CS 875 Information theory, Pattern recognition, and Neural networks

Information Theory for Deep Learning

Arjun Verma IMT2017008
Bishal Pandia IMT2017010

IIIT Bangalore

06/12/2021

Outline

Introduction

Motivation

The inner-workings of deep learning models has been a puzzling problem for theoreticians ever since its advent. With the large amounts of parameters they have, the fact that they seem to generalize well is a confounding one. Moreover, each model has a number of parameters and hyperparameters to tune and the major guide for setting them up ends up being our intuition. There is thus a highly motivating need for a theory that not only serves as an X-ray to look into the blackbox of neural networks but could also be a guide to practitioners.

Need for a Theory

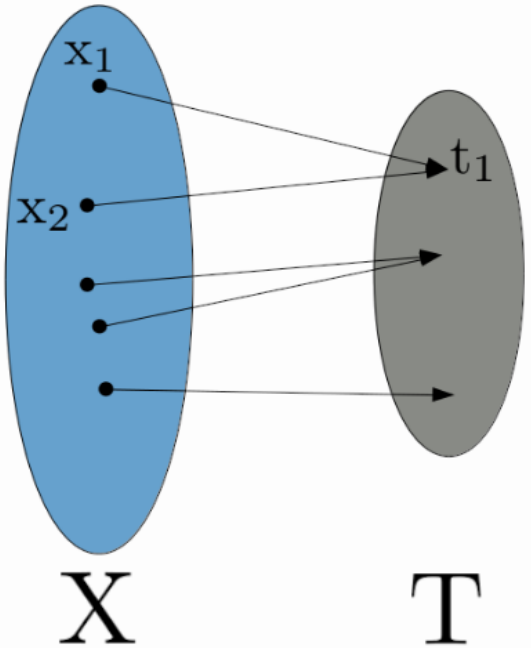
1. Why do DNN's work so well?
2. How can they be improved?
   - Optimality Bounds
     Are DNN's the best we can do?
   - Design Principles
     What determines the number and width of layers?
   - Interpretability
     What do the layers and neurons represent?
   - Better Learning Algorithms
     Is stochastic gradient descent the best we can do?

What's the connection?

- Rather than jump right into connection between the two it actually helps to first understand the Information Bottleneck (IB) principle. As we will see, once we understand IB, the connection between IT and DL arises quite naturally.
- The main purpose of IB is to answer the following question: "How do we define what information is **relevant**, in a rigorous way?" This seems rather counter-intuitive, as the 'relevancy' of information seems rather subjective. Information theory as laid out by Claude Shannon omits any notion of 'meaning' in the information, and without meaning, how can we possibly measure relevancy?
- Information theory does hold one possible answer to the question of relevancy however, in the form of lossy compression and Rate Distortion Theory helps us to formailze these concepts.

Rate Distortion Theory

Defining Compression : We can formally think of compression as defining a (possibly stochastic) function that maps an element $x \in X$ to its corresponding compressed representation. We will refer to the set of codewords of X as T, with an arbitrary element of T being denoted as t. This map is implicitly defined through the probability distribution $p(t|x)$, so by defining the distribution $p(t|x)$ we will have defined our method of compression/assignment of codewords.



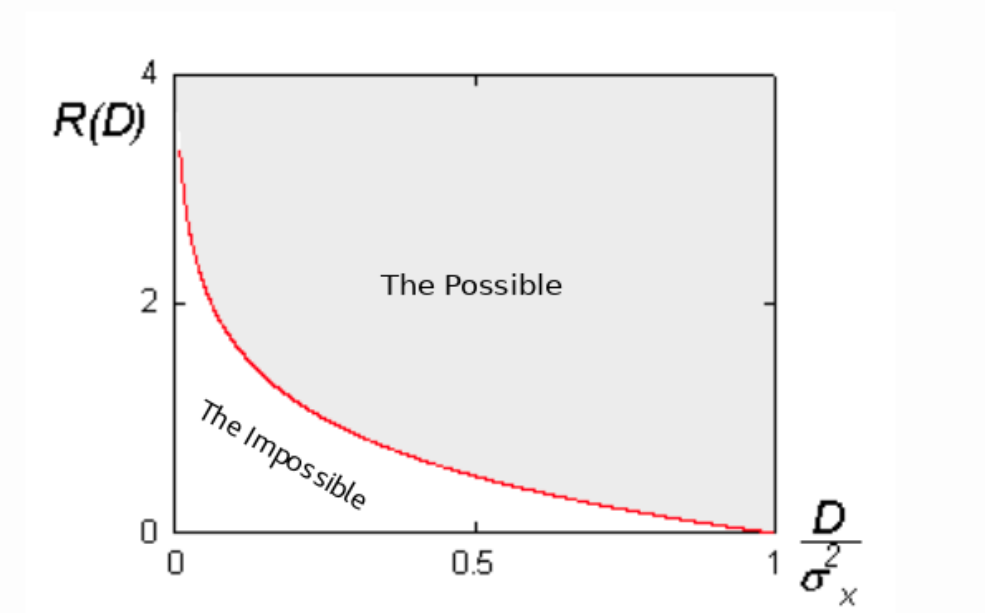
Rate Distortion Theory

- **Measuring Compression** We now need a good mathematical way to measure how 'compressed' our representation (decided by $p(t|x)$) is. $I(X, T)$ serves as this quantity for us.
- **Losing Information** One more thing we need to formalize: we need to define how much information loss is too much information loss, and more particularly, what constitutes as information loss. The expected distortion $D(p) = \Sigma_{x,t} p(x, t) d(x, t)$ serves as this quantity.
- **Rate Distortion Curves** With all this in place we can now answer the golden question: What's the most we can possibly compress an input (equivalently, how low can we get the rate of T) if we are allowed to distort the input up to threshold D$*$?

