Analyzing the hGRU model with the insights of the information bottleneck theory

Xinhao Fan[1,*], *Drew* Linsley[2,**], and *Thomas* Serre[2,***]

[1]Institute of Physics, Nankai University
[2]Department of Cognitive Linguistic & Psychological Sciences, Brown University

Abstract. Artificial neural networks have achieved great success in the field of computer vision. However, the deep neural network itself still remains a blackbox to us and there lacks an universal approach to judge the efficiency of the network structure. Besides that, the state-of-the-art computer vision algorithm struggles in some tasks which are easy to human. Our work applies a method promising in opening the blackbox, information bottleneck theory, to analyze a newly developed RNN cell model, hGRU, which managed to solve pattern recognition problem with long-range dependencies. Three points are made in our work: First, the hGRU is superior in long-range dependent problems. Second, the learning of hGRU is concentrated. Third, connection between hGRU layer and output layer restricts the model's performance.

1 Introduction

Naftali Tishby first analyzed the Deep Neural Network(DNN) under the information theory framework. In his work[1, 2], mutual information between hidden layers and data are calculated. He proposed that, during training process, DNN tries to minimize each hidden layer's mutual information with the input raw data while maximizing that with labels, which is named the information bottleneck thoery. A theoretical optimal bound was given following this intuition, pointing out the best performance a DNN can reach from information perspective. Besides that, by visualizing DNN on the information plane, where mutual information with input and labels are taken as x and y coordinate, he found the network experiences an initial "fitting" phase and a subsequent "compression" phase during training. "Compression" phase was hypothesized to be highly related with the generalization performance of the network.

But the following work[3] proved some points do not hold true. David D.Cox et al. demonstrated that the "compression" phase is from the tanh activation used in hidden neurons, and it is the nonlinearity of *arctan* function's two ends that cause the mutual information compression. They disassociated networks generalization in performance with the "compression" phase by training a deep linear network which generalizes well without information compression. However, applying mutual information theory and drawing information plane

[*]e-mail: 1610201@mail.nankai.edu.cn
[**]e-mail: drew_linsley@brown.edu
[***]e-mail: thomas_serre@brown.edu

is still inspiring and may serve as a tool to analyze the network's attribute. Thus, we plotted the information plane of a newly constructed model, hGRU[4], which managed to solve long-range dependent classification problems, to see what happens inside.

2 Mutual information estimation

The basic idea of Tishby's work[1] is to take every layer in network as a random variable. e.g. the i^{th} hidden layer can be treated as a variable, T_i. T_i can take different values, here we denote them as $t_0^i, t_1^i, ..., t_k^i$, where t is a vector comprised of neuron's activation value in i^{th} hidden layer. It is the same with input data, X, which can be different xs and an x should be a picture in the computer vision tasks. The corresponding labels, Y, is the assembly of various classes, ys. From this point of view, some statistic work can be done to estimate the probability needed to get mutual information. Here we show two useful ways to calculate it[1, 3].

2.1 Simple binning method

As mentioned above, hidden layer T can take various value ts. In the *arctan* activation case, activation value for each neuron is a real number ranging from -1 to 1. Because of the infinite accuracy of real number, the number of possible ts, k, is infinite. This makes it a problem if we want to count the probability because t is always different as long as input x changes. $p(x|t), p(t)$ would be only dependent on data sample number and thus training process makes no difference. To solve this, activation value was binned into 30 equal intervals in previous work[1]. For a layer with n neurons, the number of possible ts drops to 30^n, which is acceptable. Mutual information should then be calculated using the following basic equations:

$$\begin{aligned}
I(X;T) &= H(T) - H(T|X) && (1)\\
I(Y;T) &= H(T) - H(T|Y) && (2)\\
H(T) &= -\sum_{x\in X, y\in Y} p(t) log_2 p(t) && (3)\\
H(T|X) &= -\sum_{x\in X, y\in Y} p(x,t) log_2 p(t|x) && (4)\\
H(T|Y) &= -\sum_{x\in X, y\in Y} p(y,t) log_2 p(t|y) && (5)
\end{aligned}$$

