- Tishby produced a paper [**?**] claiming to understand the basic principles of how DNNs work.

- He decided to examine neural networks through the information domain, visualizing the results via Information Plane method section 0.3.

- Tishby made the claim that the incredible performance DNNs are able to achieve is due to the their ability to compress information inherent in the input data. Compressing data means the network is only able to keep relevant input features and it must forget the irrelevant bits of information, leading to the ability to generalize.

- Tishby made interesting claims and provided experimental evidence to support his claims, however he did not provide a formal proof leaving his results up for debate.

- Paper released by Saxe has contested the claims made by Tishby arguing that compression cannot happen in Neural Networks and the results are a consequence of the hyper parameters Tishby used.

- However Saxes problem suffers from the same problem as Tishby's as it does not provide a formal proof only experimental evidence, as such it doesn't settle the rebuttal.

- To fully understand the discussion we need to understand the following topics Entropy and Mutual Information, Neural Networks, and Information Plane, described in section 0.1, section 0.2, and section 0.3.

0.1 Entropy and Mutual Information

Entropy – quantifies information content of a random variable. It is generally measured in bits and can be though of as the expected information content when we sample a random variable once. Let X be a discrete random variable that can take values in $\{x_1, ..., x_n\}$. $H(X)$, the entropy of X, is defined by Equation 1.

$$H(X) = -\sum_{i=1}^{n} P(x_i) \log P(x_i) \tag{1}$$

Consider a random variable Y s.t $P(Y = 1) = P(Y = 0) = 0.5$, Equation 1 defines $H(Y)$ to be 1.

Similarly for a random variable Y s.t $P(Y = 0) = 0.5, P(Y = 1) = P(Y = 2) = 0.25$, we have $H(Y) = 1.5$.

Conditional Entropy – quantifies the amount of information needed to describe an outcome of variable Y given that value of another random variable X is already known. Conditional Entropy of Y given X is written as $H(Y|X)$. Let X be defined as before, Let Y be a discrete

random variable that can take values in $\{y_i, ..., y_n\}$. The conditional entropy $H(Y|X)$ is defined by Equation 2

$$H(Y|X) = - \sum_{x \in X, y \in Y} P(x,y) \log \frac{P(x,y)}{P(x)} \tag{2}$$

Let the correlated variables X and Y be defined by Table 1.

X \ Y	0	1
0	0.25	0.25
1	0.5	0

Table 1: Joint probability distribution for X and Y

Equation 1 and Equation 2 defines entropy values to be:

$$H(Y|X) = 0.5$$
$$H(X|Y) \approx 0.6887$$
$$H(X) = 1$$
$$H(Y) \approx 0.8112 \tag{3}$$

Mutual Information (MI) – measures how much information two random variables have in common. It quantifies information gained about one variable when observing the other. Equation 4 and Equation 5 are two examples of how we can compute mutual information, using explicit probability computations or entropies of the random variables respectively, here X and Y are as previously defined.

$$I(X,Y) = \sum_{y \in Y} \sum_{x \in X} P(x,y) \log \left(\frac{P(x,y)}{P(x)\,P(y)} \right) \tag{4}$$

$$I(X,Y) = H(X) - H(X|Y) \tag{5}$$

For example of mutual information consider the random variables X and Y as before in the conditional entropy section – defined by Table 1.

We computed the entropy values in Equation 3, so let us use them in Equation 5 to compute $I(X,Y)$.

$$I(X,Y) = H(X) - H(X|Y) \approx 1 - 0.6887 = 0.3113 \tag{6}$$

0.2 Neural Network

0.2.1 The Prediction problem

- A common task

- we have data that and we want to know to extract some property from the data

- such problems are hard to solve

- many examples

  - medical symptoms, diagnosis
  - picture, object in the picture
  - face, identity
  - stock market history, future stock prices
  - mail, spam or not spam

- we cannot construct an algorithm that always give the correct answer, sometimes it's impossible to know the correct answer with the knowledge we have.

- we turn to machine learning frameworks, which learn from data in some form or the other, make this a subparagraph

---

Neural Networks are machine learning frameworks intended to learn from data, it has been used effectively in predicting and classifying data.