$$R(D*) = min_{p(t|x):D(p)\leq D*} I(X; T)$$

Rate Distortion Curve

Note that for each value of D$*$ we get a different value of best possible rate, R. We can plot R as a function of D$*$ to get what is called the rate distortion curve. This curve represents the optimal values of R, any values above the curve are sub optimal (meaning the representations T are sub optimal), any values below the curve are theoretically impossible to achieve.



The Information Bottleneck Method

- **Problems with RDT** We have to provide a distortion measure, and because of this, Rate Distortion Theory doesn't serve our mission of defining information relevancy in a rigorous way.
- **The IB Method** The information bottleneck principle (IB) is quite similar to RDT, with some key differences. Instead of defining relevance directly through the set of codewords T, IB instead defines relevance through another variable Y.

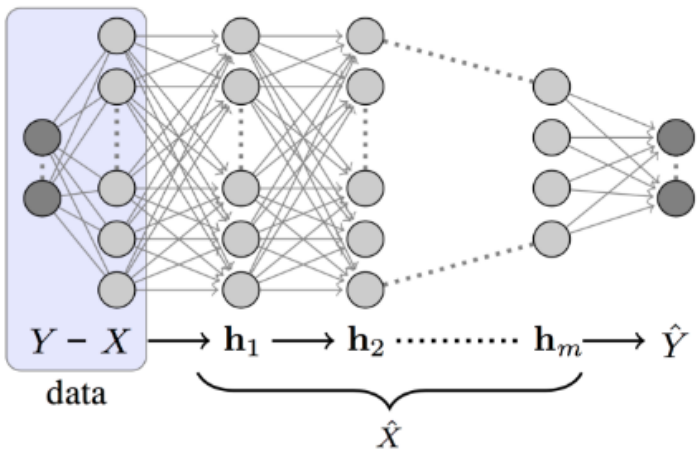
$$R(I*) = min_{I(T|Y) \geq I*} I(X;T)$$

In order to get a good representation T, we must 'squeeze out' any information in X that is irrelevant to Y, leaving us only the parts of X relevant to Y. This is the bottleneck in the information bottleneck principle.

Deep Learning and the Information Bottleneck Method

- As you might have guessed, IB sort of 'smells' like deep learning. In particular, you may have noticed these following analogs: X=Inputs, T=Hidden Layers, Y=Outputs.
- If we label the hidden layers of a DNN as h1,h2,. . . ,hm as shown below, we can view each layer as one state of a Markov Chain: hi→ hi+1. According to DPI, we would have:

$$I(X;Y) \geq I(h_1;Y) \geq I(h_2;Y) \geq ... \geq I(h_m;Y) \geq I(\hat{Y};Y)$$



Deep Learning and the Information Bottleneck Method

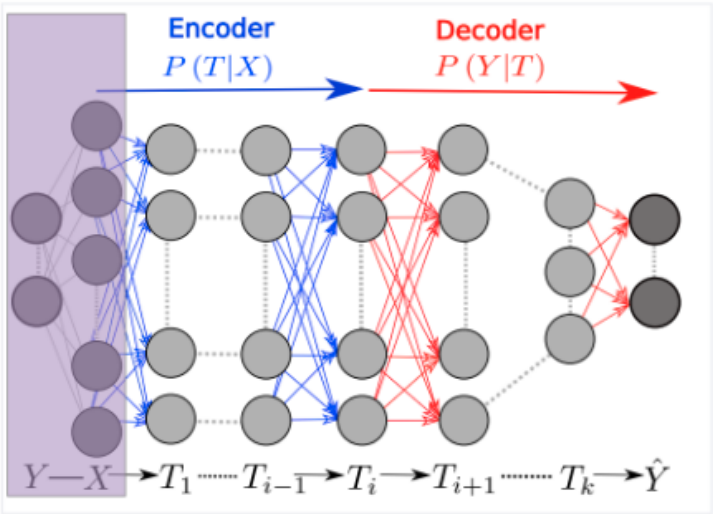
What do the DNN layers represent?

- A Markov chain of topologically distinct [soft] partitions of the input variable X.
- Successive Refinement of Relevant Information. A DNN is designed to learn how to describe X to predict Y and eventually, to compress X to only hold the information related to Y. Tishby describes this processing as "successive refinement of relevant information".
- Individual neurons can be easily "scrambled" within each layer

Information Plane Theorem

Theorem

For large typical X, the sample complexity of a DNN is completely determined by the encoder mutual information, I(X;T), of the last hidden layer; the accuracy (generalization error) is determined by the decoder information, I(T;Y), of the last hidden layer.



