Information Compression in Simple Neural Networks

Nathan C. L. Kong
Stanford University
nclkong@stanford.edu

March 24, 2019

Abstract

In the past decade, high throughput training of deep learning models has bolstered their success in completing difficult tasks. Unfortunately, a theoretical understanding of why these models are so successful is missing. Some work investigating why deep learning models generalize so well utilize concepts from information theory and analyze the information gain between the inputs (and outputs) and the internal representations. A problem that arose in this kind of approach was related to how mutual information was computed in deterministic neural networks. Goldfeld et al. (2018) developed a new method for estimating mutual information by analyzing information theoretic quantities using noisy neural networks and observed that the reduction in mutual information between the internal representation and the inputs (compression) is associated with the clustering of internal representations.

In this work, we reproduce some simple empirical observations in Goldfeld et al. (2018). Furthermore, we conduct some experiments related to modifying the data distribution, as previous work studying information flow in neural networks used a uniform input data distribution. We observe that for a single Gaussian data distribution, using a non-saturating non-linearity in the hidden layer such as LeakyReLU, we do not observe a clustering of the internal representations.

1 Introduction

Deep neural networks and their optimization algorithms have been shown to be extremely successful in a variety of complex tasks including object categorization (Szegedy et al. (2015)), image segmentation (Long et al. (2015)) and speech recognition (Graves et al. (2013)). Unfortunately, a theoretical understanding of why neural networks with millions of parameters can generalize so well on a variety of tasks is arguably non-existent. There is, however, some work that analyzes generalization capabilities using Rademacher complexity (such as Golowich et al. (2018), Neyshabur et al. (2017)). In this work, we focus on another method that uses information theoretic quantities to analyze neural networks.

Previous studies such as Shwartz-Ziv and Tishby (2017) and Saxe et al. (2018) investigate the evolution of hidden layer representations in neural networks by analyzing the information gain between the hidden layers and the inputs and outputs. Notably, Shwartz-Ziv and Tishby (2017) observe that when a neural network is trained on a binary classification task, there are two distinct phases in the changes of mutual information over training epochs. In the first phase, termed the "fitting phase", the mutual information between the inputs and the intermediate layer representations increases. The next phase that was proposed by the authors is known as the "representation compression" phase, where the mutual information between the inputs and the intermediate layer representations decrease. This phase was observed to be much longer than the fitting phase in terms of training epochs and the authors suggested it was during this second phase that parts of the representation not used for the classification task were eliminated.

Some technical issues arose in work studying neural network generalization from an information theoretic point of view. These issues relate to the estimation of mutual information in deterministic neural networks. In deterministic neural networks, the hidden layer representations are deterministic functions of the inputs. Thus, the conditional entropy of a hidden layer representation given the input is zero (i.e. $H(f(X) \mid X) = 0$, where f is a deterministic transformation of the input through layers of a neural network and X is the input). Since the conditional entropy is zero, the mutual information between the inputs and hidden layer representations should be constant (equal to the entropy of the inputs or the hidden representations), which is the opposite of the previous observations that the mutual information between internal representations and the input varies over training epochs.

Goldfeld et al. (2018) proposed training noisy neural networks in order to rectify the issue with computing mutual information in a deterministic setting. It was shown that noisy neural network performance is not drastically affected by the injected (independent, Gaussian) noise (as long as the "signal to noise" ratio is not too small). In addition, the authors suggest that training with injected noise at each layer can be related to the improved generalization ability of the neural network.

The report is organized as follows. In Section 2, we summarize the mutual information estimator and how mutual information is computed. We also summarize the simple neural networks we use and the input data distributions that the networks were trained on. The results and observations are presented in Section 3. We conclude and present future directions in Section 4.

2 Methods

The code can be found at `https://github.com/nathankong/information_bottleneck`.

2.1 Noisy Neural Networks

In all the models, Gaussian noise is injected after the point-wise non-linearity is applied at each layer. Noise is sampled according to $\mathcal{N}(0, \beta^2)$, where $\beta = 0.05$. The schematic for each layer is shown in Figure 1.

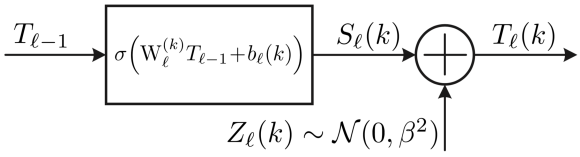


Figure 1: Flow diagram through a single neuron. σ is the non-linearity, $W_\ell^{(k)}$ is the weight matrix and b_ℓ is the bias for layer ℓ and $Z_\ell^{(k)}$ is the spherical Gaussian noise injected at each layer at epoch k. Adapted from Goldfeld et al. (2018).

