**Andrius Grabauskas**

# Measuring mutual information in Neural Networks

Computer Science Tripos – Part II

Robinson College

Tuesday 23rd April, 2019

# Proforma

| | |
|---|---|
| Name: | **Andrius Grabauskas** |
| College: | **Robinson College** |
| Project Title: | **Measuring mutual information in Neural Networks** |
| Examination: | **Computer Science Tripos – Part II, July 2001** |
| Word Count: | **2800**[1] |
| Project Originator: | Dr. Damon Wischik |
| Supervisor: | Prof. Alan Mycroft |

## Original Aims of the Project

## Work Completed

## Special Difficulties

[1] This word count was computed by `detex *.tex | tr -cd '0-9A-Za-z \n' | wc -w`

# Declaration

I, Andrius Grabauskas of Robinson College, being a candidate for Part II of the Computer Science Tripos, hereby declare that this dissertation and the work described in it are my own work, unaided except as may be specified below, and that the dissertation does not contain material that has already been used to any substantial extent for a comparable purpose.

Signed

Date

# Contents

# List of Figures

# Acknowledgements

# Chapter 1

# Introduction

Deep Neural Networks (DNNs) are an extremely successful tool, they are widely adopted commercially and closely studied academically, however even given the attention they have gathered there is no comprehensive understanding of how these models generalize data and provide such impressive performance - in fact very little is know about how DNNs learn or about their inner workings. Recently Prof. Tishby produced a paper claiming to understand the basic principle of how DNNs work. He suggested that there are two phases that the network goes trough while being trained - the fitting phase and the compression phase. During the fitting phase the network memorizes the training data and makes predictions based on what it has seen before, during the compression phase the network generalizes, it forgets the unnecessary information from the training data. Tishby suggested that the incredible performance that DNNs are able to achieve is due to this compression phase, and that this process of compression is a result of randomness inherent in Stochastic gradient descent. Tishby showed this by looking at DNNs trough the information domain, most notably he used what is now called the information plane. The information plane summarizes how the information is flowing trough the DNN, for every neuron layer the plane shows mutual information it has with the input data and the label data. In his experiments Tishby has concluded that every layer loses unnecessary information from the input data and tries to keep information of the label. Tishby made some interesting and significant claims about how DNNs work, however he did no provide a formal proof, his conclusion are based only on experimental evidence.

In our work we look at Tishby's claim that DNNs compress data and throw away unnecessary information about the input. We reimplement his experiments as a form of independent verification, showing that the results Tishby got are robust and are stable to parameter changes. We also take a look at a paper produced by Saxe that provides an opposing view to that of Tishby's. Saxe showed that compression that Tishby showed is only a result of Tishby's choice of activation function for the neural network. He showed that compression only happens when Tanh activation function is used and does not happen when ReLu is used.

Lastly, we think that the experiments presented in both papers don't fully align with the ideas that Tishby presented to us, specifically his idea that weights should be treated "as if" they are random variables. Tishby's and Saxe's experiment deal with this "as if" random notion quite crudely or try to sidestep it completely. As a result we devised an

experiment that tries to capture this idea more explicitly, although it is still relatively crude and more work should be put into it in the future.

## 1.1 The Information plane

If we look at a neural network trough the lens of information we can think of it as a form of a Markov chain. Where every node takes in a piece of data processes it and passes to the next node. For every such node $t \in T$ we can measure it's mutual information with the input patterns $x \in X$ and labels $y \in Y$.

$$Y - X \rightarrow T_1 \rightarrow ... \rightarrow T_i \rightarrow \hat{Y}$$

In this analogy a node corresponds to a layer in a non-convolutional neural network. When looking at a neural network in this way we can say that it's job is to preserve as much information as possible about the label, or to minimise information loss about the label during the transitions from node to node.

The Information plane summarized this Markov chain view and applies it to the entirety of the training process.

