Understanding Generalization and Memorization in Neural Networks

Nikitha Shravan Manisha Padala Shashank Mishra

May 5, 2018

Abstract

In this project, our primary aim is to investigate into issue of generalizability in neural networks. Through experiments, we show that (i) explicit and implicit regularizers are neither sufficient nor necessary to prevent over-fitting. (ii) Deep architectures incorporate good generic priors and hence tend to prioritize learning simple patterns in the data as opposed to memorizing each sample. We mention briefly about how information bottleneck principle explains the role of sequential processing of data and hidden layers in generalization.

1 Introduction

As elaborated in Zhang *et. al.* [3] most deep networks have far more trainable parameters compared to the number of input samples. In such a scenario, one might expect the model to memorize each sample. Generalization error (GE) is the primary measure to estimate such over-fitting in models. A model is said to have high generalization error if the difference between the train and test error is high indicating that the model generalizes poorly. As of now, there has been no fundamental explanation for the generalization observed in the neural network models. A good explanation to generalizability of neural networks would not only help neural networks make more interpretable but it will also help in designing better architectures.

2 Major Analysis

In this section we elaborate on the three main concepts that we have analyzed in our project. The following are supported either through experiments or are mathematically proven.

2.1 Generalizability Vs Memorization

Generalizability error is defined as the difference between train and test accuracy of a model. Higher the GE error, implies more over-fitting in the model. In a sense, over-fitting indicates how much the network has memorized the training samples. Deep Networks are known to be universal approximators. Owing to the large number of parameters in a deep model, the capacity of a neural network is sufficient for memorizing the entire dataset. This usually is referred to as the representational capacity and does not rule out memorization.

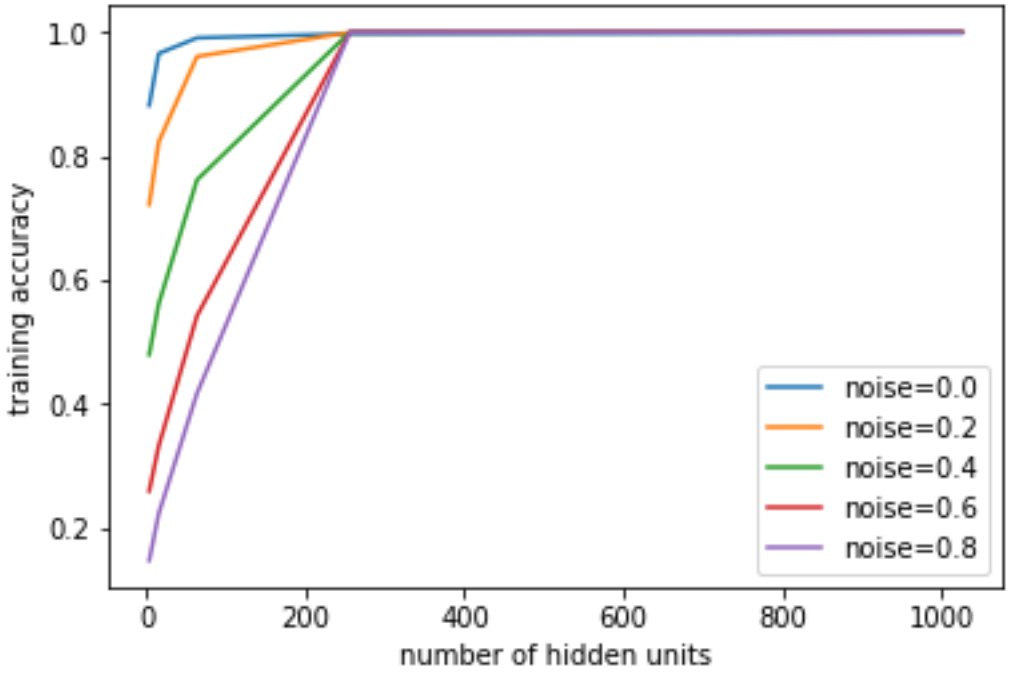
Definition 1 (Representational Capacity) *The set of hypotheses a model is capable of expressing via some value of its parameters.*

Now let us consider the following two cases, where on hand we have a normal dataset, and on the other hand we have the same dataset where the labels have been randomized.