We build a network with layer size of 12-10-7-5-4-3-2, following the setting of Tishby's work[1]. We take *arctan* function as activation function for the five hidden layers and softmax function for output layer. Loss is defined as the cross entropy between output and labels. We use the same training and testing dataset created in work[1], where input data is a vector containing 12 binary numbers and label is a binary number. This dataset is created with the intuition that it should subject to O(3) rotation invariant binary decision rules. As for initialization, all the bias are initialized as zero and weights are initialized using truncated normal distribution with standard deviation equal to reciprocal of the square root for last layer's size. Training set contains 4096 samples in total. The network is trained 8000 epochs with a batch size of 512.

[1]Code for Tishby's work is availble at: https://github.com/ravidziv/IDNNs

The procedure of mutual information calculation is as follows: First, we decide the epochs in which we calculate mutual information. This mainly depends on experience. Second, train the model. When training reaches those epochs, we do feed-forward for whole input dataset and store the corresponding activations. Third, do binning and counting on the stored data to get $p(t)$, $p(x)$, $p(y)$, $p(t|x)$, $p(t|y)$. Mutual information $I(X;T)$, $I(Y;T)$ are taken as x, y coordinates to plot one information plane. At last, all the steps above are repeated 50 times to get an average final information plane. See figure 1 for our results.

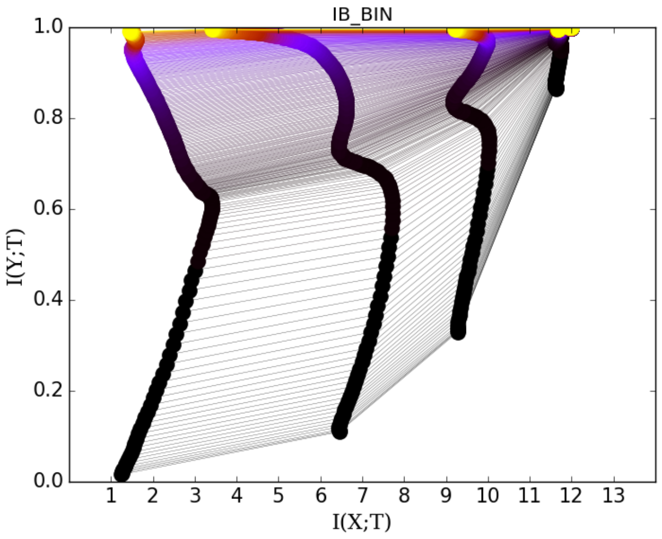

Figure 1. Reproducing result of information bottleneck theory. Each trajectory presents one hidden layer's dynamic during the whole training process. The left one is the output layer and the right ones are hidden layers. As for layer sequence, previous hidden layer is on the right.

Previous hidden layer has more mutual information with input data compared with the following layer. Mutual information with inputs is compressed layer by layer while information with labels is kept as much as possible. Compressing task-unrelated information and keep task-unrelated information is the principle of DNN, as work[3] pointed out. Besides, the network finishes most of the learning in first 1000 epochs, which agrees with the experience that training is always faster in the beginning for a DNN.

2.2 The KDE estimator

Simple binning method is good at dealing with discrete random variables with small state spaces. In experience, the information plane drawn by this method tends to have more fluctuation hence an average over many repeats is needed. However, repetition is not practical if network has a large size and needs long time to train. Thus David.D.Cox et al[3] applied a nonparametric KDE estimator[5, 6] to calculate mutual information. Besides that, it also guarantees that the result is not an artifact of simple binning-based information estimator.