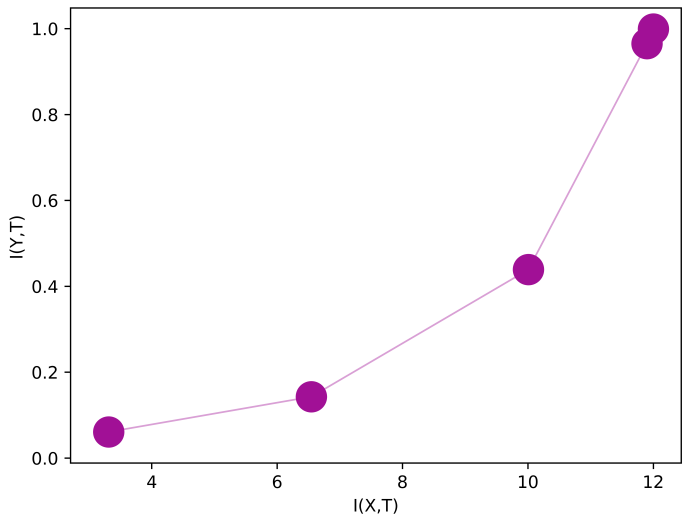


Figure 1.1: Information plane for a neural network with 5 layers, which was only trained for one epoch.

Figure 1.1 shows an example of an information plane - in this case the network has 5 layers and has only been trained for one epoch. Every node corresponds to a layer, and the lines between them just help us distinguish the order of the layers. The $x - axis$ shows mutual information between any layer $T$ and the input $X$, while the $y - axis$ shows mutual information with the label $Y$.

The figure bellow corresponds to a network that takes in a 12bit number and maps it to a 1bit number hence the mutual information ranges from 0 to 12 for the $x - axis$ and from 0 to 1 for the $y - axis$.

The upper right most node corresponds with the very first layer before any data processing occurs, therefore it has maximum mutual information with the input and the label. As the network was only trained for one epoch the weight are essentially random and we see a steep drop in every subsequent layer, with the last layer of the network having close to 0 mutual information with $X$ and $Y$, which is what we expect if we assume that the network guesses at random before any training is done.

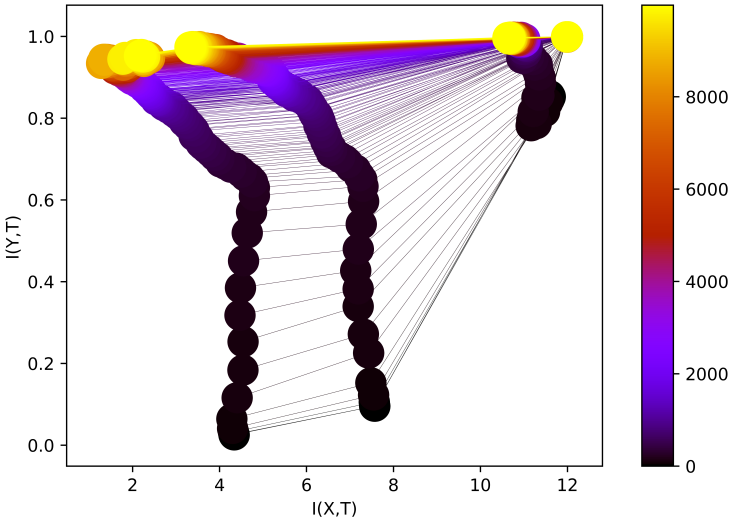


Figure 1.2: Information plane for a neural network with 4 layers, which was trained for approximately 10 000 epochs

Figure 1.2 shows an example of a full information plane. The gradient maps the colour to the epoch. From this figure we can clearly see the fitting and the compression phases Tishby was describing.

- The Fitting phase is the rapid improvement phase at the start of the training period, where information about the label is rapidly increasing while information about the input remains approximately the same. In this case the training is in the fitting phase for less than 1000 epochs.

- The Compression phase begins when the after the fitting phase ends and the rapid improvement stops. We can see during this phase we start losing information about the input while still gaining information about the label. This phase the was majority of the training time.

# Chapter 2

# Preparation

Before developing a plan for how we are going to realize the project in code we needed to fully understand the ideas presented in the paper:

- We needed to identify the main ideas of the paper and understand why some parts of the paper are not agreed upon in the scientific community. Understand why his ideas are contentious and whether reproducing his experiments could bring more validity to his claims. This involved reading papers published by Tishby and academics who shown an opposing view to him.

