SMAI

UNDERSTANDING DEEP LEARNING REQUIRES RETHINKING GENERALIZATION (Team-50)

PADALA SUDHEER REDDY

MEKA SAI MUKUND

KATHA ROHAN REDDY

BHUMIREDDY KIRAN REDDY

Introduction

PROBLEM STATEMENT:

Generalization error is defined as the difference between training error and test error. In general, if a model is learning, that is , if the training epochs increase the training error continuously decreases while the testing error first decreases and then increases. Simply, the model starts to overfit after a certain point of time and the generalization error will increase. Generally, if the number of parameters are more, the model is more likely to overfit as it can memorize the data well.

But some deep neural networks have **less** generalization error although having **more** no of parameters. This paper tries to understand what separates neural networks that generalize well from other neural networks. In order to measure the generalization error certain statistical techniques are used such as VC dimension

But the VC dimension formula only applies if input samples are much larger than no of parameters which is not in case with deep learning. So general traditional techniques to find model capacity will fail when applied to neural networks.

Universal Approximation Theorem

A feed forward network with a single hidden layer containing a finite number of neurons, can approximate a continuous function on compact subsets of an n-dimensional Euclidean space. But this Theorem does not guarantee that the weights of this function can be learnt very effectively.

Vapnik-Chervonenkis Theory(VC Dimension)

A classification model f with a set of parameters 'theta' is said to shatter a set of data points (x1, x2, ..., xn) if, for all assignments of labels to those points, there exists a 'theta' s.t. The model f makes no errors when evaluating that set of data points. VC dimension of a model f is the maximum number of data points that can be arranged so that f is able to shatter them. If there are two classes in 2D space then a linear classifier is able to separate them well if the number of data points are less than or equal to 3. (Consider XOR problem for four data points)

So , the VC dimension of a linear classifier is 3.

VC dimension – Statistical Learning Theory

This theory provides us with a formula for the Probabilistic upper bound on test error for a model. However, this is applicable only when the number of parameters D << number of samples N. Therefore, this theory cannot be applied to Neural Networks. The formula for the error is given by:

$$P\left(error_{test} \leq error_{training} + \sqrt{\frac{1}{N}\left[D\left(log\left(\frac{2N}{D}\right)+1\right) - log\left(\frac{\eta}{4}\right)\right]}\right) = 1-\eta$$

Randomisation Tests

To understand how deep networks generalize the data the paper implemented various techniques. One such method is where they tried to feed the deep network with random labels and try to understand the results. They observed that training error was still reduced to zero after some training epochs. Multiple experiments were done by changing the degree of randomization. This established the fact that deep networks were able to memorize the entire training data even when there is no correlation between input and the labels. Thus, deep learning easily fits random labels.

Another method is the shuffling of input pixels. They also used completely random pixels for training. In all cases, the model continues to fit the training images, although it takes a little longer to train the data.

Inferences:

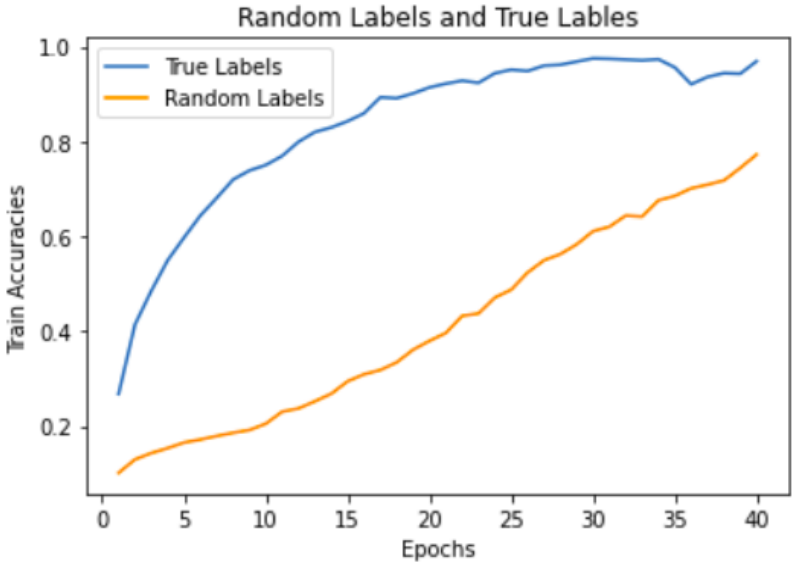
→ Deep neural networks easily fit random labels.