The KDE approach estimates the mutual information between the data and hidden layer by assuming that the hidden activity is distributed as a mixture of Gaussians. The input data is thought to be distributed as delta functions at each sample in the dataset. i.e. we take the empirical distribution as true distribution for input activities. Hidden layer activity, t, is a deterministic function of the input, x. As claimed in work[3], the mutual information would be infinite if no noise is added[2]. Therefore a Gaussian noise with variance σ^2 is added, that is, $t^{'} = t + \epsilon$, where $\epsilon \sim N(0, \sigma^2 I)$. Under these assumptions, the distribution of T is genuinely a mixtures of Gaussians, with each Gaussian centered at the hidden layer activity t corresponding to its input example x.

In order to make it clear, we'll briefly introduce the mathwork of work[5, 6] here. Suppose variable Z is a mixture of N Gaussian distribution, whose probability density can be written as $p_Z(z) = \sum_{i=1}^{N} c_i p_i(z)$. $p_i(z)$ subjects to d-dimensional Guassian distribution $N(\mu_i, \Sigma_i)$. c_i means the weight for different Gaussian component and $\sum_i c_i = 1$. This can be treated as the joint probability distribution of variable Z and C, in which $p_{Z,C}(z,i) = p_i(z)c_i$. Kolchinsky & Tracey[5] demonstrated for a variable Z subjecting to mixture of Gaussians, its entropy of information has the following two tight bounds[3]:

$$H(Z) \le \hat{H}_{KL}(Z) := H(Z|C) - \sum_i c_i ln \sum_j c_j e^{-KL(p_i \| p_j)} \quad (6)$$

$$H(Z) \ge \hat{H}_{BD}(Z) := H(Z|C) - \sum_i c_i ln \sum_j c_j e^{-BD(p_i \| p_j)} \quad (7)$$

The definition of *Bhattacharyya distance* and *Kullback − Leibler divergence* are:

$$BD(p\|q) := -ln \int \sqrt{p(z)q(z)}\, dz \quad (8)$$

$$KL(p\|q) := \int p(z)\, ln \frac{p(z)}{q(z)}\, dz \quad (9)$$

When all of the mixture components have the same covariance matrix, as we assumed for noise ϵ, the upper and lower bound have a simpler form:

$$\hat{H}_{KL}(Z) = \frac{d}{2} - \sum_i c_i ln \sum_j c_j p_j(\mu_i) \quad (10)$$

$$\hat{H}_{BD}(Z) = \frac{d}{2} + \frac{d}{2} ln \frac{1}{4} - \sum_i c_i ln \sum_j c_j \tilde{p}_{j,0.5}(\mu_i) \quad (11)$$

$\tilde{p}_{j,\alpha}$ is mapped from p_j, which has the same mean but the covariance matrix is rescaled by $\frac{1}{\alpha(1-\alpha)}$. In our case, dimension for each Gaussian component d, equals to the size of hidden layer. Each hidden layer, $T^{'}$,[4] is a variable that subjects to a mixture of Gaussian distribution, each d-dimensional Gaussian component has the covariance matrix of $\sigma^2 I$ and the mean of t. i.e. $\mu = t$. When all the input examples are different, c should be the reciprocal of example numbers. $H(T^{'}|X)$ can be directly computed using the entropy for a d-dimensional Gaussian $N(\mu, \Sigma)$,

$$H(Z) = \frac{1}{2}[ln\ |\Sigma| + d\ ln\ 2\pi + d] \quad (12)$$

[2]The binning method is an approach to add noise in essence.

[3]The KDE estimator refers to $\hat{H}_D(Z) := H(Z|C) - \sum_i c_i ln \sum_j c_j e^{-D(p_i \| p_j)}$. $D(p_i \| p_j)$ is some (generalized) distance function between probability densities p_i and p_j. There are many distribution-distance function pairs that can be qualified for it.

[4]Note that noise is only added when doing MI calculation, so $T^{'}$ is used to distinguish from T

This is because T is deterministic of X, while $T^{'}$ is the noise-added T. Using equation (10)(11), we can get the bounds for $H(T^{'}), H(T^{'}|y)$. $H(T^{'}|Y)$ is computable by doing expectation of $H(T^{'}|y)$ over $p(y)$. Like the binning interval for simple binning method, KDE estimator has its own hyperparameter σ^2. By adjusting variance, this method gives the same answer with binning approach. In practice, it gives smoother information trajectory on information plane than that gotten from unrepeated binning method.