2.2 Mutual Information

Given two continuous random variables, X and Y, and the joint distribution defined as $p_{X,Y}$, the mutual information between them is defined as:

$$
\begin{aligned}
I(X;Y) &\triangleq \int_{x\in\mathcal{X}} \int_{y\in\mathcal{Y}} p_{X,Y}(x,y) \log \frac{p_{X,Y}(x,y)}{p_X(x)p_Y(y)} \,\mathrm{d}y \,\mathrm{d}x \\
&= \int_{x\in\mathcal{X}} p_X(x) \int_{y\in\mathcal{Y}} p_{Y|X}(y \mid x) \log \frac{p_{Y|X}(y \mid x)}{p_Y(y)} \,\mathrm{d}y \,\mathrm{d}x
\end{aligned}
$$

where $(x, y) \in \mathcal{X} \times \mathcal{Y}$. For another continuous random variable, Z, and assuming that it is distributed according to p_Z, the differential entropy of Z is defined as:

$$h(p_Z) \triangleq -\int_{z \in \mathcal{Z}} p_Z(z) \log p_Z(z) \, \mathrm{d}z$$

Using the expression for differential entropy, the mutual information can be rewritten as:

$$\begin{aligned} I(X;Y) &= h(p_Y) - h(p_{Y|X}) \\ &= h(p_Y) - \int_{x \in \mathcal{X}} p_X(x) h(p_{Y|X=x}) \, \mathrm{d}x \\ &= h(p_Y) - \mathbb{E}_{X \sim p_X} \left[h(p_{Y|X=x}) \right] \end{aligned}$$

Note that when the distribution of X is continuous, we can approximate the expectation (the second term above) using Monte Carlo sampling. Concretely, we can approximate the expectation using N samples from p_X (where $x^{(i)}$ is the i-th sample) as follows:

$$\mathbb{E}_{X \sim p_X} \left[h(p_{Y|X=x}) \right] \approx \frac{1}{N} \sum_{i=1}^{N} h\left(p_{Y|X=x^{(i)}} \right)$$

2.2.1 Mutual Information Estimator

The mutual information estimator used in this work was developed in Goldfeld et al. (2018). We briefly summarize it here using notation shown in Figure 1. Since $T_\ell = S_\ell + Z_\ell$, we have that $p_{T_\ell} = p_{S_\ell} * \varphi$, where φ is a Gaussian distribution. We can estimate p_{S_ℓ} empirically by computing the outputs at a layer before noise is injected (i.e. S_ℓ) using n samples from the data distribution. The n samples define $\hat{p}_{S_\ell}$. Thus, p_{T_ℓ} is estimated as:

$$\begin{aligned} p_{T_\ell}(x) &= \left(p_{S_\ell} * \varphi \right)(x) \approx \left(\hat{p}_{S_\ell} * \varphi \right)(x) \\ &= \int \hat{p}_{S_\ell}(y) \varphi(x - y) \, \mathrm{d}y \\ &= \int \frac{1}{n} \sum_{j=1}^{n} \delta\left(y - x_j\right) \varphi(x - y) \, \mathrm{d}y \\ &= \frac{1}{n} \sum_{j=1}^{n} \varphi\left(x - x_j\right) \end{aligned}$$

where $\delta(r) = 1$ if $r = 0$, and 0 otherwise and x_j is the j-th output at a particular layer.

Similarly, in order to estimate the conditional entropy of an internal layer given a particular input, we need to estimate $p_{T_\ell|X=x_i}$. To do this, we similarly compute the convolution of the output before noise is injected with the noise distribution. Concretely:

$$p_{T_\ell|X=x_i} = p_{S_\ell|X=x_i} * \varphi$$

The conditional distribution, $p_{S_\ell|X=x^{(i)}}$, is estimated by first sampling x_i from p_X and then repeatedly fed into the neural network n_i times. The n_i outputs define the empirical distribution $\hat{p}_{S_\ell|X=x^{(i)}}$. Thus, we can estimate $p_{T_\ell|X=x^{(i)}}$ as follows:

$$p_{T_\ell|X=x^{(i)}}(y) \approx \hat{p}_{S_\ell^{(i)}}(y) = \frac{1}{n_i} \sum_{j=1}^{n_i} \varphi\left(y - x_j^{(i)}\right)$$

