i by $b_i(x)$, where we consider the 0-th layer as the input layer, thus $b_0 = x$. The problem is formulated as follows. Given the following:

1. A representation function $b_i : \mathbb{R}^{d_0} \rightarrow \mathbb{R}^{d_i}$ for each layer i, where d_0 is the dimension of input layer 0 and d_i is the dimension of the output at layer i (this representation function is specified directly by the trained DNN model)
2. An input image x_0
3. A representation $b_i(x_0)$ at a specific layer i

The goal is to find the input image x so that the error between $b_i(x)$ and $b_i(x_0)$ is the smallest:

$$x^* = \arg\min_{x \in \mathbb{R}^{d_0}} \|b_i(x) - b_i(x_0)\|^2 \tag{1}$$

The image x^* found from (1) should visually look similar as x_0.

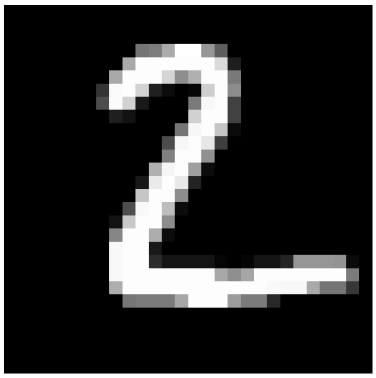
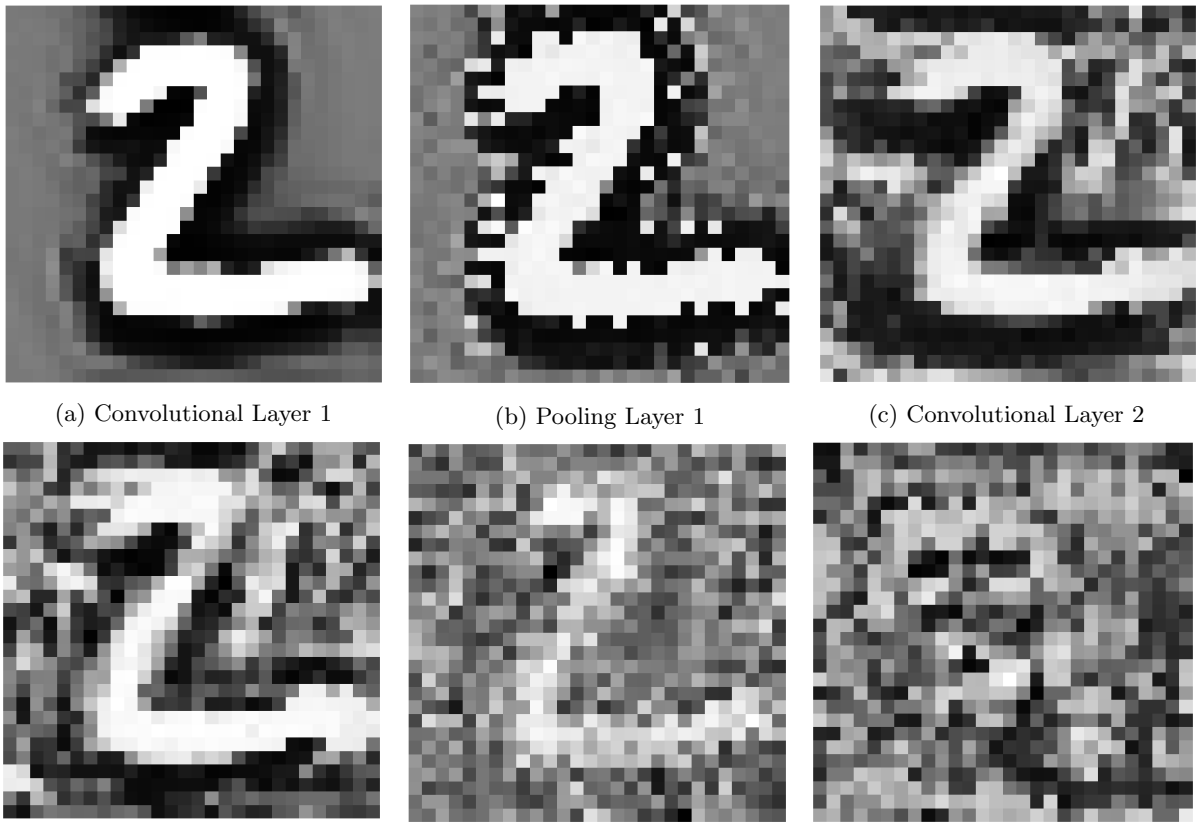