3 Information analysis of hGRU

3.1 Brief introduction to horizontal gated-recurrent unit

Drew Linsley & Thomas Serre[4] developed an updated version of the famous gated recurrent units(GRU) by introducing separate populations of inhibitory and excitory neurons into the GRU framework. Their work is biologically intuited from the horizontal connections in the visual cortex. hGRU is good at solving visual recognition problems with long-range dependencies which the modern CNNs struggle to solve. A pathfinder problem dataset was created to prove the superiority of hGRU in which the network has to learn whether two points in the graph is connected by a line or not. The result on this dataset showed this model outperformed state-of-the-art architectures even with orders of magnitude fewer free parameters.

3.2 Information analysis experiment setup

The training of hGRU only takes two epochs of training dataset thus we don't use the epoch to indicate training progress as in [4]. Batch size for the training is 32 and the network is trained over 56108 batches. The network is trained using one batch at a time, hence we use training step to show its progress. There are three layers in hGRU network: a convolution layer to input data(l1 layer), the hGRU layer and the output layer. The input data is from the pathfinder challenge dataset with the path length of 14 paddles. Input pictures has the size of 150×150×1 while the activation of l1 layer, hGRU layer and output layer are 150×150×25, 150×150×25, 2. The hGRU layer is recurrent so we fix its time step to 4, 6, 8 to see the change of its information properties. In some specific training steps, we do validation to the network, during which we take these layers' activities. After that the activations of l1 and hgru layer are compressed to 1×1×25 by summing over feature maps to ease the information calculation.[5] Then the summation activation is re-scaled to keep its range close to one. i.e. the hgru model is treated as a 22500-25-25-2 network. 16384 validation examples are used to generate activations. We decide the hyperparameter of the KDE estimator, the variance σ^2, following two rules: First, mutual information with labels $I(Y; T^{'})$ should be close to one at the end of training because the network must have captured the almost all one bit information of binary label. Second, the shape of trajectories on information plane should remain unchanged when the number of examples changes, as long as examples are enough for mutual information computation. Besides that, binning method is also used to be a reference for KDE method, as what work[] did. In practice, we take $2 \times 10^{-8}, 1 \times 10^{-5}, 4 \times 10^{-4}$ as the variance for l1, hGRU and output layers.

[5]Directly take all the activation units of hGRU or l1 layer will meet great difficulty because it has the size of 150×150×25. For binning method taking 30 equal intervals, it means there are 30^{562500} possible states. For KDE estimator, the Gaussian distributions are 562500 dimensional. The number of data examples needed to compute information would be larger than the size of training dataset. In work[3], they built network with size 784-1024-20-20-20-10 on the MNIST handwritten digit classification dataset, layer size being much smaller than 562500, and 10,000 samples were used to computed mutual information.

3.3 Experiment results and discussion

Figure 2 shows the information planes containing output and hGRU layer with 4, 6, 8 timesteps. The trajectory for output layer is on the left with an irregular shape while hGRU information trajectory appears to be a concave curve. During training process, both two layers' mutual information with labels increases, from zero to around one. Only the output layer in the four timesteps information plane reaches much lower mutual information with labels than others, which agrees with the truth that the four timesteps model fails in the pathfinder problem.[6]