The complexity of the problem shifts from the decoder to the encoder, across the layers.

Two Optimization Phases

On plotting hidden layers on the information plane, we observe two phases during the optimization process using SGD, namely the **drift** and the **diffusion** phase. Let us see this process in action.

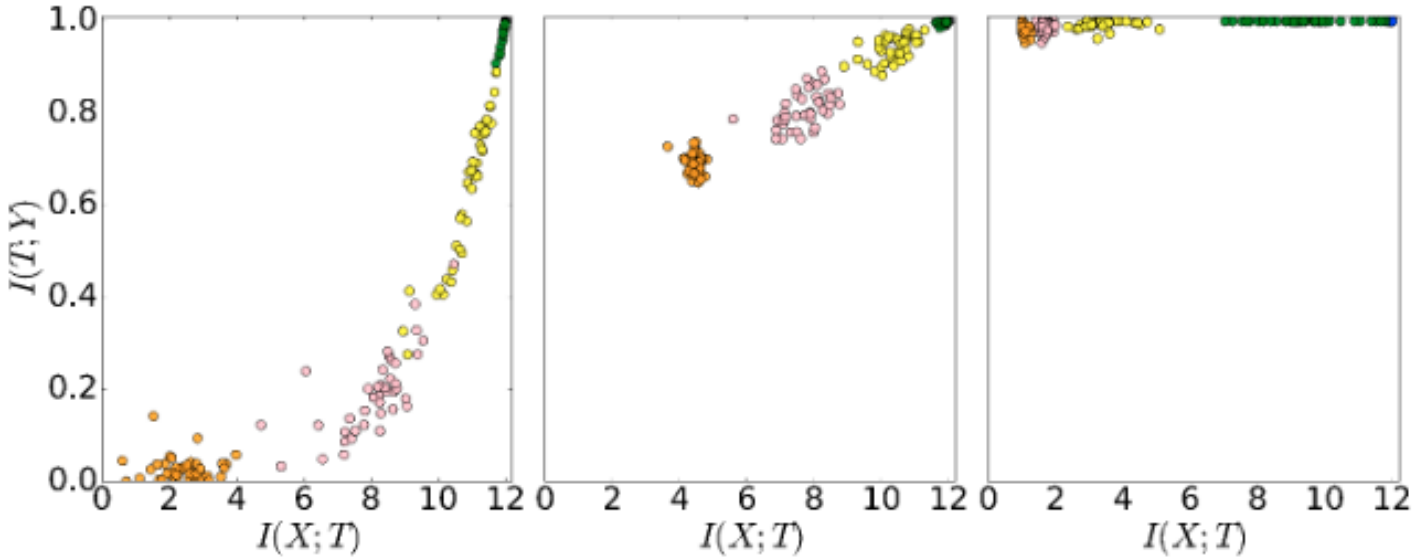


Figure 2: A snapshot of the layers of the network on the information plane during SGD optimization (left - initial weights, mid - after 400 epochs, right - after 9000 epochs). The different points correspond to 50 different weight initializations.