→ Effective capacity of neural networks is sufficient for memorizing entire data sets.

1. **Results**

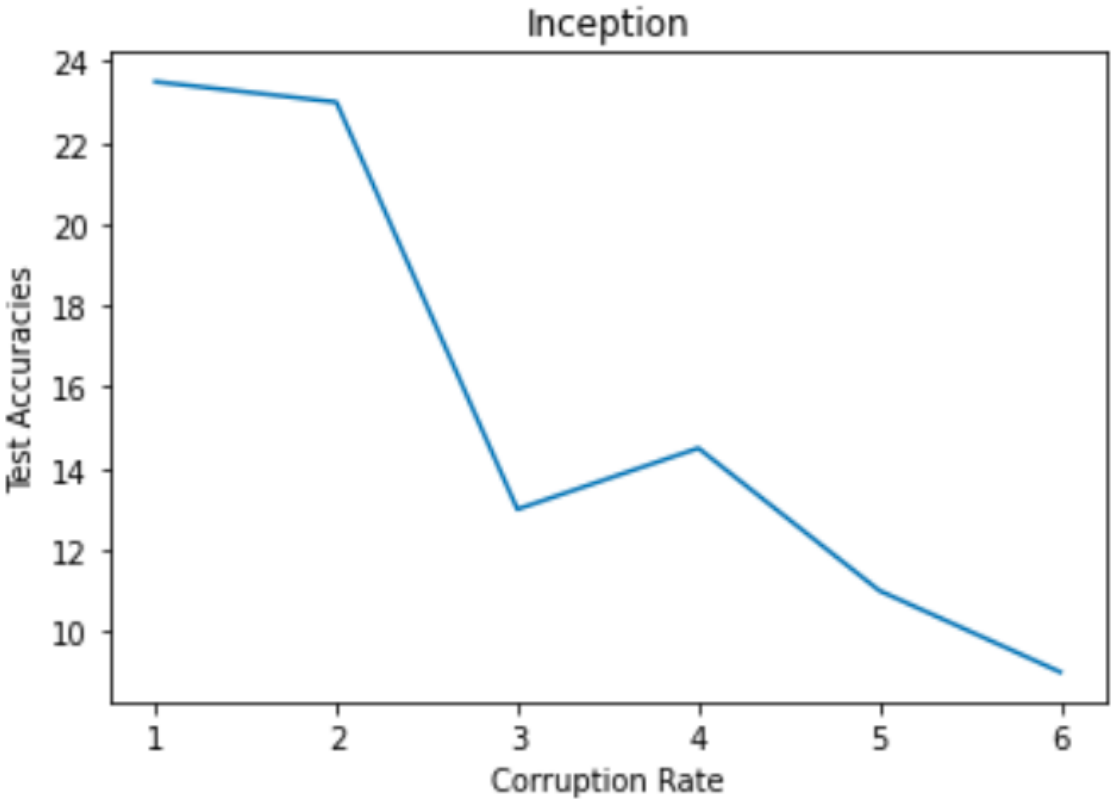
   1. If we use true labels, the results converge to 100% training accuracy after just 30 epochs.