Data - True labels	Data - Random labels
GE = low	GE = high
Train Error = low	Train Error = low
Test Error = low	Test Error = high
Train Time = less	Train time = high

Table 1: Comparing Two Cases

Given Table 1, it is for us to ponder why the same model memorizes the dataset with true labels but does not do so when the data set are random. As given in Arpit *et. al.* [1], just the representational capacity could not have accounted for such behavior hence we needed a new measure for the models which also takes into account the dataset on which the model has been trained



Training accuracy always reaches 1 with sufficient capacity

Definition 2 (Effective Capacity)

$$EC(\mathcal{A}) = \{h \mid \exists\ \mathcal{D},\ h \in \mathcal{A}(\mathcal{D})\}$$

where $\mathcal{A}(\mathcal{D})$ is the set of hypothesis that is reachable by an algorithm $\mathcal{A}$ on a dataset $\mathcal{D}$

The above definition points out that irrespective of the architecture, generalization in neural networks is inherently captured based on the kind of data encountered, realistic data vs complete random one.

The major claim is that, although neural networks easily fit random labels they tend to prioritize learning simple patterns.

One part of the claim stating that neural networks learn simple patterns has been experimentally verified using two proxy measures in [1]

- **Loss Sensitivity**: Given $\mathcal{L}_t$ is the loss after t updates, the sensitivity measure is given by

  $$g_x^t = \| \partial L_t / \partial x \|_1$$

  $\bar{g}_x$ is the average over g_x^t after T time steps. It is observed that only a subset of training data has high values for $\bar{g}_x$, while for random data, this value is high for all examples.

- **Critical Sample Ratio (CSR)**: Critical Sample is a subset of data with an adversarial example in its proximity. $x \in \mathbb{R}^k$, is a critical sample when $\hat{x}$ the adversarial example is in its proximity (box size r), networks output vector $f(x) = (f_1(x) \cdots f_k(x)) \in \mathbb{R}^k$

  $$\underset{i}{argmax}\ f_i(x) \neq \underset{j}{argmax}\ f_j(\hat{x})$$

  $$s.t.\ \| x - \hat{x} \|_\infty \leq r$$

  $$CSR = \frac{\#criticalsamples}{\#datapoints}$$

  It is observed that the number of critical samples is much higher for noisy data than realistic data. This indicates the need for more complex hypothesis in a noisy data.

The other part of the claim which states that neural networks can be easily fit for random labellings has been supported theoretically using **Finite sample expressivity** in [3]. This is the expressive power of neural networks in a finite sample of size n. It s

Theorem 1 *There exists a two-layer neural network with ReLU activations and $2n + d$ weights that can represent any function on a sample of size n in d dimensions*

Proof

Lemma 1. For any two interleaving sequences of n real numbers $b_1 < x_1 < b_2 < x_2 \cdots < b_n < x_n$, the $n \times n$ matrix $A = [\max\{x_i - b_j, 0\}]_{ij}$ has full rank. Its smallest eigen value is $\min_i x_i - b_i$

For weight vectors $w, b \in \mathbb{R}^n$, $a \in \mathbb{R}^d$ consider a function

$c : \mathbb{R}^n \to \mathbb{R}$ $\quad c(x) = \sum_{j=1} w_j \max\{\langle a, x\rangle - b_j, 0\}$

c can be expressed as a 2 layer NN with ReLU

Let $S = \{z_1, \ldots z_n\}$ of size n and a target vector $y \in \mathbb{R}^n$.
We need to find a, b, w so that $y_i = c(z_i)$ for all $i \in \{1 \ldots n\}$

Chose a, b such that $x_i = \langle a, z_i \rangle$ we have
$b_1 < x_1 < b_2 < x_2 \ldots b_n < x_n$. This is possible since all my z_i's are different.