- A main tool that the paper relies on is MIE (Mutual Information Estimation). Reading about MIE we quickly understood that MIE is a contentious part of the project as a result we had to do a decent amount of research regarding the subject. MIE is difficult because we are trying to estimate information between two continuous distributions using only a discrete sample set. This area has not seen much academic attention so the tools we ended up using could be greatly improved in the future.

Once we had a reasonable understanding of the ideas in the paper and which areas needed more attention we diverted our attention to figuring out the details of how the experiments were conducted figure out what hyper parameters Tishby decided are important and what assumptions he made whilst devising the experiments.

In addition we needed to find out what resources are available to us online, what programming frameworks we are going to use for the projects implementation, and to think about possible extensions to the project once the success criteria has been achieved.

- Online Resources: The two main papers by Tishby and by Saxe have made their code public online via Github, we made

  Online Resources: The two main papers we were looking at has made their code available to the public via Github, the papers are Tishby's paper and the main opposing paper by Saxe.

- Programming frameworks: The original experiment implementation by Tishby has used the Tensorflow framework. We have decided to use the Keras framework as it produces code that is more concise and is easier to read/maintain. Furthermore

rewriting the experiments in a different framework means that we cannot rely on the details of Tishby's and potentially avoid any mistakes that may exist in the original implementation.

- Thinking about how we could extend the project helped us understand the scope of the project and what areas were most important and/or interesting to us.

  We came up with a couple of extensions before having written any code but the most interesting one only materialized after a good deal amount of work into the project (that is the AS IF Random experiment described below)

  - Different Datasets : the most straight forward extension to the project just using different dataset to the one Tishby used. This is essentially just varying one of the parameters in the Neural Network. (Implemented)
  - Quantized Neural Network : the idea behind this was to only allow single neurons to acquire values in a given range say 1...256. This would make the distribution within a DNN later discrete and hence it would make calculating mutual information straightforward. (Not Implemented)
  - As If Random : one problem with Tishby's work is that he calculates mutual information for a single epoch at a time which by definition is zero (in his paper he tries to justify the result will explore this later) this extension tries to explore the weights of a neural network as random variables by calculating mutual information for multiple epochs at a time.

# Chapter 3

# Implementation

## 3.1 Mutual Information Estimation

As stated before we are trying to validate the fact that compression is happening within the hidden layers in a neural network as such having robust tools to measure mutual information is of utmost importance.

### 3.1.1 Mutual Information Definition

Mutual information is a measure of dependence between two variables $X$ and $Y$, it quantifies the number of bits obtained about one when observing the other.

Suppose we have two random variables $X$ and $Y$ if we want to measure mutual information between the two of them we usually refer to one of the following equations:

Using the entropy of the distributions to infer the value

$$I(X,Y) = H(X) - H(X|Y) \tag{3.1}$$

$$I(X,Y) = H(X) + H(Y) - H(X,Y) \tag{3.2}$$

Or calculating the mutual information explicitly from probabilities, for discrete and continuous distributions respectively

$$I(X;Y) = \sum_{y \in \mathcal{Y}} \sum_{x \in \mathcal{X}} p(x,y) \log \left( \frac{p(x,y)}{p(x)\, p(y)} \right) \tag{3.3}$$

$$I(X;Y) = \int_{\mathcal{Y}} \int_{\mathcal{X}} p(x,y) \log \left( \frac{p(x,y)}{p(x)\, p(y)} \right) dx\, dy \tag{3.4}$$

### 3.1.2 Theoretically undefined

Calculating mutual information is mathematically defined for both continuous and discrete distributions by using one of the functions above. However since we are looking at neural networks we do not have the full distribution - only an empirical sample of the unknown joint distribution $P(X,Y)$. As such we cannot use conventional methods to calculate the mutual information and instead have to use tools to estimate it.