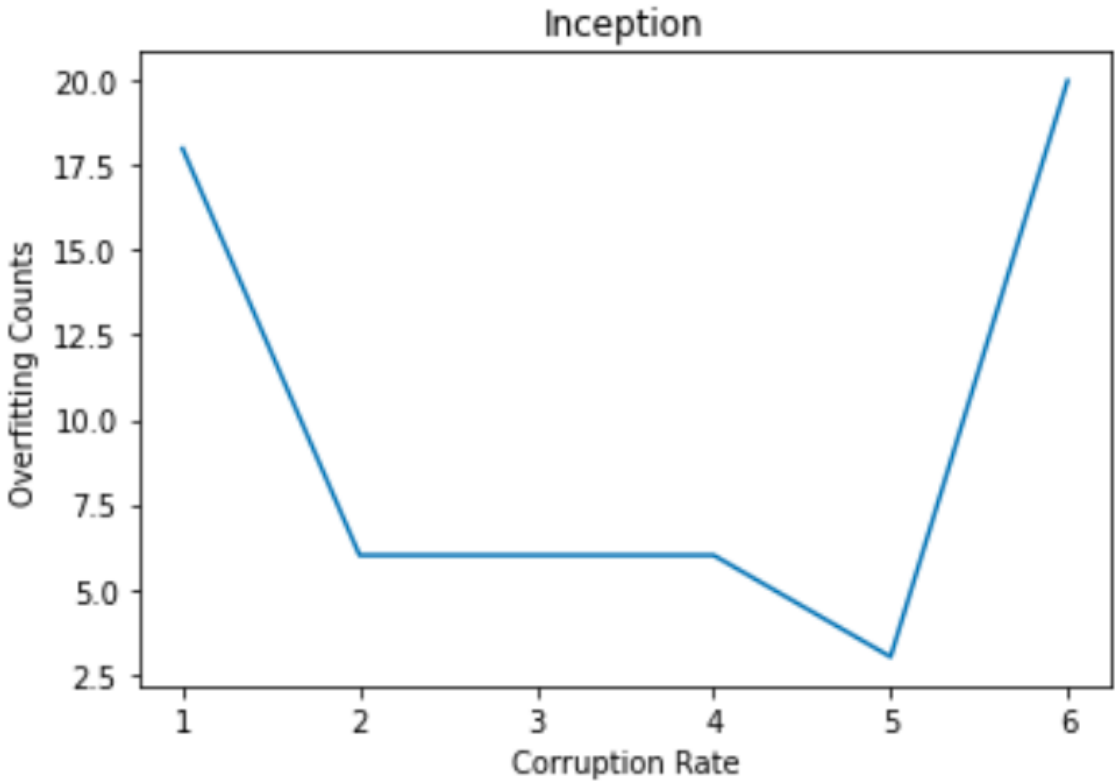
2. However, with random labels, the accuracy reaches 80% just after 50 epochs. This is in line with the results obtained in the original paper (if we consider 50 epochs in both cases).
3. By extrapolation, we can see that the random labels will also converge to 1.0 after several hundred epochs (could not run due to lack of GPU resources).