Figure 2: Original Input



(a) Convolutional Layer 1 (b) Pooling Layer 1 (c) Convolutional Layer 2

(d) Pooling Layer 2 (e) Dense Layer 1 (f) Dense Layer 2 (Output Layer)

Figure 3: Reconstruction from Different Layers of the DNN

We performed an experiment on MNIST dataset,[3] which contains 60,000 training examples and 10,000 test examples of handwritten digits (0-9). In order to obtain the representation function, we trained a DNN with the following architecture[†]:

[†]This architecture is used in the Tensorflow tutorial : https://www.tensorflow.org/tutorials/layers. The original architecture has a drop out in the Dense Layer #1 but we removed it. For this small convolutional neural network, the performance is actually nearly the same with and without dropout.

- Convolutional Layer 1: 32 of 5×5 filters with ReLU activation function
- Pooling Layer 1: Max pooling with a 2×2 filter and a stride of 2
- Convolutional Layer 2: 64 of 5×5 filters with ReLU activation function
- Pooling Layer 2: Max pooling with a 2×2 filter and a stride of 2
- Dense Layer 1: 1024 neurons with ReLU activation function
- Dense Layer 2 (Output Layer): 10 with Softmax activation function, one neuron for each digit class

Note that although this architecture is not the state-of-the art, it can get around 99.2% testing accuracy.

Figure 2 shows the original input we attempt to reconstruct and Figure 3 shows reconstructions obtained from the representations obtained at different layers (i.e., $b_i(x_0)$ at different i).

Our experiment confirms that we leak information about raw data when performing distributed inference. Most of the layers seem to retain lots of information about the raw image. We manage to reconstruct visually similar image as the original input until the penultimate layer. From Figure 3, we can see that the reconstruction error is larger when reconstructing from a deeper representation. We also note that the reconstructed image retains the characteristics of this specific handwriting of the digit "2".

3. INFORMATION THEORY BACKGROUND

3.1 Entropy

Formally, information entropy[4] measures how unpredictable the outcome of a stochastic source is. It is defined as:

$$H(X) = -\sum_{i=1}^{n} p(x_i) \log_2 p(x_i)$$

where we define $p(x) = \Pr\{X = x\}$ and use similar conventions later in the paper.

3.2 Conditional Entropy

The conditional entropy quantifies the amount of information required to describe the outcome of a random variable X given the value another random variable Y. The entropy of X conditioned on Y is written as $H(X|Y)$

$$H(X|Y) = \sum_{y \in Y, x \in X} p(x,y) \log \left(\frac{p(y)}{p(x,y)} \right) = H(X,Y) - H(Y)$$

where, with a slight abuse of notation, we use $x \in X$ to denote that x is a possible outcome (with non-zero probability) of the random variable X. Note that $H(X|Y) = 0$ if and only if X is completely determined by the value of Y. Conversely, $H(X|Y) = H(X)$ if and only if X and Y are independent random variables.

3.3 Mutual Information

Intuitively, the mutual information characterizes given one random variable X, how much new information is needed on average to completely describe Y. The mutual information between X and Y with joint distribution $p(x,y)$ is defined as:

$$\begin{aligned} I(X,Y) &= \sum_{x \in X, y \in Y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)} \\ &= H(Y) - H(Y|X) \end{aligned}$$

For continuous random variables, the mutual information is defined as:

$$I(X,Y) = \int_Y \int_X p(x,y) \log \frac{p(x,y)}{p(x)p(y)} dxdy$$

Note that $I(X,Y) = 0$ if and only if X and Y are independent random variables. Further, we always have $I(X,Y) \geq 0$ and $I(X,Y) = I(Y,X)$. Let $X \to Y \to Z$ denote a Markov chain of variables X, Y, and Z. An important property of mutual information is the data processing inequality: If $X \to Y \to Z$, then $I(X,Y) \geq I(X,Z)$, i.e., processing Y cannot add new information about X. The mutual information is invariant to invertible re-parametrization. This means that for two invertible functions ϕ and ψ, we have $I(X,Y) = I(\phi(X), \psi(Y))$.