The field of estimating mutual information is relatively new as a result the tools that are available are quite primitive. There has been some papers published that use more advanced techniques to tackle the problem, we will talk about in subsection 3.2.4

## 3.2 Calculating Mutual Information

### 3.2.1 Experimental setup

```
Input:
T := number of layers
X := training data
Y := label data
N := number of epochs
Output:
I_x(epoch, layer), # mutual information with X
I_y(epoch, layer)  # mutual information with Y
Algorithm:
NN = setup_neural_network(T, X, Y)
for e in 0..N:
  NN.run_SGD_once()
  for t in 0..T:
    data = []
    for x ∈ X:
      t̂ = NN.predict(x).layer_activation(t)
      data.append(t̂)
    I_x(e, t) = calculate_mutual_information(data, X)
    I_y(e, t) = calculate_mutual_information(data, Y)
```

Figure 3.1: The general algorithm for calculating mutual information inside a neural network.

- X input is an empirical sample of all possible inputs.
- Y label data. Every $x \in X$ has a corresponding label $y \in Y$.
- $T_{e,i}$ data for a specific epoch $e$ and specific layer $i$ in the neural network. The dataset is generated by feeding every $x \in X$ trough the neural network and recording the activations.

### 3.2.2 Discrete method

The method used by Tishby in his paper. It assumes that the discrete empirical distributions we manage to collect are a good representation of entropy with in the dataset. The method relies heavily on computing entropy values for dataset and it's subsets, as outlined in Equation 3.2.2

$$I(X,Y) = H(X) - H(X|Y) = H(X) - \sum_{y \in Y} H(X|Y=y)p(Y=y) \tag{3.5}$$

However just calculating mutual information for a discrete distribution is not enough as this yields mutual information equal to 0, in order to sidestep this issue we need to simulate randomness Tishby achieves this by grouping multiple values together. A more detailed explanation why this is done is given in subsection 4.0.1

Figure 3.2 trough Figure 3.4 outline the full algorithm.

```
1 Input:
2 X = x_1, x_2, ..., x_n
3 Y = y_1, y_2, ..., y_n
4 Output: I(X : Y)
5 X = Batch close values of X together
6 Y = Batch close values of Y together
7 H(X) = Calculate entropy of X
8 H(X|Y) = Calculate conditional entropy of X given Y
9 I = H(X) − H(X|Y)
```

Figure 3.2: Pseudo code for computing mutual information refer to Figure 3.3 and Figure 3.4 for entropy computation.

```
1 Input: X = x_1, x_2, ..., x_n
2 Output: H(X)
3 for x̂ ∈ Unique(X):
4   count = 0
5   for x ∈ X:
6     if x̂ = x:
7       count = count + 1
8   P_x = count / len(X)
9 H(X) = − Σ_{x∈Unique(X)} P_x log(P_x)
```

Figure 3.3: Algorithm for computing entropy

```
Input:
X = x_1, x_2, ..., x_n
Y = y_1, y_2, ..., y_n
Output: H(X|Y)
H(X|Y) = H(X)
for ŷ ∈ Unique(Y):
  X̂ = []
  for x, y ∈ zip(X, Y):
    if ŷ = y:
      X̂.append(x)
    H(X|Y) = H(X|Y) − H(X̂)
```

Figure 3.4: Algorithm for computing conditional entropy

### 3.2.3 Kernel Density Estimation

### 3.2.4 Advanced methods

#### Mutual Information by Gaussian approximation

failure

#### Geometrical Adaptive Entropy Estimation

complete and utter failure

## 3.3 Implementation Optimizations

- parallelizing mutual information calculation. (good for kernel density estimation bad for discrete method.)
- calculating mutual information for multiple epochs at the same time. (bad for memory, good example for MNIST SET wasn't able to apply this method for it) parallelization
- choosing next epoch logic

---

Calculating mutual information is fully defined for the discrete and continuous distributions, however we have a continuous distributions and a discrete sample sets of those distributions. Our distributions are:

- $X$ - the input set (training set) to the NN (Neural Network)
- $T_i$ - activation of every NN layer, and

- $Y$ - the output set (labels of the training data)
- As stated before Mutual information measurement is a contentious part of the project. Understanding why it's hard is useful in order to correctly evaluate the experiments.
- ¡++¿
- why Mutual Information Estimation is hard

  Yeah why?????

  Explain what it is - two continuous distributions we have a discrete sample

  Why is it weird in this case our X is a discrete (sometimes continuous distribution) uniform distribution

  A layer produces a continuous distribution

  Estimating mutual information boils down to how well we can measure entropy.