Finally, with the expressions for conditional and unconditional entropy, we can compute the mutual information between the input and a hidden layer representation.

$$\begin{aligned} I(X;T_\ell) &= H(T_\ell) - H(T_\ell \mid X) \\ &\approx [h(\hat{p}_{S_\ell} * \varphi)] - \mathbb{E}_{x\sim p_X}\left[h\left(\hat{p}_{S_\ell^{(x)}} * \varphi\right)\right] \\ &\approx h(\hat{p}_{S_\ell} * \varphi) - \frac{1}{N}\sum_{j=1}^{N}\left[h\left(\hat{p}_{S_\ell^{(x_j)}} * \varphi\right)\right] \end{aligned}$$

where N is the number of Monte Carlo samples to use and x_j is the j-th Monte Carlo sample from p_X.

2.3 Simple Neural Networks

The training dynamics for binary classification tasks (output is either $+1$ or -1) and simple neural networks were analyzed. Each network was trained using stochastic gradient descent. The leaky ReLU non-linearity is defined as: $\text{LeakyReLU}(x) = \max\{x, x/10\}$. z_1 and z_2 are noise samples from $\mathcal{N}(0, 0.05^2)$. The neural network models are described below.

1. Single Neuron Tanh Network

   This network is defined as: $y = \tanh(wx + b)$. The weight and bias was initialized to 0.

2. Two Neuron LeakyReLU–LeakyReLU Network

   This network is defined as: $y = \text{LeakyReLU}\left(w_2 \cdot (\text{LeakyReLU}(w_1 \cdot x + b_1) + z_1) + b_2\right) + z_2$. The weight initializations were as follows: $w_1 = 0, b_1 = 4.5, w_2 = -1, b_2 = 0$.

3. Two Neuron Tanh–Tanh Network

   This network is defined as: $y = \tanh\left(w_2 \cdot (\tanh(w_1 \cdot x + b_1) + z_1) + b_2\right) + z_2$. The weights and biases were initialized to 0.

4. Two Neuron LeakyReLU–Tanh Network

   This network is defined as: $y = \tanh\left(w_2 \cdot (\text{LeakyReLU}(w_1 \cdot x + b_1) + z_1) + b_2\right) + z_2$. The weights and biases were initialized to 0.

Networks 1 and 2 were used in Goldfeld et al. (2018) as simple examples to show the empirical behaviour of mutual information evolution over training epochs.

2.4 Data Distributions

Three different data distributions were used in the experiments. Refer to the previous section for the numbering of the network models.

1. Uniform distribution

   Two different datasets were used in this scenario. Both were also used in Goldfeld et al. (2018) as simple examples. The first dataset consisted of four values: $\{-3, -1, 1, 3\}$. $\{-3, -1, 1\}$ were labelled as -1 and $\{1\}$ was labelled as $+1$. The second dataset consisted of eight values: $\{1, 2, 3, 4, 5, 6, 7, 8\}$. $\{1, 2, 3, 4\}$ were labelled as 0 and $\{5, 6, 7, 8\}$ were labelled as 0.25. We used $N = 100$ Monte Carlo samples for the dataset with four values and $N = 200$ for the dataset with eight values.

   The four value dataset was used to train network 1 with learning rate 0.0005. The eight value dataset was used to train network 2 with learning rate 0.0001 and decreased by a factor of 20 when the test set accuracy was at least 0.99.

2. Univariate Gaussian distribution

   The data were generated from a univariate Gaussian distribution with mean 0 and variance 0.25 (i.e. $X \sim \mathcal{N}(0, 0.25)$). Generated values greater than 0 had labels of $+1$ and values less than 0 had labels of -1. We used $N = 500$ Monte Carlo samples for this data distribution.

   This dataset was used to train networks 3 and 4. Both networks were trained with a learning rate of 0.005, which was reduced by a factor of 20 when the test set accuracy was at least 0.99.

3. Univariate Gaussian mixture distribution

   The data were generated from an equal mixture of two Gaussian distributions: $\mathcal{N}(-1, 0.25)$ and $\mathcal{N}(1, 0.25)$. Generated values greater than 0 had labels of $+1$ and values less than 0 had labels of -1. We used $N = 500$ Monte Carlo samples for this data distribution.