Consider n equations in n unknowns w,

$$y_i = c(z_i), \quad i \in \{1, \ldots n\}$$

$$y = \begin{Bmatrix} c(z_1) \\ \vdots \\ c(z_n) \end{Bmatrix} = Aw$$

$\longrightarrow$ Full Rank and invertible

We can solve to find suitable w.

$\Rightarrow$ Assume $x_1 \ldots x_n \in [0, 1]$. Partition into b disjoint intervals $I_1 \ldots I_b$ so that each interval contains I_j contains n/b.
At layer j apply the construction from the proof. This requires $O(n/b)$ nodes and $(b+1)$ depth.

This results in a very wide neural network but it can be shown that,

Corollary 1 *For every $k \geq 2$, there exists neural network with ReLU activations of depth k, width $O(n/k)$ and $O(n+d)$ weights that can represent any function on a sample of size n in d dimensions.*

model	# params	random crop	weight decay	train accuracy	test accuracy
Inception	1,649,402	yes	yes	100.0	89.05
		yes	no	100.0	89.31
		no	yes	100.0	86.03
		no	no	100.0	85.75
(fitting random labels)		no	no	100.0	9.78
Inception w/o BatchNorm	1,649,402	no	yes	100.0	83.00
		no	no	100.0	82.00
(fitting random labels)		no	no	100.0	10.12
Alexnet	1,387,786	yes	yes	99.90	81.22
		yes	no	99.82	79.66
		no	yes	100.0	77.36
		no	no	100.0	76.07
(fitting random labels)		no	no	99.82	9.86
MLP 3x512	1,735,178	no	yes	100.0	53.35
		no	no	100.0	52.39
(fitting random labels)		no	no	100.0	10.48
MLP 1x512	1,209,866	no	yes	99.80	50.39
		no	no	100.0	50.51
(fitting random labels)		no	no	99.34	10.61

Figure 1: Explicit regularizers

2.2 Role of Regularizers

In this subsection, we analyze if regularizers are actually helpful in generalization of models. Regularizers are external factors other than the model architecture itself, helping the network to generalize. There are two possible kinds of regularizers explicit and implicit.

- *Explicit Regularizers*: From the following experiments Figure 1 by Zhang et al [3] we can clearly see that explicit regularizers like Weight Decay and Random crop do not add much to the generalizability of the models. Further we have various experiments studying the effect of regularizers with different noise levels on the input data. As mentioned in [3] they are not the fundamental cause of generalizability.

- *Implicit Regularization*: Does SGD converge to the minimum norm solution ? Consider linear model and n distinct data points (x_i, y_i) where x_i are d-dimensional. The following risk has to be minimized

$$min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} loss(w^T x_i, y_i)$$

If $d \geq n$, then any labeling can be fit. What about generalization? Let X denote the $n \times d$ data matrix whose i^{th} row is x_i^T. If X has rank n, then the system of equations $Xw = y$ has an infinite number of solutions. So is it possible to converge to one unique global optima ? Looking into SGD algorithm, $w_{t+1} = w_t - \eta_t e_t x_{i_t}$, if $w_0 = 0$, then optimal w has the form $w = \sum_{i=1}^{n} \alpha_i x_i$. Hence, if SGD gives $w = X^T \alpha$, that is the w lies in the span of the data points. This along with $Xw = y$ gives us,

$$XX^T \alpha = y$$

Here α has unique solution. So therefore, any set of labels can be fit by forming the Gram matrix $K = XX^T$ and solving $K\alpha = y$. Huge size of Kernel Matrix!

The Kernel solution is the minimum l2-norm solution of $Xw = y$. That is, out of all models that exactly fit the data, SGD will often converge to the solution with minimum norm.

Proof 1

$$\min_w \parallel w \parallel_2^2$$

s.t

$$y = Xw$$

Using Lagrangian,

$$\min_w (\parallel w \parallel_2^2 + \lambda \parallel y - Xw \parallel_2^2)$$

First order condition,

$$w(I + \lambda X^T X) = \lambda X^T y$$

As $\lambda \to \infty$

$$w^* = (X^T X)^{-1} X^T y$$

Using the kernel solution $y = XX^T \alpha$ *we get,*

$$w^* = (X^T X)^{-1} X^T X X^T \alpha = X^T \alpha = w(SGD)$$

It is very easy to construct solutions of $Xw = y$ that don't generalize: for example, one could fit a Gaussian kernel to data and place the centers at random points. Another simple example would be to force the data to fit random labels on the test data. In both cases, the norm of the solution is significantly larger than the minimum norm solution.