Two Optimization Phases

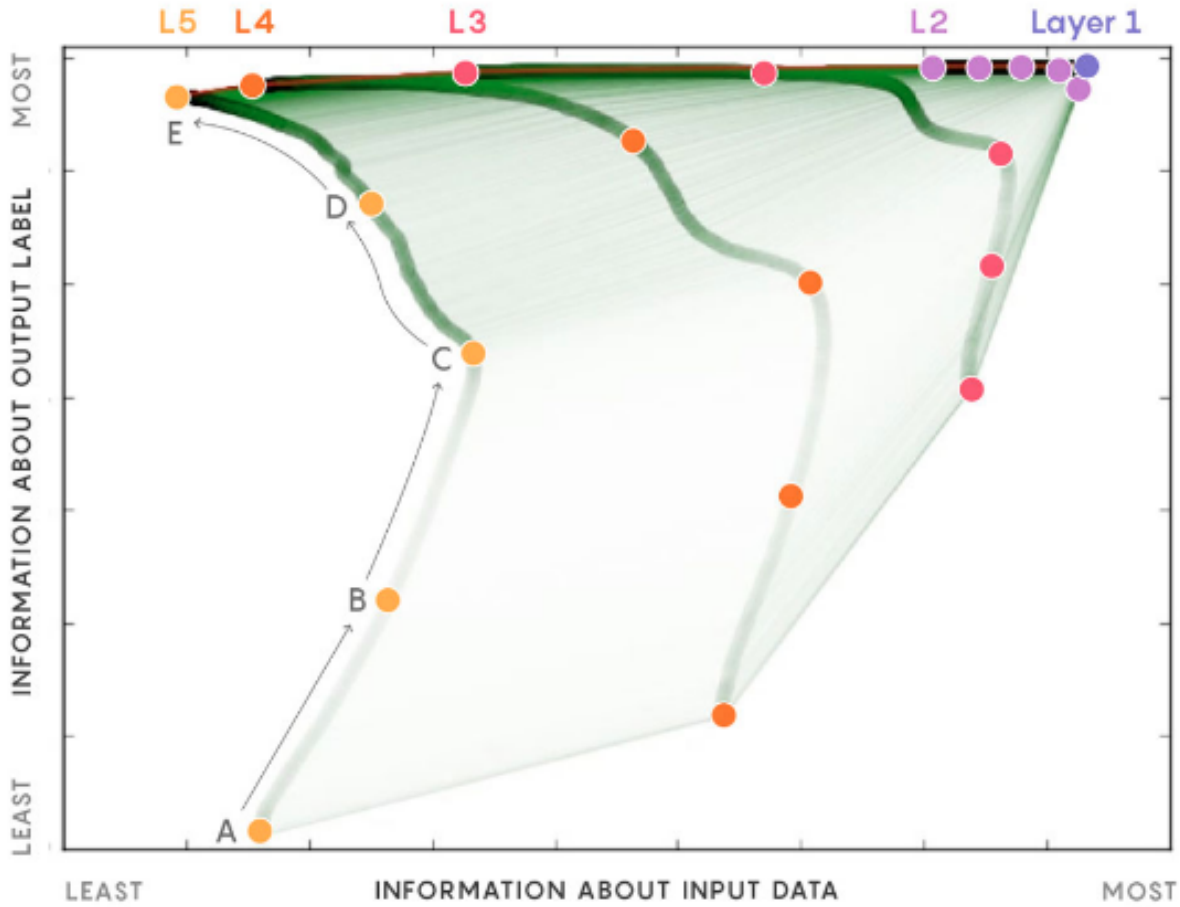


Figure 3: The average trajectory of the hidden layers during the optimization process on the information plane.

Two Optimization Phases

Tishby et al. explain the phases as follows :

- The **drift** phase is explained by the cross-entropy loss minimization during which the layers increase the information on the labels as well as the input while preserving the Data Processing Inequality order (lower layers have higher information).
- The **diffusion** phase is a little mysterious as the compression that can be seen taking place is not because of any explicit regularization or a related technique. This phase is a lot slower as compared to the first phase and the layers lose irrelevant information during this phase until convergence.

Effect of different number of training examples

Less number of training examples lead to convergence very far from the convergence with more number of examples in the data. The label information is reduced in the case with less number of examples during the compression phase. In this case, we are overcompressing to the point that the representation is now too simple to capture the label information.

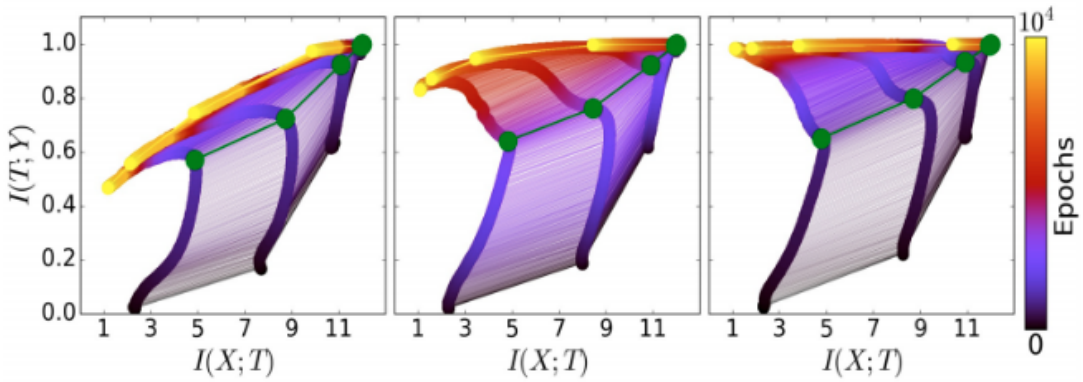


Figure 4: The evolution of the layers with the training epochs in the information plane, for different training samples. On the left - 5% of the data, middle - 45% of the data, and right - 85% of the data. The colors indicate the number of training epochs with Stochastic Gradient Descent from 0 to 10000 epochs.

Generalization Theory

Definition

The goal of generalization theory is to explain and justify why and how improving accuracy on a training set improves accuracy on a test set. The difference between these two accuracies is called **generalization error** or **generalization gap**

Generalization Upper Bounds

$$\epsilon^2 < \frac{log|H_\epsilon| + log(1/\delta)}{2m}$$

ϵ: Generalization error.

H_ϵ: ϵ-cover of the hypothesis class. Typically we assume the size $|H_\epsilon| \sim (\frac{1}{\epsilon})^d$

δ: Confidence.

m: The number of training examples.

d: The VC dimension of the hypothesis.

Useless Bound!!

- Plug in $|H_\epsilon| \sim (\frac{1}{\epsilon})^d$ into the equation above
- After simplification, a factor of $\frac{d}{m}$ appears as a dominant factor

Remarks

Generally in deep learning, we have dimensions in the order of the weights of the network which is very high as compared to the number of examples and therefore, there won't be any phase where generalization occurs. But this is not true!!

New Approach by Tishby & Zaslavsky

Approach

Don't estimate the upper bound based on what all functions can be approximated(hypothesis cardinality), rather look at reducing the complexity of the input size and then generate a more tighter bound.