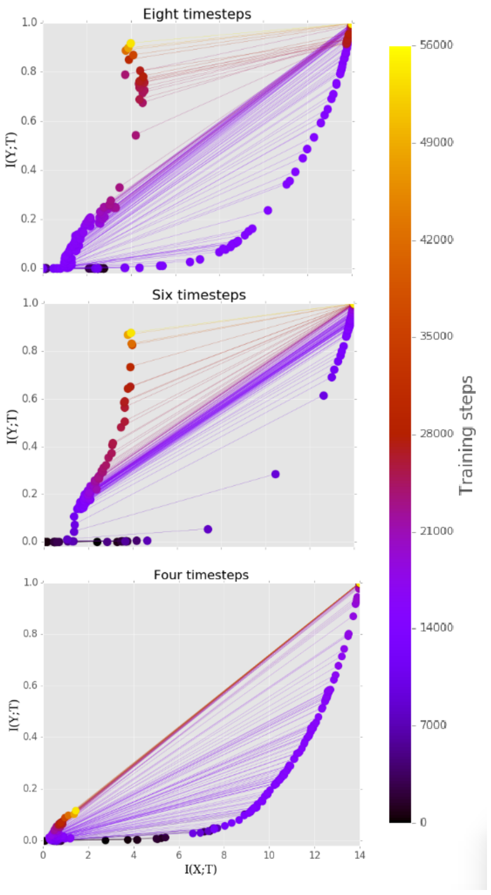

Figure 2. Information plane for output and hGRU layers. The eight timesteps model and six timesteps model get accuracy close to one after 56000 training steps. Only the four timesteps model fails, which can been seen easily from information plane because the output layer doesn't take enough useful information about labels.

Three conclusions can be judged from this result: First, hGRU layer has its superiority in capturing information from long-range dependent problems. Noticed that hGRU layer makes progress steadily instead of moving in roundabout ways, its trajectory is relatively smooth considering that this is from only one repeat. With 25 features maps, hGRU layer has

[6]The training accuracy only reaches 0.6 after 56000 training steps.

large capacity in capturing information and it tends to learn without compressing $I(X;T)$. In the beginning of training, the trajectory slope is smaller as a result of absorbing more task-unrelated information. Sometimes it even goes backward as what black dots show in the six timesteps model. After that, it starts to take more and more task-related information as the slope grows bigger. This might be that it finds the correct way for capturing the information for target after the first exploring stage. It stays and moves little for the following 35000 steps. Second, the learning for hGRU layer is concentrating. Most of the learning happens in the steps around 14000. This agrees with our claim about hGRU layer's superiority because it can learn with fewer training steps. Even in the four timesteps model, the hGRU layer finishes its learning before 21000^{th} training step. This is different to what happens to DNNs which learn fastest at the beginning of training, from the information perspective. Third, the information transmittion between hGRU layer and output layer is the restriction for model performance. In figure 2, the mutual information with labels only reaches 0.2 when hGRU layer has finished its learning process. As mentioned above, most of the learning for hGRU layer happens around 14000^{th} step, however output layer learns mainly after 21000^{th} step. It possibly results from the sharp dimension decrease from $150\times150\times25$ to 2. From step 21000 to 56000, the hGRU stays almost unchanged, waiting for the learning of the output layer. To make an analogy, output becomes the bottleneck for the whole networks performance. Enhancing the representing ability between these two layers might be a good choice by adding some hidden layers.