3.4 The Problem of Information Bottleneck (IB)

The information bottleneck (IB) method[5] extracts relevant information that an input random variable X contains about an output random variable Y, assuming that X and Y are dependent and $p(x,y)$ is their joint distribution. The relevant information is defined as the mutual information $I(X,Y)$. The IB method finds the optimal trade-off between the compression of X and the prediction of Y.

If $b \in B$ is the compressed representation of $x \in X$, then the representation of x is defined by the mapping $p(b|x)$. Assuming the Markov Chain $Y \to X \to B$, the IB can be formulated as the following optimization problem

$$\min_{p(b|x),p(y|b),p(b)} \beta I(X;B) - I(Y;B) \tag{2}$$

where β handles the trade-off between $I(X;B)$, the complexity of the representation, and $I(Y;B)$, the amount of preserved information of the target output. When β is large, maximal compression of X is favored. When β is small, IB favors solutions where B retains maximum information about Y. An implicit solution to this problem has been derived in existing work.[5]

4. AN INFORMATION THEORETIC VIEW OF DNN

It has been observed[6] that a DNN can be seen as a Markov chain[‡]. As it is shown in Figure 4, the DNN is composed of successive layers, where each layer can be viewed as a random variable and is a representation of the input. The data in a DNN progresses in a similar way as a Markov chain, i.e., each layer can be calculated only from its previous layer. As a consequence of the data processing inequality, information about Y that is not captured in one layer cannot be recaptured in successive layers. Hence, for $i \geq j$:

$$I(Y;X) \leq I(Y;B_j) \leq I(Y;B_i) \leq I(Y;\hat{Y})$$

The amount of relevant information retained by the network is quantified by $\frac{I(Y;\hat{Y})}{I(X;Y)}$.

As mentioned in Section 3.4, IB is a technique that allows to extract feature in some high-dimensional input that are relevant for predicting some output variable.

4.1 Reformulated Goal of DNN

Many works[6–10] use this concept to study DNN where the goal of DNN is reformulated as an information theoretic trade-off between compression and prediction. IB also is used to understand the DNN training process.[6–8]

[‡]A Markov chain is a particular case of stochastic process, in which future state depends only on the present state. This property is called the Markovian property or memoryless property. In other words, the probability of transitioning to any other state depends only on the current state.

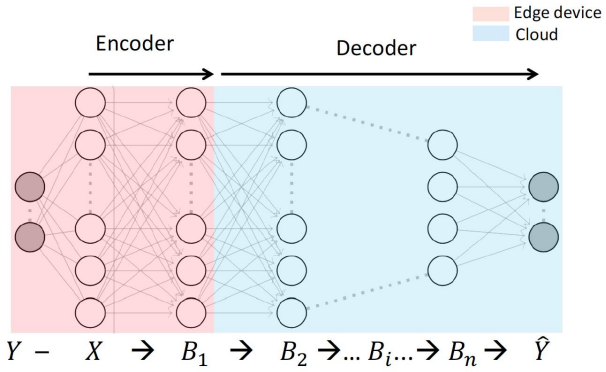


Figure 4: DNN as an Encoding and Decoding Pipeline

If we denote the representation of the input at a particular layer of the DNN as B (the "bottleneck variable"), B can be obtained by a (stochastic) encoder that maps the input X to B. The remaining part of the DNN forms a (stochastic) decoder that maps B to $\hat{Y}$, where $\hat{Y}$ is the predicted output variable (see Figure 4). The encoder/decoder representation allows us to quantify B by the amount of information it captures on the input variable and on the desired output, as well as on the predicted output of the DNN. We aim to find and encoding B that is maximally expressive about Y (i.e., that selects the important bits in order to accurately predict Y) as well as maximally compressive about X (i.e., that throws away bits that are not relevant for predicting Y). Hence, the optimal encoder is selected by finding the encoding that optimizes the IB defined in (2).