2.3 IB principle's Explanation for Generalization in Deep Neural Networks (DNN)

Tishby *et. al.* [2] venture into quantifying the DNN using mutual information between the layers and the input and the output variables. X the input to a network being very high dimansional, is not entirely informative about Y the output. Hence we need only the relevant features from X which are highly distributed, to predict Y. The remarkable success of DNNs in learning to extract relevant features is mainly attributed to the sequential processing of the data, namely that each hidden layer operates as the input to the next one, which allows the construction of higher level distributed representations.

Information Bottleneck Principle (IBP)

The principle defines the concept of extracting relevant information that an input random variable $X \in \mathcal{X}$ contains about $Y \in \mathcal{Y}$ assuming Y is dependent on X.

Definition 3 (Minimal Sufficient Statistics) *$\hat{X}$ is a minimal sufficient statistics of X with respect to Y, if it is the simplest mapping of X that captures the mutual information $I(X;Y)$*

It is desired that $\hat{X}$ compresses X as much as possible at the same time keeping the relevant information about Y instact, hence is given by the minimization of the following Lagrangian,

$$\mathcal{L}[p(\hat{x}/x)] = I(X;\hat{X}) - \beta I(\hat{X};Y)$$

The optimal solutions follow the following self-consistent equations for some value of β

$$p(\hat{x}/x)$$

$$p(y/\hat{x})$$

$$p(\hat{x})$$

Information characteristics of the layers

In a DNN the network layers form a Markov Chain. The information about Y that is lost in the one layer cannot be recovered in higher layers. That is for $i \geq j$

$$I(Y;X) \geq I(Y;h_j) \geq I(Y;h_i) \geq I(Y;\hat{Y})$$

The above statement can be proved in the following way

Theorem 2 *Given $X \to Y \to Z$, $I(X;Y) \geq I(X;Z)$*

Proof 2

$$p(x,y,z) = p(x)p(y|x)p(z|x,y)$$

From the markov chain, Z is independent of X.

$$Hence, p(x,y,z) = p(x)p(y|x)p(z|y)$$

$$\begin{aligned} p(x,z|y) = \frac{p(x,y,z)}{p(y)} &= \frac{p(x)p(y|x)p(z|y)}{p(y)} \\ &= \frac{p(x,y)}{p(y)}p(z|y) \\ &= p(x|y)p(z|y) \end{aligned}$$

This shows that x and y are conditionally independent on y if the y follow the given markov chain. By the definition of Mutual Information:

$$I(X;Y,Z) = I(X;Z) + I(X;Y|Z) = I(X;Y) + I(X;Z|Y)$$

As X and Z are conditionally independent on Y

$$I(X;Z|Y) = 0 \text{ and } I(X;Y|Z) \geq 0$$

Hence,

$$I(X;Y) \geq I(X;Z)$$

Achieving the sufficient statistics is possible only if each layer is the sufficient statistics of its input. $I(Y;\hat{Y})$ is the natural qualifier of the quantity of DNN, but IB principle provides a new measure for the optimality for evaluating the performance at each hidden layer.

This may seem counter intuitive as to why does it help to have more layers in Deep learning. The mutual information $I(X, h_i)$ is defined solely by the distribution $p(x, y)$, which we do not have access to. However we do have access to a set of n samples S from the true distribution. Let $\hat{p}(X, y)$ be an empirical estimate of the true joint distribution. Using this we get the empirical mutual information $\hat{I}(X, h_j)$. The error between the true mutual information and empirical is proven to have the following bound,