Not a new strategy -

- K-nearest neighbours
- Nyquist sampling theorem

Input Compression Bound

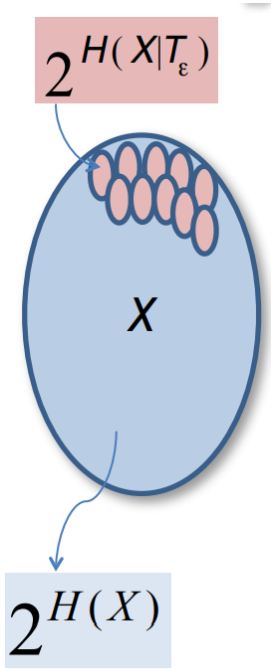
Let $X_1, X_2, X_3, ..., X_n$ be a sequence of random variables which are identically and independently distributed. Considering n to be very large, AEP(Asymptotic Equipartition Property) can be applied.

Definition of AEP

Let $p(X_1, X_2, X_3, ..., X_n)$ be the probability of observing $X_1, X_2, X_3, ..., X_n$, then $\frac{-1}{n} log_2 p(X_1, X_2, X_3, ..., X_n)$ is close to H(X). In other words, $p(X1, X2, ..., Xn)$ is close to $2^{-nH(X)}$

Input Compression Bound

For large partitions T, $p(X_1, X_2, X_3, ..., X_n|T) = 2^{-nH(X|T)}$
Transform : $|H_\epsilon| \sim 2^{|X|} \rightarrow 2^{|T_\epsilon|}$



No. of typical elements $= 2^{H(X)}$
Avg. Partition Size $= 2^{H(X|T_\epsilon)}$
Therefore,
No. of partitions $= \frac{2^{H(X)}}{2^{H(X|T_\epsilon)}} = 2^{H(X)-H(X|T_\epsilon)} = 2^{I(X;T_\epsilon)}$

Input Compression Bound

$$\epsilon^2 < \frac{2^{I(X;T_\epsilon)} + log(1/\delta)}{2m}$$

- $2^{I(X;T_\epsilon)}$ contains the compressed information of the partition

> *"K bits of compression of X are same as a factor of 2^k training samples"*

IB Limit

Consider a markov chain,

$$X \rightarrow S(X) \rightarrow \hat{X} \rightarrow \hat{Y}$$

$\hat{X}$: Best representation of input X to predict the most accurate output $\hat{Y}$
$\hat{Y}$: Best predicted output
$S(X)$: Internal representation of input X

IB Limit

IB Bound optimality equation:

$$\hat{X} = \underset{p(\hat{X}|X)}{\operatorname{argmin}}\, I(\hat{X}; X) - \beta I(\hat{X}; Y), \beta > 0$$

$p(X|\hat{X}) = \frac{p(X)}{Z(X,\beta)} exp(-\beta D[p(y|x)||p(y|\hat{x})])$
$Z(x, \beta) = \sum p(\hat{x}) exp(-\beta D[p(y|x)||p(y|\hat{x})])$
$p(\hat{x}) = \sum p(\hat{x}|x)p(x)$
$p(y|\hat{x}) = \sum p(y|x)p(y|\hat{x})$

Solvable by Arimoto-Blahut like Iterations

IB Limit

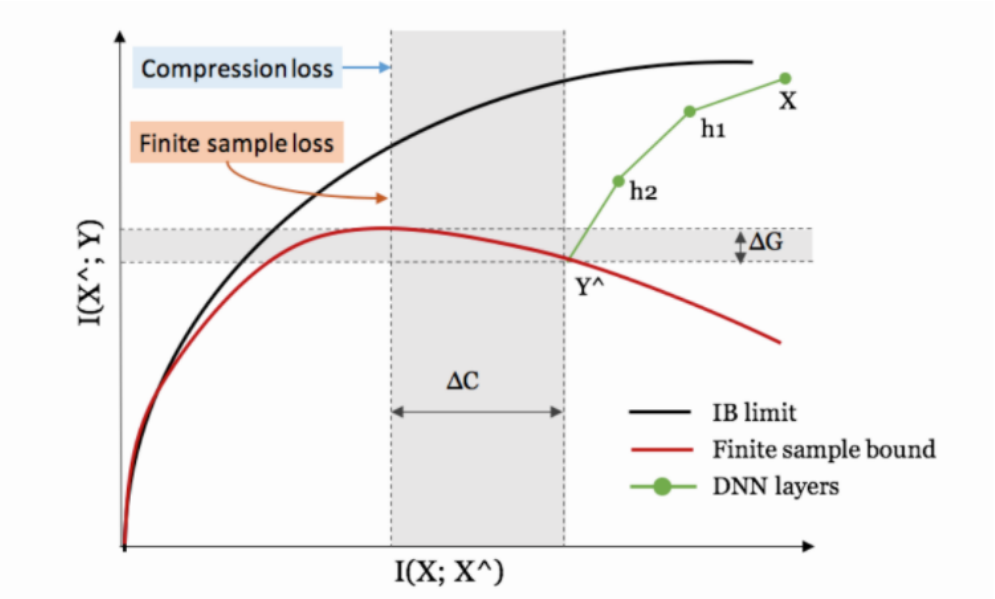


Figure 5: Black line curve depicting the optimal theoretical IB limit

But, we do not work with infinite amount of data. In reality, we work with a very small amount(though in millions) of finite samples. We require a different finite sample upper bound for comparison!!

