Opening the black box of Deep Neural Networks with More Information

March 23, 2018

Abstract

A lot of methods and approaches exist for effective training and application of neural networks, however they are still not well understood theoretically. In this project we explore the evolution of various neural networks trained with stochastic gradient descent using information plane approach proposed by Schwartz-Ziv and Tishby, replicate their experiments as well as introduce our own modifications.

1 Introduction and related work

Although neural networks have proven to be invaluable in research as well as in automating various business processes, a lot of questions remain open as to how they work and why they are so effective. One model developed (by *Shwartz Ziv, Tishby*) to answer them is the **information plane**

From a statistical point of view, the data X and labels Y used to train a neural network that outputs prediction $\hat{Y}$ are two random variables drawn from some unknown distribution. The *activation value* (output of the activation function) of each layer of the network $T_i \in \{T_1, T_2, ..., T_n = \hat{Y}\}$ is a function of X and, thus, a random variable as well.

As a way to understand this sequence of random variables, the original paper by *Schwartz-Ziv and Tishby* proposes using *mutual information*:

$$MI(X;Y) = -\sum_{x,y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)}$$

where X,Y are distributions and $p(x)$, $p(y)$, $p(x,y)$ are probability mass functions

Every layer T_i of a neural network thus is a point in the **information plane**: $(MI(X;T_i), MI(T_i;Y))$ where $MI(X;T_i) \geq MI(X;T_{i+1})$ and $MI(T_i;Y) \geq MI(T_{i+1};Y)$. This gives rise to an intuitive model of neural network being an **information funnel** that layer by layer extracts the most *relevant* (informative of Y) information out of X and discards the rest.

Schwartz-Ziv and Tishby present certain that support the information funnel model as well as demonstrate that the process of *Stochastic Gradient Descent* for a neural network tends to decompose into 2 stages:

- Fitting the network to the data ("moving the funnel"). During this stage $MI(\hat{Y};Y)$ and $MI(\hat{Y};X)$ grow.
- Compressing the output of the network ("shrinking the funnel"). During this stage $MI(\hat{Y};Y)$ grows slower and $MI(\hat{Y};X)$ shrinks.

Our goal in this project is to replicate these experiments as well as further explore this model with our own tools and experiments

2 Experiments

Please note, that we ran a lot more than 3 experiments in the process of preparing this report, but here we omit small modifications of experimental setup that don't change the output as they are remarkably not interesting to read. Because of the 5-page limit, the jupyter notebook has more experiments, than the text report

2.1 Estimating mutual information

According to the formula

$$MI(X;Y) = -\sum_{x,y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)}$$

in order to estimate the mutual information between X, Y and a layer, one *simply* needs to estimate $p(x)$, $p(y)$ and $p(x,y)$ for all x and y. Thus, we need The following procedure is used (**but not described**, one has to look through the code attached to the paper to find that out) in the original paper: 1. Given dataset X, Y estimate $p(x) = \frac{\text{count}(X=x)}{\text{count}(X)}$, $p(y) = \frac{\text{count}(Y=y)}{\text{count}(Y)}$ as well as joint $p(x,y) = \frac{\text{count}(X=x,Y=y)}{\text{count}(X,Y)}$ 2. Run *all* data points x in X through the neural network, obtain T_i for all i 3. Repeat step 1 for every (T_i, X)

Note that this approach requires all random variales to be discrete, hence the activation values of neurons have to be discretized

Because the procedure is not described in the paper and the code is very hard to parse, we had to give up the idea of re-implementing it and make heavy use of it's API. However, we did implement an improvement upon the original estimation approach:

2.1.1 Extended MI estimator

The model described above treats X, Y and T_i as random variables, but assumes the function $T(X)$ is completely deterministic. Which it is not, because we are training the neural network with *stochastic gradient descent*: every time we retrain it, the results will be slightly different.

We have implemented an MI estimator that accounts for this. Instead of working with one neural network, we can train several and sample elements of T_i (activation values of neurons) randomly from several T_{ij} (i - layer, j - index). From now on, this will be called *extended MI estimator* and the original approach with one network - *simple MI estimator*. We will demonstrate the benefits of this approach further.

2.2 Experimental setup

Every experiment was conducted as follows:

- Several copies of a neural network with identical architecture (number, size and type of layers) are created.

- They are trained iteratively using pytorch implementation of *Stochastic Gradient Descent* After each iteration of the algorithm, the full procedure *extended MI estimator* is run and the estimated mutual information $MI(X;T_i)$, $MI(T_i;Y)$ is plotted on a graphical representation fo the **information plane**

Note that MI estimation is much more resource intensive than SGD itself and it doesn't run on the GPU. For this reason, going from primitive multi-layer perceptrons to experiments with more complicated neural networks is *very hard*.