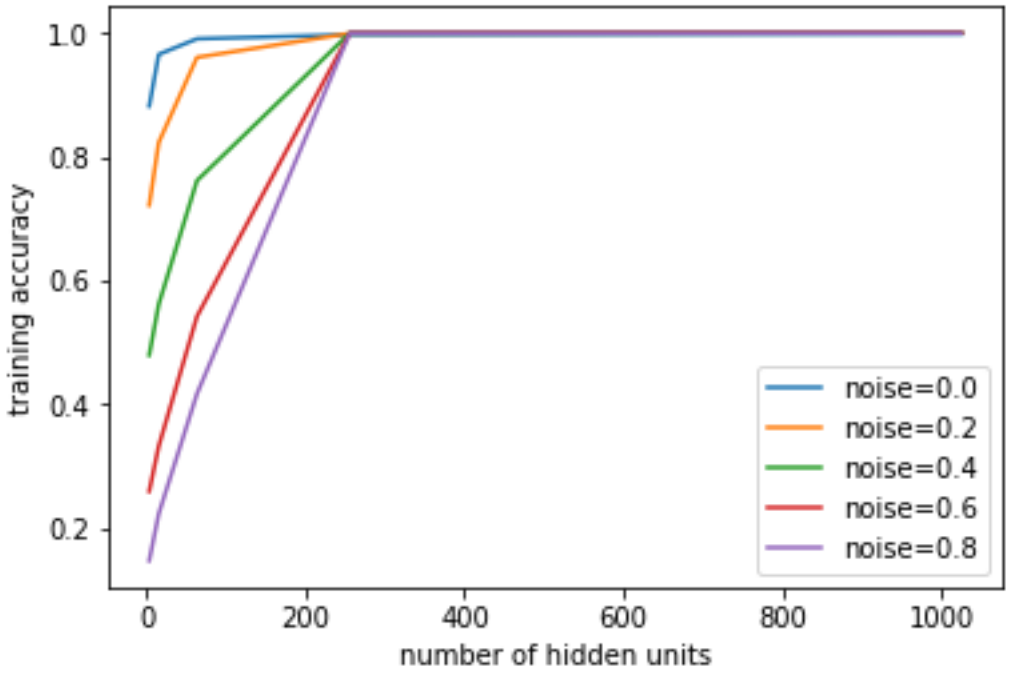
$$I(X, h_i) \leq \hat{I}(X, h_i) + O(\frac{|h_i| * |Y|}{\sqrt{n}})$$

where $|h_i|$ is the cardinality of h_j and is approximately equal to $2^{I(X,h_i)}$. Hence the tightness of the bound depends on how compressed h_j is. The compressed the representation, the empirical is more closer to the true distribution. we care about $I(X, h_i)$ because it is the stand for the generalization error.