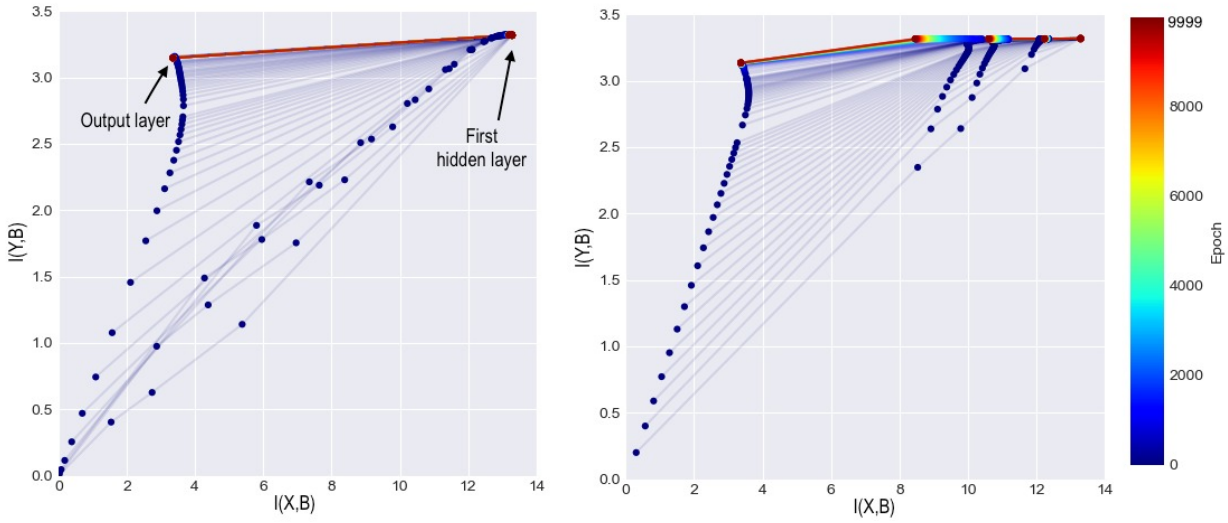
4.2 Understanding DNN in the Information Plane – An Ongoing Debate

A new visualization of DNN, the *Information Plane*, which shows the values of $I(X;B)$ and $I(Y;B)$ in a graph for each representation B, has been introduced recently.[6] On the horizontal axis, the information plane shows the mutual information between each layer and the input variable; on the vertical axis, it shows the mutual information between each layer and the desired output. The authors then extended this result and demonstrate that visualizing DNN in the information plane has some benefits.[7] More explicitly, the Information Plane gives insight on the training dynamics and learning processes of DNN. It is shown that stochastic gradient descent (SGD) optimization (the optimization technique used for training many DNN models) has two distinct phases: a fitting phase, in which the empirical error is minimized and a compression phase, in which the representation is compressed for better generalization. These two phases are visible in snapshots of the information plane at different epochs of the training. It is believed that the excellent generalization performance of deep networks is due to the compression phase.

More recently, it has been argued that the two (fitting and compression) phases are a consequence of the non-linearity of activation functions in the DNN.[8] It is claimed that the compression phase is due to the double-sided saturating non-linearities of the tanh activation function and does not appear with other linear activation functions and single-sided saturating non-linearities such as ReLU. It is claimed that DNNs still generalize well even without a noticeable compression phase. A similar observation has also been made earlier.[11] Both works[7,8] claim to estimate the mutual information using a common approach based on variational bounds.[9] In Figure 5, we replicate the result using the code supplied by the authors[8] in https://github.com/artemyk/ibsgd/tree/iclr2018. Figure 5 illustrates information plane dynamics of a network of size $784-1024-20-20-20-10$, trained using MNIST dataset and using their default estimator.[9] Figure 5a shows the information plane using a ReLU network, and Figure 5b shows information plane of a tanh network. For ReLU networks, compression is not observed expect for the final output layer.[8] However for tanh networks, a compression is observable at most of the layers.[7]

The debate between has not ended[§]. Some uncertainty remains surrounding the results where mutual information is estimated empirically. It can be very challenging to evaluate the mutual information in the IB

[§]https://openreview.net/forum?id=ry_WPG-A-



(a) ReLU (b) tanh

Figure 5: Information Plane Visualization

objective. This is why analytical solutions to IB has been obtained only for two limited cases: (1) when the random variables have a small number of discrete outcomes and the conditional probability distribution $p(b|x)$ can be explicitly represented for all entries;[5] (2) X and Y are continuous random variables which are jointly Gaussian distributed.[12] There exist other works that propose methods for approximating IB in much more general settings (for continuous non-Gaussian random variable),[9,10,13,14] which are based on variational bounds and is used in both of the works summarized above.