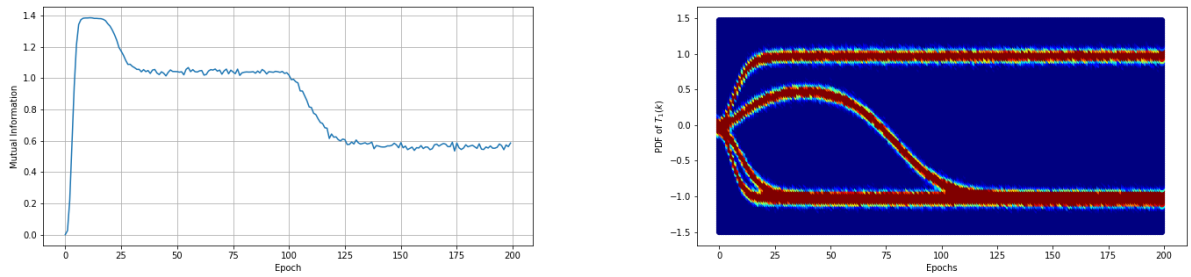
   This dataset was used to train network 1 with learning rate 0.0001. The learning rate was decreased by a factor of 20 when the test set accuracy was at least 0.99.

Dataset 1 (both the four and eight value datasets) were used in Goldfeld et al. (2018) to show the clustering behaviour of the hidden layer and output representations and the changes in mutual information as a function of training epochs.

3 Results

3.1 Uniform Data Distribution

The role of this set of experiments was to reproduce some key phenomena that were presented in Goldfeld et al. (2018). Figure 2 shows how the mutual information between the output and input changes as a function of the training epoch for network 1. We can observe that the mutual information decreases at two points in the training. The first time it decreases is when outputs cluster from four to three clusters (around epoch 25). The second time it decreases occurs when the outputs cluster from three to two clusters (around epoch 100), where each cluster corresponds to the classification label (i.e. $+1$ or -1). This qualitatively recapitulates observations found in Goldfeld et al. (2018) (cf. Figure 5 in Goldfeld et al. (2018)).



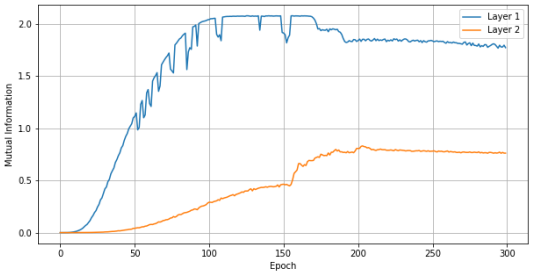
(a) Evolution of mutual information between T_1 and the inputs over training epochs.

(b) Evolution of the PDF of $T_1(x)$ as a function of the training epoch.

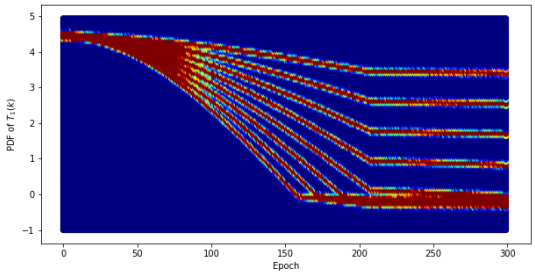
Figure 2: Observations from network 1 trained on the four value dataset.

Next, we trained network 2 to perform binary classification on the eight value dataset. Recall that this network utilizes a non-saturating non-linearity. Figure 3 shows how mutual information between hidden layers and the inputs and how the distribution at each hidden layer evolves over training epochs. We similarly observe a clustering of the inputs at the hidden layers and at the outputs. In the hidden

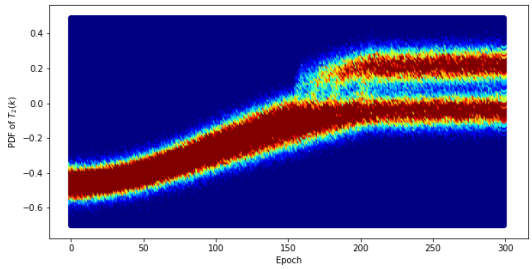
layer, we see that mutual information increases to its maximum when the inputs separate into the eight clusters (see Figure 3b). It starts to decrease when the inputs start to cluster (starting at around epoch 150). At the output, the inputs separate into two clusters corresponding the labels of the dataset, which are 0 and 0.25 (see Figure 3c). When this occurs, the mutual information between the outputs and the inputs increases (around epoch 150).



(a) Evolution of mutual information between T_1 and T_2 and the inputs over training epochs.



(b) Evolution of the PDF of $T_1(x)$ as a function of the training epoch.