Finite Sample Bound

For finite samples, there is an information loss which is formulated as follows:

$$I(\hat{X}; Y) \leq \hat{I}(\hat{X}; Y) + O(K|y|/\sqrt{n})$$

Remarks

This implies that there exists some finite sample losses which puts an upper bound on the theoretical IB Limit for finite samples, hence leading us to the red curve

Drift and Diffusion Phases

- The number of epochs that the training process takes in order to reach the green path boundary is very less when compared to the total number of epochs or the number of epochs needed in the compression phase.
- The training error reduces to a very small quantity at this point after which the diffusion phase starts in which the noise in the gradient dominates the training process. Tishby calls this phase as the **forgetting phase** and argues that it is the most important phase in the learning process as the model learns to forget.
- This can be bolstered by the fact that even though all the fitting of the data happens in the first phase, most of the information gain about the label happens in the second phase. Most of the irrelevant information about the patterns is discarded in this phase and this results in an overall compression.

Drift and Diffusion Phases

- The region of high ∇ - SNR phase to the left of the grey line marks the **memorization phase**.
- The region of low ∇ - SNR phase to the right of the grey line marks the **generalization phase**.

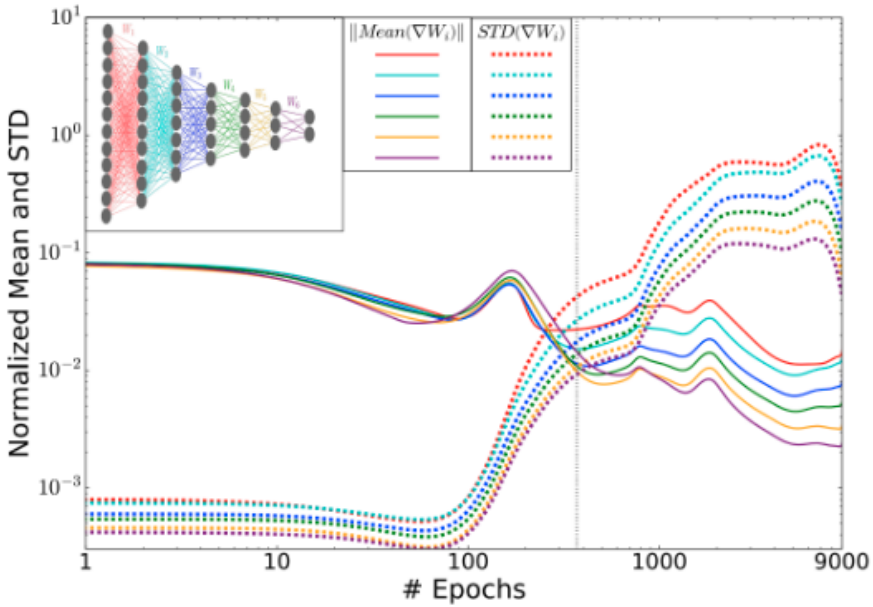


Figure 6: The mean and the standard deviation of the gradients during the SGD optimization process as a function of training epochs in log-log scale. The grey line marks the transition phase from drift to diffusion.

Benefit of Stochastic Relaxation

- Tishby talks about the role of noise induced by the mini-batches in the SGD process. He says that in a given problem, there are only a small number of relevant directions. All the other million dimensions are not relevant.
- This results in a covariance matrix of the noise which is quite small in the dimension of the relevant features and really elongated in the dimension of irrelevant features but this property is good and not bad.
- This is because during the drift phase, we quickly converge in the direction of the relevant dimensions and we stay there. But now we start doing random walks in the remaining millions of irrelevant dimensions and their total averaged effect increases the entropy in the irrelevant dimensions of the problem.

Benefit of Multiple Hidden Layers

- In general, the time that it takes to increase the entropy is exponential in the entropy which explains why the diffusion phase is so slow as compared to the drift phase. It thus takes exponential time for every bit of compression.
- What multiple hidden layers do for us is that they run the Markov chain using parallel diffusion process. Noise is added to all these layers and hence they start to push each other there is a boost in the time as they all help each other in compression. So there is an exponential improvement in running time as the layers help each other and each layer has to compress only from the point where the previous layer forgot.