  Every layer is a vector, this does not affect theories for mutual information but makes the code more difficult to handle unless heavy abstraction is used.

  There has only been little work in this field so no publicly available collection of MIE (Mutual Information Estimators) exist, I was able to find some implementations of MIE by other researchers but they were not general enough or just didn't work.

  There is some theoretical work in the subject but finding an implementation of the algorithms described in the paper has been difficult

  I was left with two options using the same methods as used by Tishby and Saxe or trying to implement an advanced algorithm from one of the papers. I tried to implement

  In our specific example:

  - We need to produce mutual information values for:
    - $I(X, T_i)$ - input and every NN layer
    - $I(Y, T_i)$ - output and every NN layer

    Where:

    - $X$ - Input distribution
    - $Y$ - Output distribution
    - $T_i$ - Distribution of $i'th$ layer in the neural network
  - Problem: we know that every input value is unique and that weight matrix for every layer in the NN is reversible, which implies that every input corresponds to a unique network activation, however that means that no information is lost in the network and compression does not happen. Which implies that for any single epoch mutual information between any two layers in NN or the input is just equal to the entropy of the input as H(X—Y) is always

  - ¡++¿

- what have I tried

  - wgao - tried to implement, but failed the implementation was too complex and decided to cut my loses as it was taking too much time to implement. Emailed the author, he wasn't able to provide any code.
  - wgao9 (lnn) - local nearest neighbour mutual information estimation. Based on a paper1 the code was available online on Github. The code produced nonsensical results highly sporadic and often even negative results for measure of mutual information. Although the code might have been fixable running it took an extremely long time so I've decided to not use this method as it already has consumed a lot of my time.

- what worked

  - Tishby - describe how Tishby calculated mutual information, what assumptions he made and what faults it has.
  - Saxe used kernel density estimation in order to calculate entropy for some of their experiments, it a rigorous way to measure entropy however the results produced have big error bounds. see Kolchinsky and Tracey "Estimating Mixture Entropy with Pairwise Distances" for upper and lower bounds

- even though the MIE we ended up using is by far not the most sophisticated technique, it is still the state of the art that has been used in regards to measuring mutual information inside neural networks.

## 3.4 Tishby's reproduction

## 3.5 Saxe's reproduction

# Chapter 4

# Evaluation

### 4.0.1 Deterministic networks

There's a very real argument to be made against compression in neural networks. Consider a generic neural network we can think of it as a function that is a series of matrix transformation, where a matrix corresponds to weights of a specific layer. However these matrices are all random (at least at the start of training) and hence probability of them being invertible is 100%.

Knowing that every single matrix is invertible allows us to conclude that that neural network as a whole is an invertible function, which means no information is lost and compression is impossible.

# Chapter 5

# Conclusion

# Appendix A

# Project Proposal

Computer Science Tripos: Part II Project Proposal

# Measuring mutual information within Neural networks

Andrius Grabauskas, ag939
Robinson College
**Saturday 20th October, 2018**

| | | |
|---|---|---|
| **Project Originator:** | Andrius Grabauskas | |
| **Project Supervisor:** | Dr. Damon Wischik | |
| **Director of Studies:** | Prof. Alan Mycroft | |
| **Overseers:** | Dr. Robert Mullins | Prof. Pietro Lio' |

## Introduction and Description of the Work

The goal of this project is to confirm or deny the results produced by Shwartz-ziv & Tishby in their paper "Opening the black box of Deep Neural Networks via Information"[1]

The paper tackles our understating of Deep Neural Networks (DNN's). As of yet there is no comprehensive theoretical understanding of how DNN's learn from data. The authors proposed to measure how information travels within the DNN's layers.

They found that training of neural networks can be split into to two distinct phases: memorization followed by the compression phase.

- memorization - each layer increases information about the input and the label
- compression - this is the generalization stage where each layer tries to forget details about the input while still increasing mutual information with the label thus improving performance of the DNN. This phase takes the wast majority of the training time.