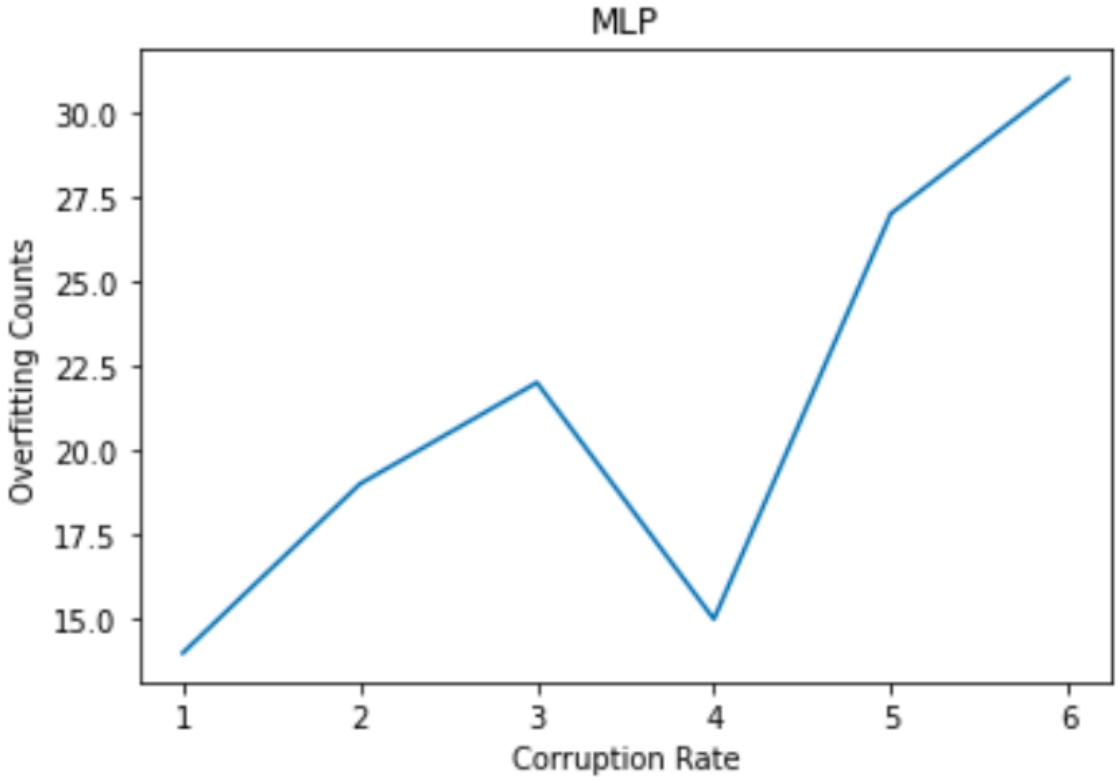


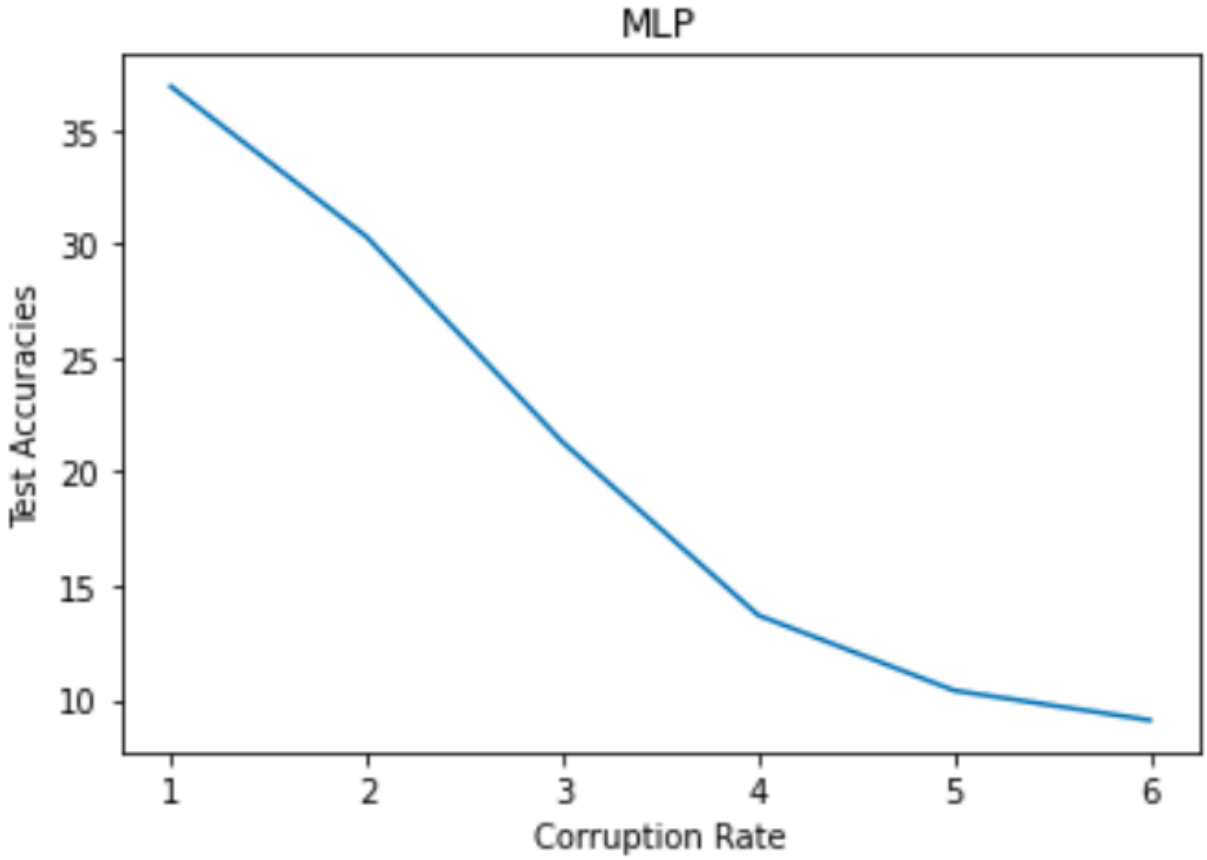
The Neural Network has the capacity to memorize the labels(overfit) even though If we randomize the labels , then the model is taking more time to converge. As there is no correlation between input images and labels the model is not able to distinguish between different classes so takes more time to memorize the whole data. If we use random pixels instead of random labels ,the input data is well separated so expected to perform quite better than using random labels. If we use different architectures , the generalization error varies by a good amount. From this we can infer that there is something implicit in the architecture which affects generalization by a good amount.