3 Experiments

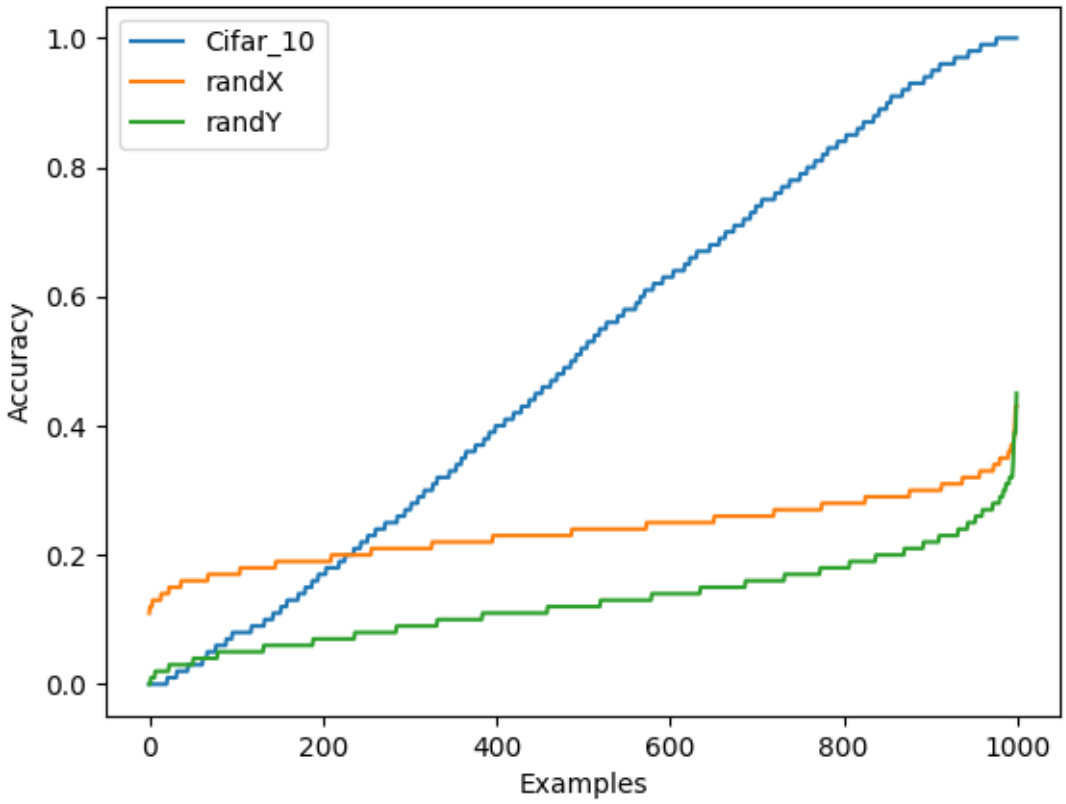
In our work, the generalization performance of neural networks is analyzed by replacing (a portion of) the output labels with random labels denoted by $randY$ or by replacing (a portion of) the inputs with gaussian noise with the same mean and standard deviation as that of the training dataset, denoted by $randX$. We also analyzed the generalization performance and memorization in deep networks by analysing the time to convergence, the number of epochs needed for convergence, the number of units in the hidden layers and the performance of dropout and using regularization.

- Random Inputs (randX): varying noise levels, i.e., the percentage of noise data samples in the input data
- Random Output (randY): randomizing the labels