They found that each layer in neural network tries to throw out unnecessary data from the input while preserving information about the output/label. As the network is trained each layer preserves more information about the label

[1] https://arxiv.org/abs/1703.00810

The results they found were interesting but also contentious as they have not yet provided a formal proof, just experimental data as a result there are many peers that are cautious and sceptical of the theory even a paper[2] was produced that tries to suggest that the theory is wrong, however this was dismissed by Tishby & Shwartz-Ziv[3]

## Starting Point

I have watched a talk that Prof. Tishby gave on this topic at Yandex, no other preparation was done.

## Resources Required

The training DNN's and measuring mutual information will be computationally expensive so I will be using Azure cloud GPU service to acquire the required compute for this project. The GPU credits will be provided by Damon Wischik

For backups I intend to store my work on GitHub and my own personal machine. In case my laptop breaks I will get another one or use the MCS machines.

## Substance and Structure of the Project

The aim of this project to reproduce the results provided by Prof. Tishby and his colleagues. The intention of my work is to help settle the debate surrounding the topic either strengthening the arguments in favour of the theory in case my results are inline with the aforementioned results or encourage discussion in case my results contradict the theory.

My work will require me to have a comprehensive understanding of Information theory, Information bottleneck and neural networks.

One of the more contentious parts of my project will be measuring mutual information between the input a layer in the DNN and the label. It will be computationally expensive to measure it in DNN since we will need to retrain the network in order to get a distribution rather than a single value. I will use Gaussian approximation to measure it (relevant paper[4])

Will need to use Python to train the neural networks and GNUplot or alternative to plot the results.

## Success Criteria

Reimplement the code that was used to generate the papers results. Confirm or deny the results produced in"Opening the black box of Deep Neural Networks via Information" paper on the same dataset as the paper. In order to do that I will need to: Train a neural network on the same dataset

[2]https://openreview.net/pdf?id=ry_WPG-A-
[3]https://openreview.net/forum?id=ry_WPG-A-&noteId=S1lBxcE1z
[4]https://arxiv.org/abs/1508.00536

that was used in the paper and measure mutual information between the layers. Analyse the results produced and address any discrepancies that may have occurred.

# Extensions

Provided I achieve the success criteria there are two main ways to extend it.

- Use different datasets to test the theory. Using different datasets would confirm that the results are not data specific. Current datasets we are considering: MNIST[5] and NOT-MNIST[6].
- Explore different ways of measuring mutual information. One interesting way would be to explore a discrete neural network where every node would only be able assigned discrete values say 1...256. This would make the distribution within a DNN layer discrete and hence it would make calculating mutual information straightforward. However quantizing the neural network could possibly hurt the performance of the network.

[5]http://yann.lecun.com/exdb/mnist/
[6]https://www.kaggle.com/quanbk/notmnist

# Schedule

- **20th Oct – 2nd Nov**

  I expect to spend the first two weeks reading up on Information theory (primarily from Mackay's book[7]) and the information bottleneck method in order to understand the nuances of the paper.

- **3rd Nov – 30th Nov**

  The following weeks I intend to spend reading up on DNN's doing some introductory courses, I will train the neural network on the same data as the paper but at this point will not yet try to measure the mutual information between the layers.

  At this point I will also start examining the code[8] provided and start to implement parts of it which don't deal with information measurement.

- **1st Dec – 28th Dec**

  Will start reading up on mutual Information measurement with local Gaussian approximation.

  Implementing mutual information measurement in code.

  At this point I expect the computation to be too demanding for my machine and will need to use provided compute.

- **29th Dec – 1st Feb**

  Having a working system to test data sets I will try to reproduce results from the paper on the same dataset. This will achieve my success criteria.

  At this point my success criteria should be completed I will spend some time writing the skeleton of the thesis. Look for any discrepancies between my results and the ones provided in the paper.

- **2nd Feb – 2nd Feb**

  Assuming everything goes as planned I will start looking into implementing one of the extensions. Which are :

  - Testing the theory on different datasets.
  - Implementing a quantized neural network implementation.

  or both, if time is in my favour.

- **3rd Feb – 2nd Mar**

  Will use the remaining time to write up the dissertation.

[7] Informaition Theory, Inference, and Learning Algorithms by David J. C. MacKay
[8] https://github.com/ravidziv/IDNNs