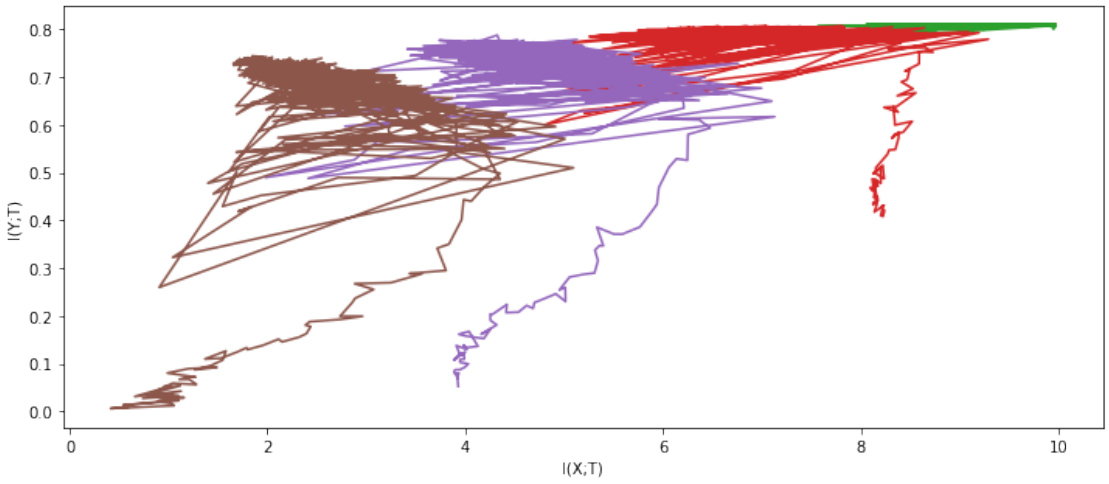
2.3 Data

In order to replicate *Tishby an Schwartz-Ziv's* original experiment, we'll generate synthetic test data using

- a simple binary decision function with small Gaussian noise.
- another, more complicated decision function, in order to compare the results.
- and finally, MNIST dataset of hand-written digits, to test if the results generalize from the synthetic dataset to a more realistic one

2.4 Experiment 2: As close to Tishby setup as possible

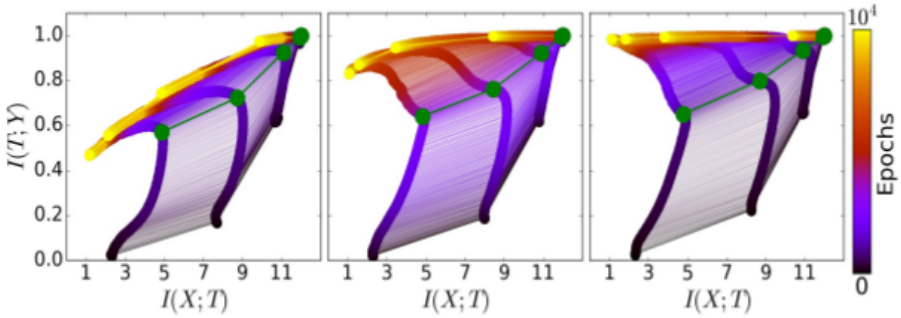
Six-layer perceptron, simple MI estimator, simple synthetic dataset



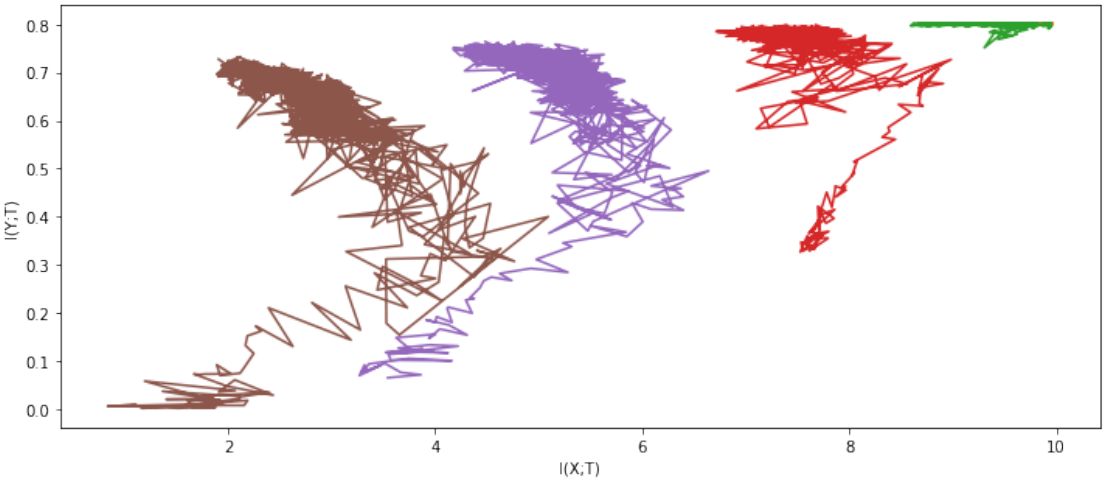
Different colors represent different layers of the neural network. As one can see, this network is too simple for the observations of *Schwartz-Ziv and Tishby* to manifest.