5. DISTRIBUTED AND PRIVACY ASPECTS

As shown in Section 2, it is possible to reconstruct part of the raw data, given the output extracted at a layer of the DNN. This shows some leakage of raw data when partitioning the inference task. In Table 1, we compute the mutual information $I(X;B_i)$ and $I(Y;B_i)$, where B_i is the output of layer i of the trained DNN model used in Section 2. The estimation of mutual information was performed using the variational approximation approach.[9] The results obtained in Table 1 are consistent with the results obtained in Figure 3. Indeed, the only level of reconstruction that does not allow us to reconstruct a visually similar image as the original input is the Dense Layer 2, and it is the only layer where $I(X;B_i)$ is very small (see Table 1). It is worth noting that a ReLU activation function was used in the experiment in Section 2 and the results presented in Table 1, which aligns with the results shown in Figure 5a where no compression is observed except for the last output layer.

Table 1: Estimated Mutual Information

i	$I(X;B_i)$ (bits)	$I(Y;B_i)$ (bits)
Convolutional Layer 1	13.3958	3.4120
Pooling Layer 1	13.3513	3.3750
Convolutional Layer 2	13.3207	3.3514
Pooling Layer 2	13.3071	3.3387
Dense Layer 1	13.2880	3.3197
Dense Layer 2	3.3462	3.1978

In order to avoid visual reconstruction from representation obtained at layer B_i, a small $I(X;B_i)$ is desirable. A small $I(X;B_i)$ also means a better compression which may reduce the communication overhead for transmitting the intermediate layer from the edge to the cloud.[9] Using IB as a regularization function during DNN training may help diminishing $I(X;B_i)$ while keeping a high value of $I(Y;B_i)$.

Training a DNN with IB objective have already shown advantages. It has been shown that models trained with the IB objective outperform those that are trained with other forms of regularization in terms of generalization performance and robustness to adversarial attack.[10]

6. CONCLUSION

In this paper, we have investigated the information leakage of performing distributed inference using DNNs. Through initial experiments, we have found that the mutual information between the input and the output at an intermediate DNN layer is related to the reconstruction error. We have also provided an overview of some other techniques related to the information theoretic understanding of DNNs.

ACKNOWLEDGMENTS

This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence under Agreement Number W911NF-16-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

REFERENCES

[1] Goodfellow, I., Bengio, Y., and Courville, A., [*Deep Learning*], MIT Press (2016). http://www.deeplearningbook.org.

[2] Mahendran, Aravindh, Vedaldi, and Andrea, "Understanding deep image representations by inverting them," in [*CVPR*], (2015).

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

[4] Cover, T. M. and Thomas, J. A., [*Elements of information theory*], John Wiley & Sons (2012).

[5] Tishby, N., Pereira, F., and Bialek, W., "The information bottleneck method," in [*37th Annual Allerton Conference on Communication, Control and Computing*], (1999).

[6] Tishby, N. and Zaslavsky, N., "Deep learning and the information bottleneck principle," in [*Information Theory Workshop (ITW)*], (2015).

[7] Schwartz-Ziv, R. and Tishby, N., "Opening the black box of deep neural networks via information," in [*arXiv preprint arXiv:1703.00810*], (2017).

[8] Saxe, A., Bansal, Y., J.Dapello, Advan, M., Kolchinsky, A., Tracey, B., and Cox, D., "On the information bottleneck theory of deep learning," in [*ICLR*], (2018).

[9] Kolchinsky, A., Tracey, B., and Wolpert, D., "Nonlinear information bottleneck," in [*arXiv preprint arXiv:1705.02436*], (2017).

[10] Alemi, A. A., Fisher, I., Dillon, J., and Murphy, K., "Deep variational information bottleneck," in [*ICLR*], (2017).

[11] Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O., "Understanding deep learning requires rethinking generalization," in [*ICLR*], (2017).

[12] Chechik, G., Globerson, A., Tishby, N., and Weiss, Y., "Information bottleneck for gaussian variables," *Journal of Machine Learning Research* **6**, 165–188 (2005).

[13] Chalk, M., Marre, O., and Tkacik, G., "Relevant sparse codes with variational information bottleneck," in [*Advances in Neural Information Processing Systems*], (2016).

[14] Achille, A. and Soatto, S., "Information dropout: learning optimal representations through noise," in [*arXiv preprint arXiv:1611.01353*], (2016).