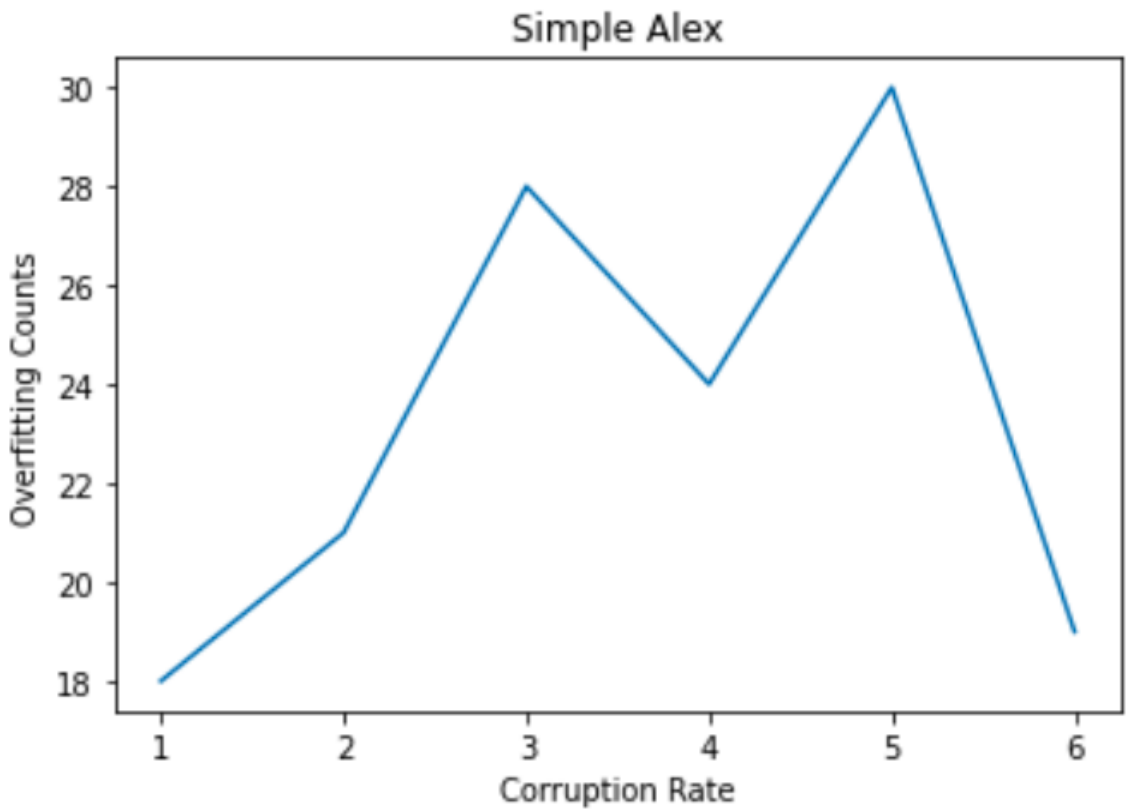
If we increase the partial label corruption rate, the overfitting time increases but the test accuracies decrease.