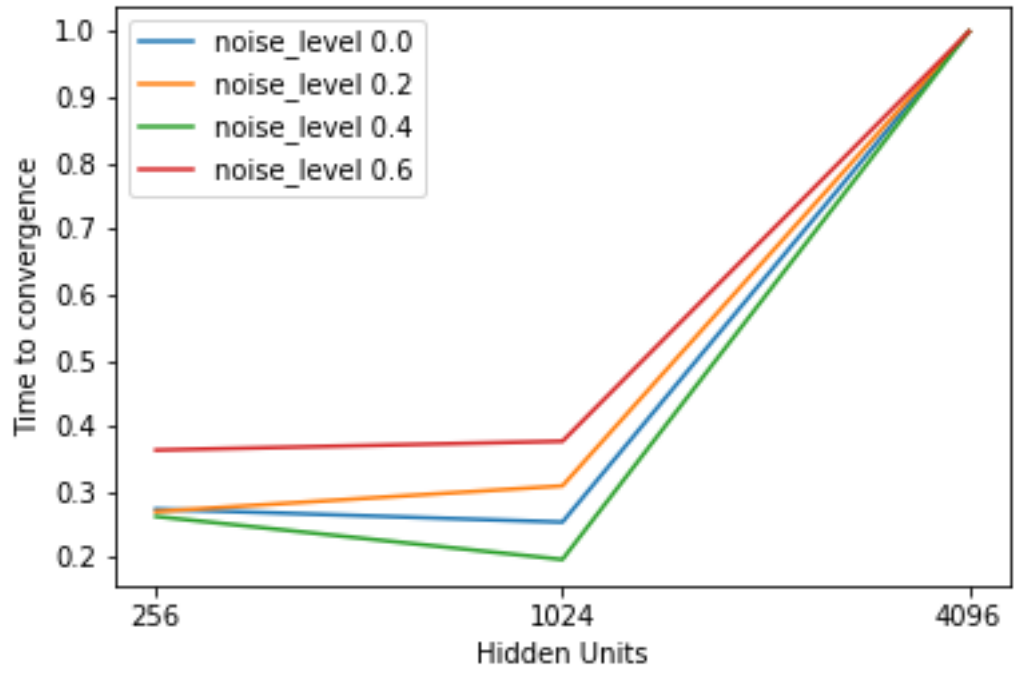


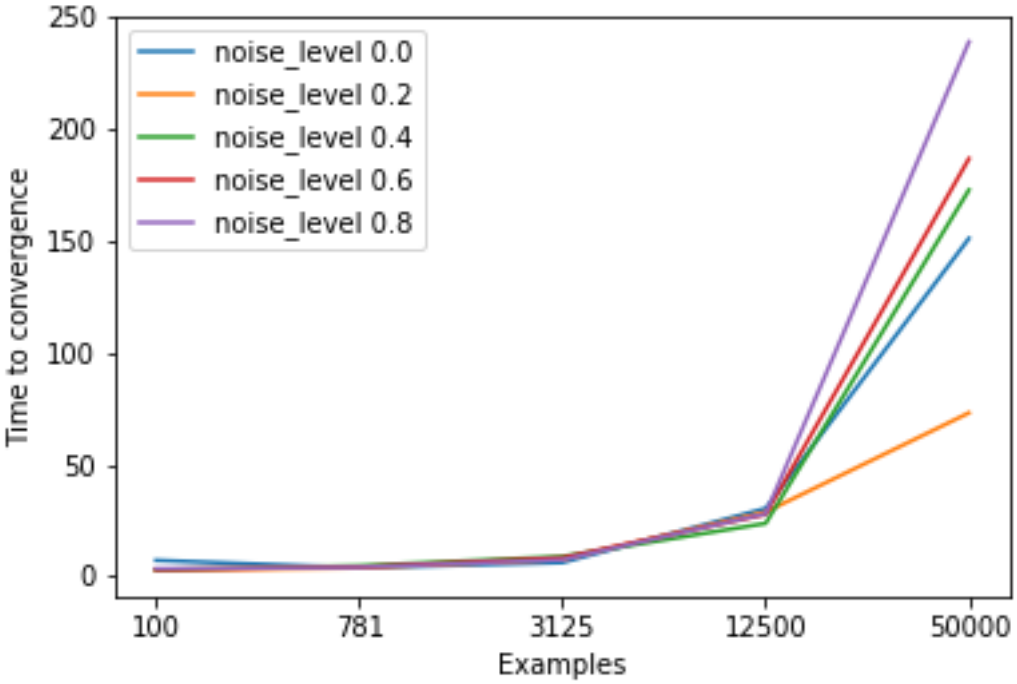
Training accuracy always reaches 1 with sufficient capacity

From the above graph, we show that no matter what the data is, the network always achieves a training accuracy of one given enough capacity, that is the number of parameters which is nothing but the number of hidden units.