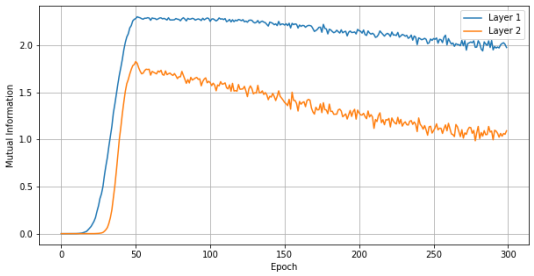
(c) Evolution of the PDF of $T_2(x)$ as a function of the training epoch.

Figure 3: Observations from network 2 trained on the eight value dataset.

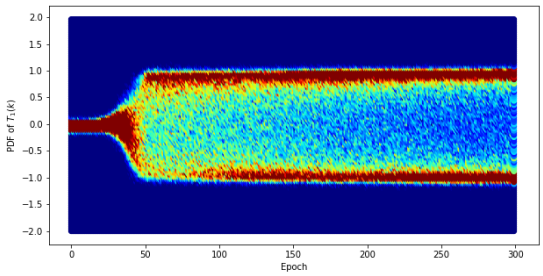
3.2 Univariate Gaussian Data Distribution

We trained two different two neuron network models on this data. The difference between the two models is the non-linearity used (tanh vs. LeakyReLU). We sought to understand how changes to the hidden layer non-linearity would affect internal representations and mutual information. Figure 4 shows how mutual information and the distribution of internal/output representations evolves over training epochs for the two neuron Tanh–Tanh network. The observation that mutual information between the inputs and hidden layer/output representations decreases with the clustering of representations is recapitulated. Note that the mutual information between the output layer and the inputs is less than that between the inputs and hidden layers, which is expected due to the stronger clustering in the output representations.

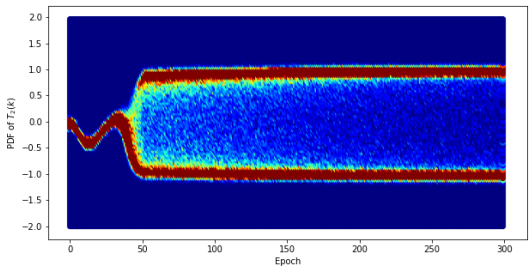
Figure 5 shows how mutual information and the distribution of internal/output representations evolves over training epochs for the two neuron LeakyReLU–Tanh network. Focussing on Figure 5a we notice that the mutual information between the inputs and the hidden layer representations continues to increase as training progresses. This is the opposite of what was observed in the previous experiments where the mutual information would decrease as a result of the representations clustering. Focussing on Figure 5b, we observe that the distribution of the internal layer representations becomes more diffuse as training progresses.



(a) Evolution of mutual information between T_1 and T_2 and the inputs over training epochs.

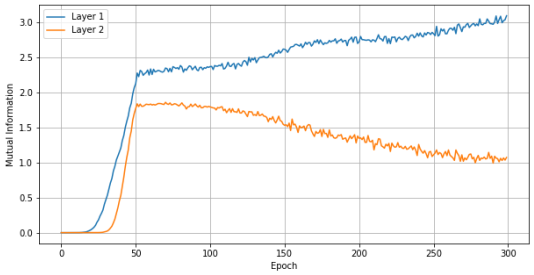


(b) Evolution of the PDF of $T_1(x)$ as a function of the training epoch.

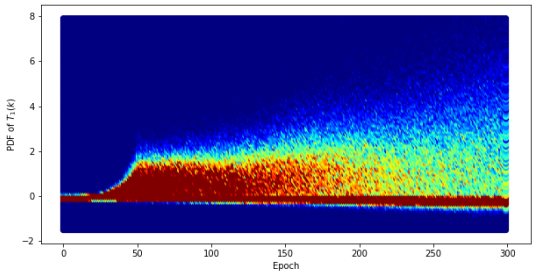


(c) Evolution of the PDF of $T_2(x)$ as a function of the training epoch.

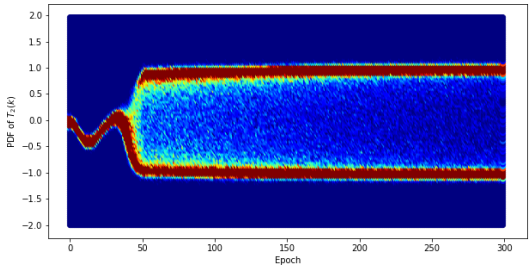
Figure 4: Observations from network 3 trained on the single Gaussian dataset. (i.e. tanh non-linearity)



(a) Evolution of mutual information between T_1 and T_2 and the inputs over training epochs.