Benefit of Multiple Hidden Layers

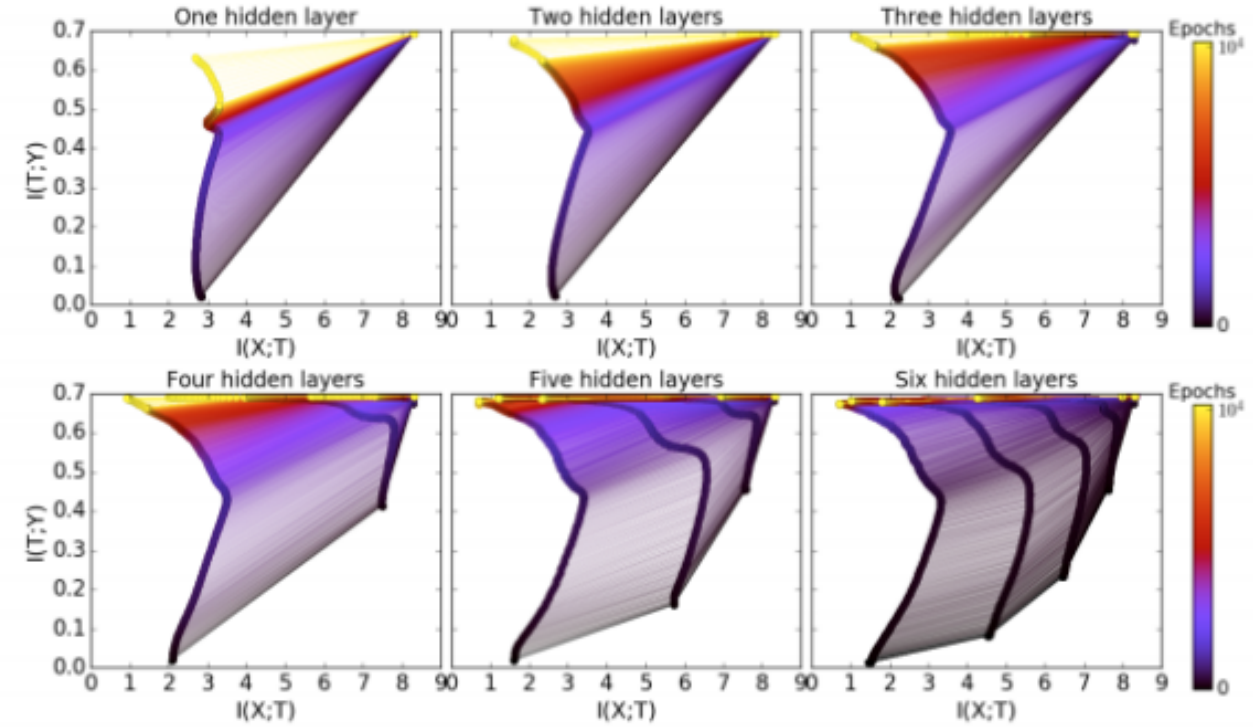


Figure 7: Variation in optimization (compression phase) with different number of hidden layers with the same number of epochs. The optimization time is shorter with more hidden layers. The width of the hidden layers start with 12, and each additional layer has 2 fewer neurons.

Benefit of Training Data

Now we know that it is the noise of the gradient which is doing most of the work for us in the compression phase and helping to forget the irrelevant information. Now comes the question of where the layers actually converge?

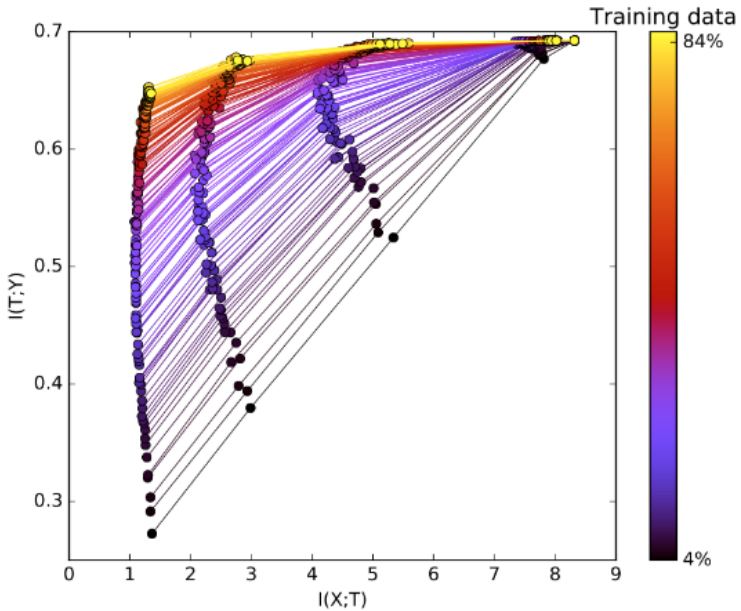


Figure 8: Convergence of hidden layers with different amounts of training data. The result is highly consistent with information plane theory. Each line (color) represents a converged network with a different training sample size.

Implications and Future Directions

- Exactly solvable DNN models through symmetry and group theory.
- New/Better Learning algorithms and design principles.
- Predictions on organizations of biological layered networks.

Ongoing Debate

- In the paper *"On the Information Bottleneck Theory of Deep Learning"*, Saxe, et.al tried to falsify the following three claims made by Naftali et.al in their paper *"Deep Learning and the Information Bottleneck Principle"* -
  1. Deep networks undergo two distinct phases consisting of an initial fitting phase and a subsequent compression phase
  2. The compression phase is causally related to the excellent generalization performance of deep networks
  3. The compression phase occurs due to the diffusion-like behavior of stochastic gradient descent

Ongoing Debate Contd.

- However, in the same year, a paper *"Scalable Mutual Information Estimation using Dependence Graphs"* by Noshad et.al was published which states that the results by Saxe et.al are invalid. Morever, this paper claims that the compression property of information bottleneck (IB) in fact holds for ReLu and other rectification functions in deep neural networks (DNN) using EDGE(Ensemble Dependency Graph Estimator) as an MI estimator.
- Another paper *"Estimating Information Flow in Deep Neural Networks"* was published by Ziv et.al which claims that the measured quantity $I(X;T)$ in the previous works is actually *Clusterization* which is very much similar to compression intuitively and besides the naming, the previous work of Naftali et.al are totally valid and accurate.