For comparison, here are the results from the original paper:

Indeed, the experiment reproduces successfully. But what if we use the same network architecture and the same data, but with our *extended MI estimator* ?



2 Stages

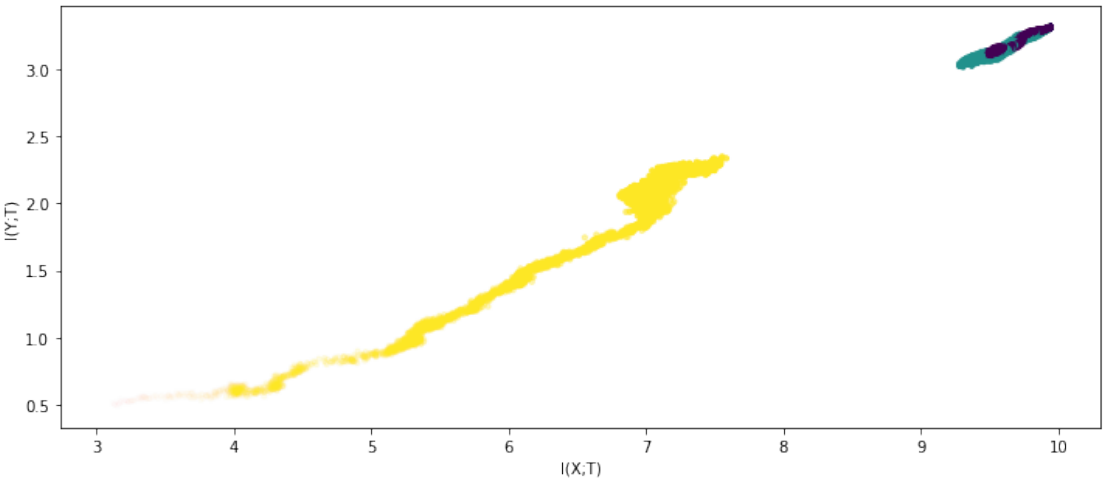


Feel free to take a second to process the awe and admiration of this clearly superior method you are experiencing at the moment.

2.5 Experiment 4: (Almost) Real-world problem

For this experiment we will use:

1. MNIST dataset of hand-written digits
2. Convolutional neural network (1 conv layer, 1 ReLU layer, 2 fully connected layer) roughly inspired by *Cireşan* et al
3. Extended mutual information estimator



2.6 Bonus: mutual information stopping criterion

It felt wrong to end this project without a more or less practical (i.e. code you can use) result, so here it is: an algorithm for training neural networks that uses mutual information profile of the last layer $I(X;\hat{Y}) - \beta I(\hat{Y};Y)$ as its stopping criterion. Because, as already mentioned, MI estimators are very slow, we do not check this criterion at every iteration.

Code is attached

2.7 Conclusions

- Original experiment **replicated successfuly**
- Small changes to the original experiment didn't change the results, but on very different (like the convolutional one) neural networks, the experiment turned out hard to reproduce. However, we cannot conclude that *Shwartz-Ziv and Tishby's* results were erroneous , because there are other possible explanations like not having enough time and resources to run the experiment till stage 2

We have also introduced:

- An improvement of the mutual information estimator with several neural networks
- A new stopping criterion for neural network optimization

3 Contributions

- **Vadim** - almost everything
- **Leonid** - a few slides, thoughts and prayers, valuable feedback

4 References

- *Ravid Shwartz-Ziv, Naftali Tishby* Opening the Black Box of Deep Neural Networks via Information
- *Dan C. Cireşan, Ueli Meier, Jonathan Masci, Luca M. Gambardella, Jürgen Schmidhuber* Flexible, High Performance Convolutional Neural Networks for Image Classification