(b) Evolution of the PDF of $T_1(x)$ as a function of the training epoch.

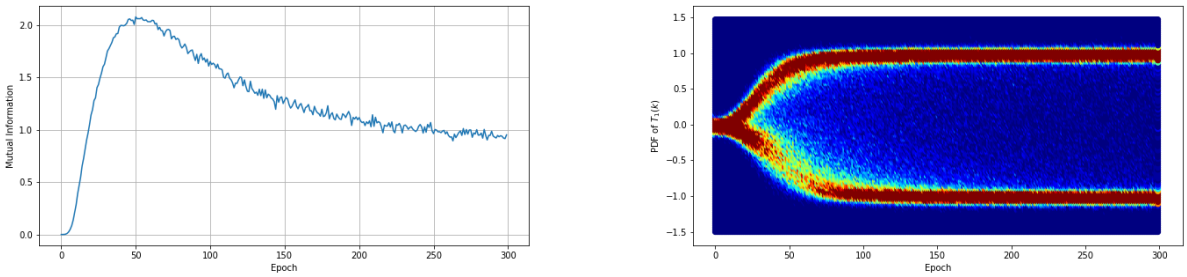


(c) Evolution of the PDF of $T_2(x)$ as a function of the training epoch.

Figure 5: Observations from network 4 trained on the single Gaussian dataset. (i.e. LeakyReLU non-linearity)

3.3 Univariate Gaussian Mixture Data Distribution

Figure 6 shows the evolution of mutual information and the distribution of the output when the single neuron network (i.e. network 1) is trained to perform binary classification on data distributed according to an equal mixture of two Gaussians. We observe that when the outputs first split into two clusters, the mutual information increases to its maximum. However, as the outputs slowly cluster to the labels (-1 and $+1$), the mutual information decreases and converges to approximately 1 nat. This reduction in mutual information is intuitive since we see that the distribution of outputs becomes less "noisy" (i.e. the distribution evolves to become more similar to point masses at $+1$ and -1) as training progresses.



(a) Evolution of mutual information between T_1 and the inputs over training epochs.

(b) Evolution of the PDF of $T_1(x)$ as a function of the training epoch.

Figure 6: Observations from network 1 trained on the mixture of Gaussians dataset.

4 Discussion and Conclusions

Part of the work performed in this study replicated empirical observations found in Goldfeld et al. (2018). The hidden layer and output distributions from our experiments (in particular, the four value dataset and the eight value dataset) also exhibit the clustering phenomenon and the associated reduction in mutual information. However, when we utilized a single univariate Gaussian data distribution and a network with a non-saturating non-linearity, we do not see a clustering of the hidden layer representations. Furthermore, we do not observe the reduction in mutual information. In fact, it appears to continue increasing as a function of training epoch. This observation is interesting and warrants further experiments including a uniform input data distribution with continuous values (i.e. instead of a discrete dataset). Future work also includes training a recurrent neural network to perform a binary classification task and analyzing the evolution of mutual information between the inputs and hidden representations.

References

Ziv Goldfeld, Ewout van den Berg, Kristjan Greenewald, Igor Melnyk, Nam Nguyen, Brian Kingsbury, and Yury Polyanskiy. Estimating information flow in neural networks. *arXiv preprint arXiv:1810.05728*, 2018.

Noah Golowich, Alexander Rakhlin, and Ohad Shamir. Size-independent sample complexity of neural networks. In Sébastien Bubeck, Vianney Perchet, and Philippe Rigollet, editors, *Proceedings of the 31st Conference On Learning Theory*, volume 75 of *Proceedings of Machine Learning Research*, pages 297–299. PMLR, 06–09 Jul 2018.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. IEEE, 2013.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

Behnam Neyshabur, Srinadh Bhojanapalli, David McAllester, and Nati Srebro. Exploring generalization in deep learning. In *Advances in Neural Information Processing Systems*, pages 5947–5956, 2017.

Andrew Michael Saxe, Yamini Bansal, Joel Dapello, Madhu Advani, Artemy Kolchinsky, Brendan Daniel Tracey, and David Daniel Cox. On the information bottleneck theory of deep learning. In *International Conference on Learning Representations*, 2018.

Ravid Shwartz-Ziv and Naftali Tishby. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, 2017.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.