Suppose we have a dataset:

$$(x_i, y_i) \text{ for } i = 1, ..., N, x_i \in \mathbb{R}^d, y_i \in \mathbb{R}^{d'}$$

where x_i is the input to the Neural Network and y_i is the label or the value we want the neural network to predict. Suppose we are trying to classify if an image contains a cat or no, then x_i would be an encoding of an image and the label y_i could be 1 if the image contains a cat and 0 if it does not.

The network is structured in layers where every layer holds an intermediate representation of the final prediction output, lets call this intermediate representation an **activation** of that specific layer. For our purposes we will define a neural network to be a sequence of functions $f_\theta^1, f_\theta^2, ..., f_\theta^N$ that are parameterized by weights θ s.t

$$\text{let } t_0 = x,$$
$$t_0 \to f_\theta^1(t_0) = t_1,$$
$$t_1 \to f_\theta^2(t_1) = t_2,$$
$$...$$
$$t_{N-1} \to f_\theta^N(t_{N-1}) = t_N,$$
$$\text{let } y = t_N$$

values $t_1, t_2, ..., t_N$ here are **activations** of layers $1, 2, ..., N$

We have one function for each layer transition of the neural network this allows us to extract the activation of a specific layer as in Figure 1, we will need to get access to intermediate layers in our experiments.

$$t_l = f_\theta^l(f_\theta^{l-1}(...(f_\theta^1(x))))$$

Figure 1: **activation** of layer l for input x

3

Stochastic Gradient Descent Each of the transition function are parameterized by the weights function θ. The θ function is directly controlled by Stochastic Gradient Descent (SGD) algorithm. SGD is an algorithm for training a neural network it gradually updates the weights θ according to some goal such as minimising the prediction error. SGD is an iterative process, it has a notion of epochs where one iteration of the algorithm advances epochs by one, this implies that θ function depends on which epoch we are currently at thus it must take the epoch number as an argument – $\theta(e)$.

0.3 The Information plane

0.3.1 Setup

The Information plane is a way of visualizing the Neural Networks training process through the information domain. By looking at mutual information between the input X the intermediate neural network layer activations T and the label Y we can see how the information flows through the network.

Mutual Information, as we defined it, is only applicable to Probability distributions, however we currently only have the datset (x_i, y_i) for $i = 1...N$, if we assume that every input x_i equally likely we can provide a routine that defines our probability distribution.

For convenience lets define F_θ^t to be the activation of layer t given input x, i.e

$$F_\theta^t(x) = f_\theta^t(f_\theta^{t-1}(...(f_\theta^1(x)))) \tag{7}$$

Consider Figure 2, the routine defines the random variables $X, T_{e,t}, Y$. Here $T_{e,t}$ is the distribution of layer t for the epoch e, X and Y is the original data with assumption that it is uniformly distributed.

Using the probability distributions we now have values:

- $I(T_{e,t}, X)$ – Mutual Information between the input distribution and the layer activations

- $I(T_{e,t}, Y)$ – Mutual Information between the label distribution and the layer activations.

This allows us to generate the Information plane.

```
1  def rxty(e, t):
2      pick i ∼ Uniform {1...N}
3      return (x_i, F_{θ(e)}^t(x_i), y_i)
```

Figure 2: Definition of correlated random variables $X, T_{e,t}$ and, Y

0.3.2 Visualization

The Information plane visualizes the whole training process, in order to generate the information plane we need $I(X, T_{e,t})$ and $I(Y, T_{e,t})$ for every epoch e and every layer t.

Consider for now that we only trained our neural network for one epoch, the Figure 3 shows an example of this.