Learning Disentangled Representations

InfoGANs help us to generate disentangled representations in an unsupervised manner.

- Writing styles from digit shapes on MNIST.
- Pose from lighting of 3D rendered images.
- Background digits from central digit on SVHN
- Hair styles, presence/absence of eyeglasses and emotions on CelebA

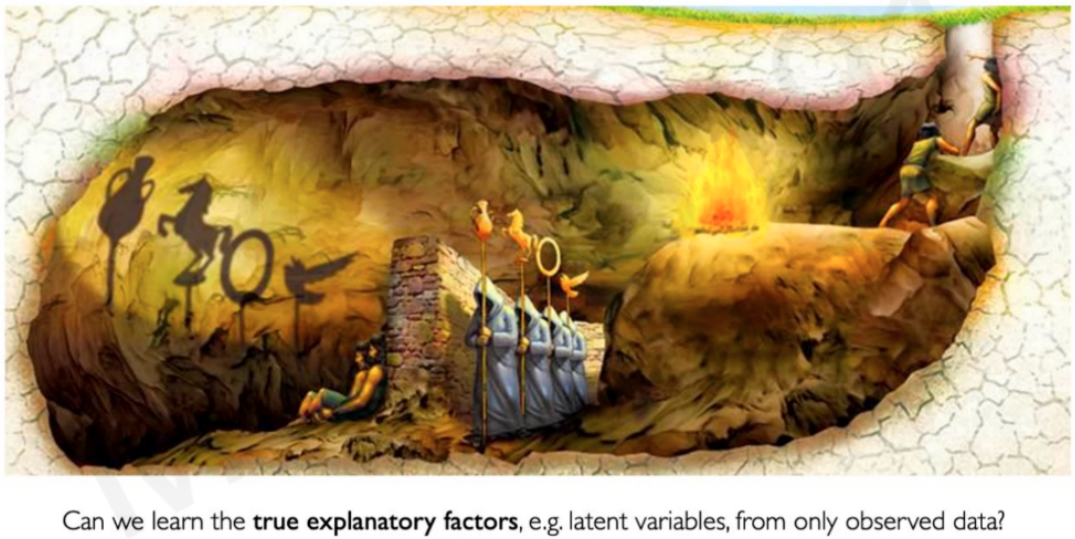


Figure 9: A depiction of Plato's allegory of the cave.

Optimization Equation

- **Regular GAN**

$$min_G max_D V(D, G) = E_{x \sim P_{data}}[logD(x)] + E_{z \sim noise}[log(1 - D(G(z)))]$$

- **InfoGAN**

$$min_G max_D V_1(D, G) = V(D, G) - \lambda I(c; G(z, c))$$

where, z $\rightarrow$ source of incompressible noise
c $= (c_1, c_2, c_3, ..., c_L) \rightarrow$ latent code (semantic features of the data)

Optimization Equation

Computing I(c; G(z, c)) requires access to the posterior $P(c|x)$
$Q(c|x) \rightarrow$ auxiliary distribution to approximate $P(c|x)$
$L_1(G, Q) = E_{c\sim P(c),x\sim G(z,c)}[logQ(c|x)] + H(c) \leq I(c; G(z, c))$
$L_1(G, Q) \rightarrow$ Variational Lower Bound

Final Optimization Function:

$$min_{G,Q} max_D V_{InfoGAN}(D, G, Q) = V(D, G) - \lambda L_1(G, Q))$$

Examples

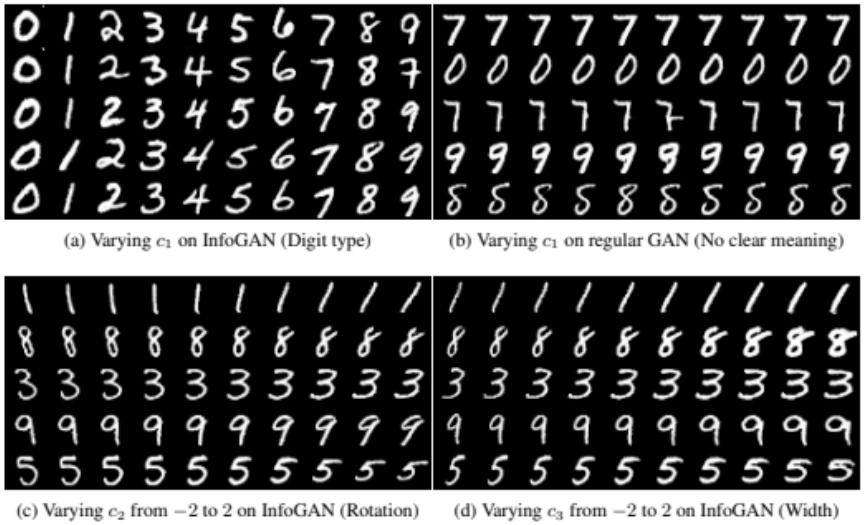


Figure 10: Manipulating Latent Codes on MNIST.

Figure 11: Manipulating Latent Codes on 3D Chairs.