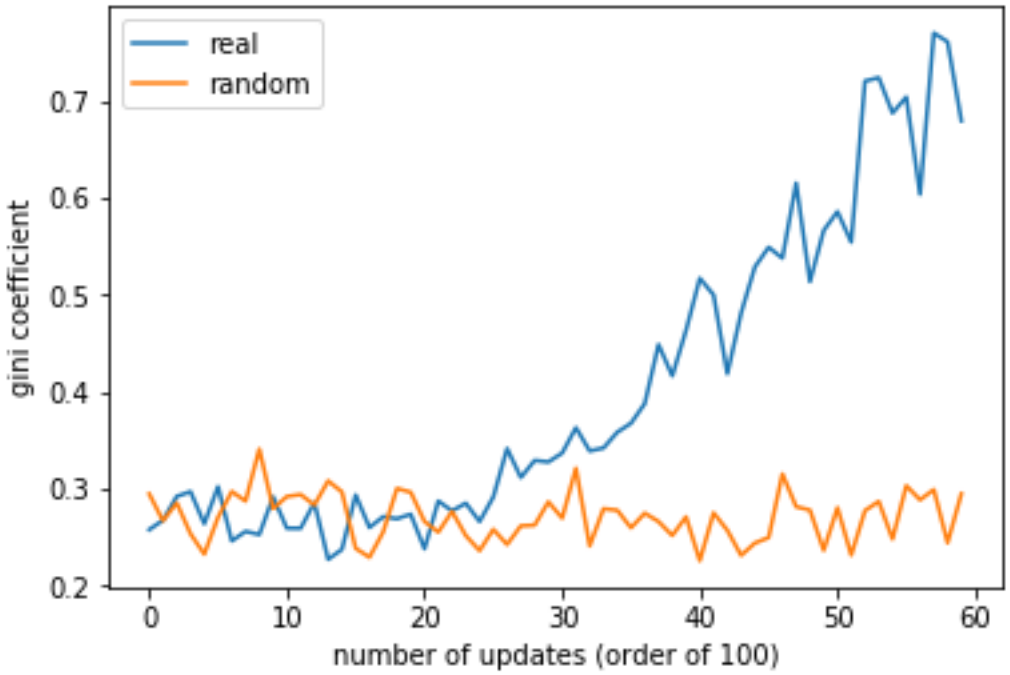


The above experiment was conducted to explain the networks behaviour after one epoch. It can be seen that the accuracy for all the examples is equally same suggesting that the network hasn't learned any specific pattern and hence is equally difficult to classify each example in random data. But for real data, the network starts learning a pattern from the first itself showing that it has learnt a pattern and all the examples that fall into this pattern have a higher probability of classification for the class they belong to.

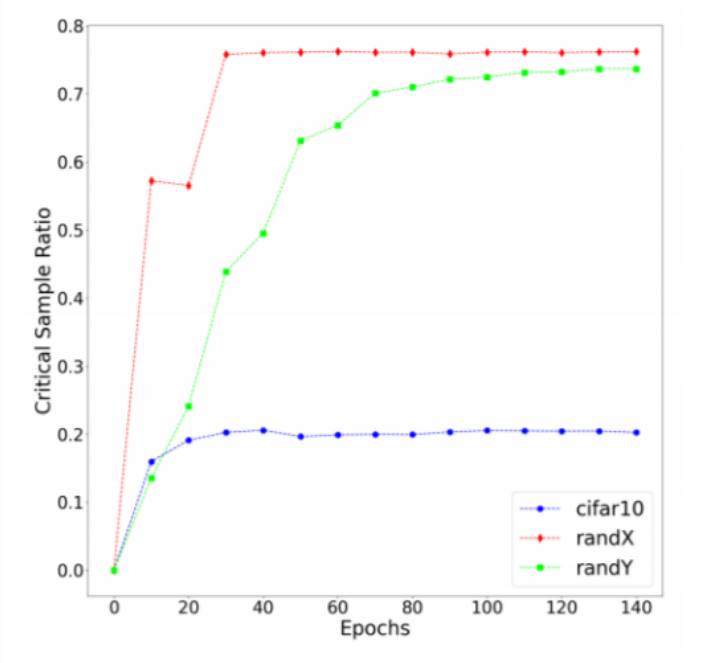




The above experiments have been conducted to measure the time to convergence for different levels of noise data versus the number of hidden units and the number of examples. From the plots it can derived that noise data takes higher time, that is more number of epochs to converge as opposed to real data. That is, in noise data, the network memorizes the data while in real data, the network learns the patterns which is why it takes lesser time to convergence.



Higher gini coefficient suggests that for real data the loss function is highly sensitive once the algorithms learns patterns in the data and for random data the gini index remains the same throughout the training



The higher number of CSRs on the noise data suggest a more complex learned decision surface

4 Conclusion

Although metrics have been defined to measure the generalization and performance of deep networks, we believe that information bottleneck principle gives more information and explains generalizability of neural networks better

References

[1] D. Arpit, S. Jastrzębski, N. Ballas, D. Krueger, E. Bengio, M. S. Kanwal, T. Maharaj, A. Fischer, A. Courville, Y. Bengio, and S. Lacoste-Julien. A Closer Look at Memorization in Deep Networks. *ArXiv e-prints*, June 2017.

[2] N. Tishby and N. Zaslavsky. Deep Learning and the Information Bottleneck Principle. *ArXiv e-prints*, March 2015.

[3] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *CoRR*, abs/1611.03530, 2016.