In this case the network consists of 5 layers, in the figure every node corresponds to a distinct layer. The lines between the nodes help us distinguish epochs from each other and gives helps us to see the order of the layers, the upper-right-most node corresponds to the first layer in the neural network, the lower-left-most node corresponds to the fifth and last layer of the network.
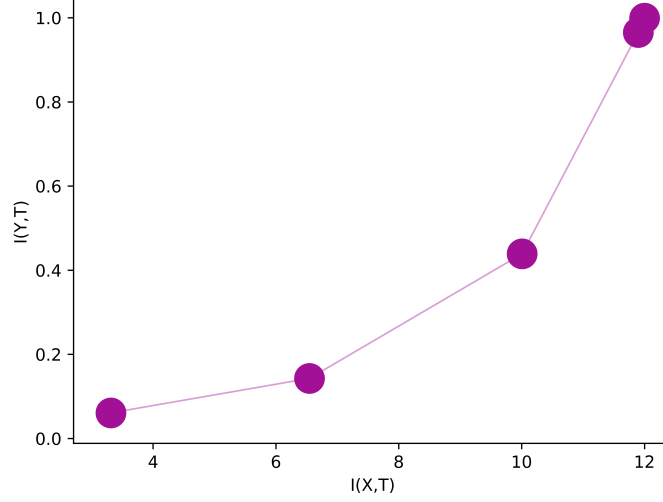

Figure 3: information plane for a neural network with 5 layers, which was only trained for one epoch.

Consider now Figure 4, it shows an information plane for a full training phase. The color signifies what epoch the data belongs to and let us see how the network progressed over time. We can see that at the start $I(Y, T_{1,4}) \approx 0$ meaning the network has not preserved any information about the label distribution Y, but by the end of the training we see $I(Y, T_{10^4,4}) \approx 1$ which means we have preserved almost all the information about the label.
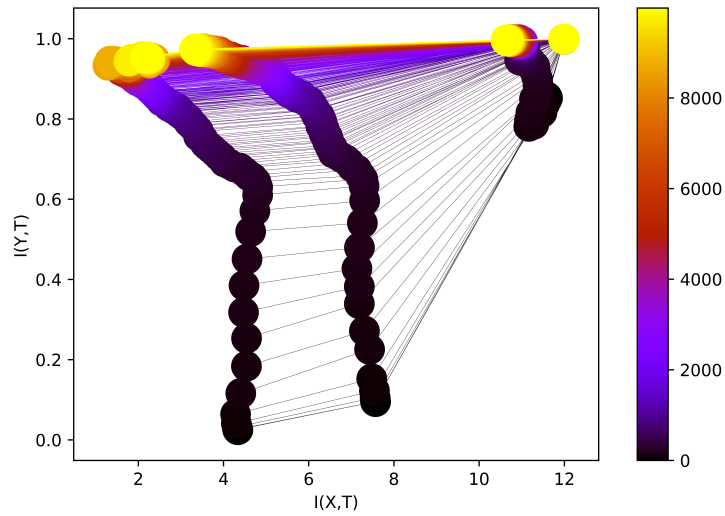

Figure 4: Information plane for a neural network with 4 layers, which was trained for approximately 10 000 epochs. The Neural Network was trained on the same dataset as used by Tishby, 2017, hence entropy of the input $H(X)$ is 12 and entropy of the label $H(Y)$ is 1.

0.3.3   Interpretation of the Information Plane

Let us once again consider Figure 4, we can see two phases in the figure Tishby has named them The Fitting Phase and The Compression Phase.

The Fitting Phase   In Figure 4 the neural network is in the fitting phase from the start of the training up until epoch ∼1500. The duration of the fitting phase varies heavily on the training parameters and is most influenced by the size of our input dataset. The fitting phase is characterized by:

- A rapid increase in $I(Y, T_{e,t})$, the information about the label, as we advance through the epochs e, the increase is especially visible in the later layers, in our case layers 3 and 4.

- Either an increase or no change in $I(X, T_{e,t})$, the information about the input, as we advance through the epochs e, in our case we see very little change in $I(X, T_{e,t})$.

During the fitting phase a neural network tries to memorize the data and make predictions based on the observations, this means that the network may learn useless features that only superficially correlate with the correct label.

The Compression Phase   In Figure 4 the neural network enters the compression phase when the fitting phase ends around epoch ∼1500 and lasts until we finish the training process. The compression phase is characterized by:

- A slowdown of how fast $I(Y, T_{e,t})$ is increasing with respect to epochs e.

- A slow decrease of $I(X, T_{e,t})$ with respect to epoch e.

During the compression phase a neural network compresses representation of the input discarding more features that did not help with predicting correct labels. Discarding irrelevant features helps the neural network generalize and produce better predictions for new data.

—————————————————————————————

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.