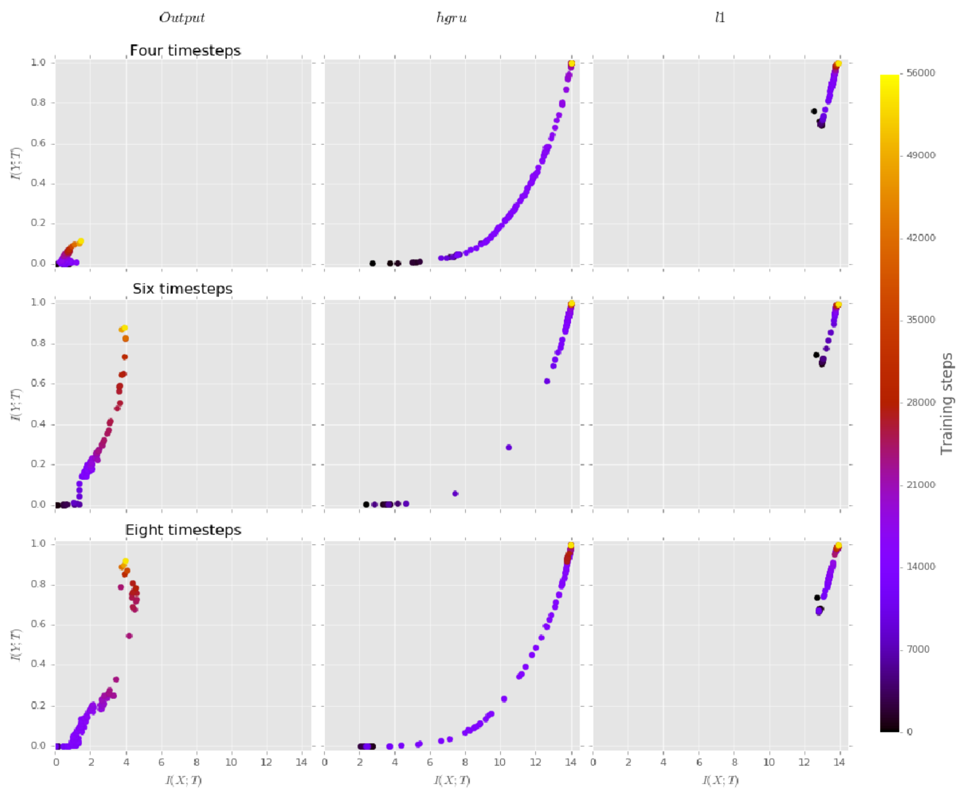

Figure 3. Information trajectory for output, hGRU and l1 layers. l1 convolution layer starts with a higher $I(X;T)$, $I(Y;T)$ and its trajectory has a much smaller span compared with hGRU layer.

In figure 3, information trajectory for different layers are plotted individually. l1, the convolution layer, starts with higher mutual information with input data and labels because its activations are more similar to input data. This agrees with Tishby's result on DNNs in which lower level layer starts with more $I(X;T)$ and $I(Y;T)$. In all three cases the hGRU and l1 layer finish learning, thus again demonstrate the idea that the link to output layer restricts the performance.[7]

References

[1] Tishby N, Zaslavsky N. Deep learning and the information bottleneck principle[C]// Information Theory Workshop. IEEE, 2015:1-5.

[2] Tishby N, Pereira F C, Bialek W. The information bottleneck method[J]. University of Illinois, 2000, 411(29-30):368–377.

[3] Saxe A M, Bansal Y, Dapello J, et al. Published as a conference paper at ICLR 2018 On The Information Bottleneck Theory Of Deep Learning[C]// Iclr. 2018.

[4] Linsley D, Kim J, Veerabadran V, et al. Learning long-range spatial dependencies with horizontal gated-recurrent units[J]. 2018.

[5] Kolchinsky A, Tracey B D. Estimating Mixture Entropy with Pairwise Distances[J]. Entropy, 2017, 19.

[6] Kolchinsky A, Tracey B D, Wolpert D H. Nonlinear Information Bottleneck[J]. 2017.

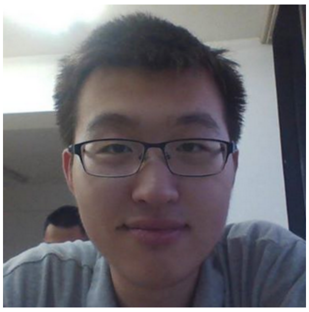

Figure 4. Xinhao Fan. (I just want to put one of my photo here.) This work takes longer than I expected. I met and solved lots of problems during this internship. I once thought it's impossible for me to solve them. Adjusting network parameters to fit Tishby's result took much time(getting hGRU data is time-consuming too). It'll be better if I can find the best parameter sooner, then I'll be able to help more in Matt's work. Whatever, I kind of experienced the feeling of reading a phd...Just write something a memento.

[7]Code is available at: https://github.com/WOOOOOOOOOOOOOO/Information-hGRU