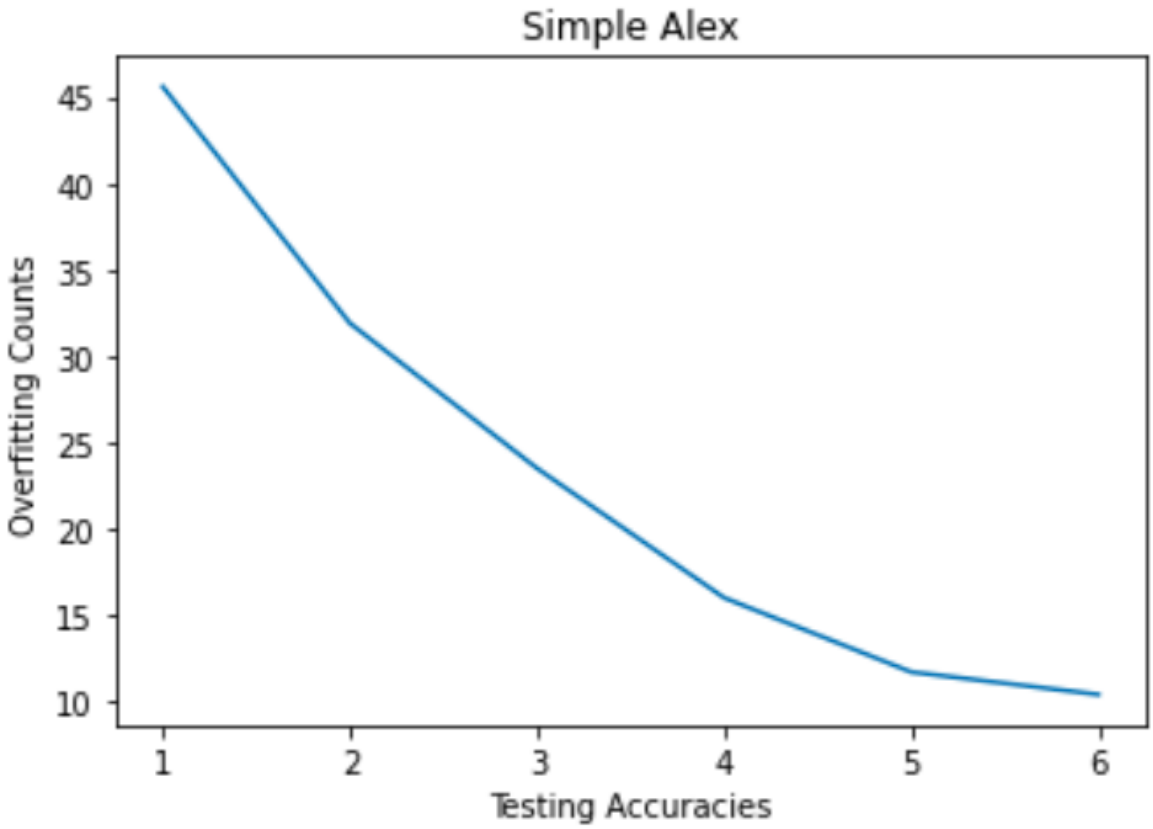

Inception
Test Accuracies
24
22
20
18
16
14
12
10
1
2
3
4
5
6
Corruption Rate


Inception
Overfitting Counts
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
1
2
3
4
5
6
Corruption Rate


MLP
Overfitting Counts
30.0
27.5
25.0
22.5
20.0
17.5
15.0
1
2
3
4
5
6
Corruption Rate


MLP
Test Accuracies
35
30
25
20
15
10
1
2
3
4
5
6
Corruption Rate


Simple Alex
Overfitting Counts
30
28
26
24
22
20
18
1
2
3
4
5
6
Corruption Rate



EXPERIMENTS

Many experiments were performed in the original paper.

The training and testing results were plotted and every time the training error converged to nearly 100%.

Rademacher complexity is a commonly used and flexible complexity measure of a hypothesis class.

Based on this complexity since our model fits random labels this should be 1 for the corresponding model class H. But this does not imply that the model is better generalized hence the traditional techniques fail to explain generalization in deep neural networks.

Regularization Tests

If the traditional techniques are unable to explain how do we explain the better generalization of neural networks One way is to punish the weights so that they don't overfit this technique is called regularization

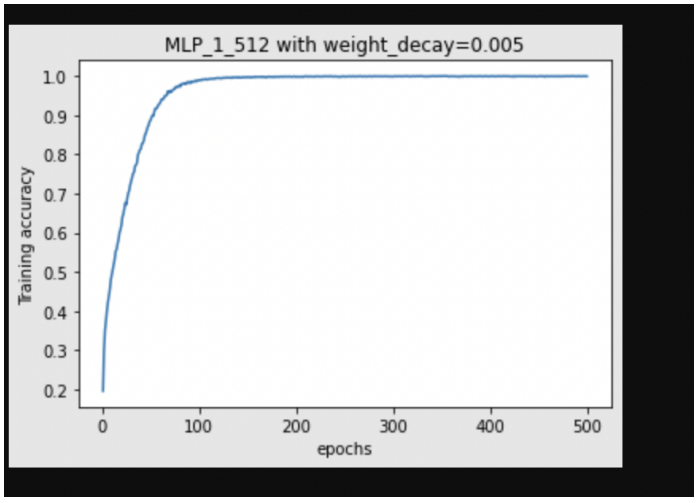
There are two types of regularization:

Explicit : Weight Decay, Dropout, Data Augmentation

Implicit SGD, Batch Normalization

Regularisation Results

Model	Random Labels	Random Cropping	Weight decay	Train accuracy	Test accuracy
Simple Alex	NO	YES	YES	0.9512	65.8
Simple Alex	NO	YES	NO	0.8946	51.0
Simple Alex	NO	NO	NO	1.0	53.7
Simple Alex	YES	NO	YES	0.9974	61.2
Simple Alex	NO	NO	NO	0.1044	8.7
MLP 1x512	NO	YES	NO	1.0	43.0
MLP 1x512	YES	NO	NO	1.0	40.2
MLP 1x512	NO	NO	NO	0.0998	9.5
MLP 3x512	NO	YES	NO	1.0	40.7
MLP 3x512	YES	NO	NO	1.0	41.4
MLP 3x512	YES	NO	NO	0.092	10.0



OBSERVATIONS:

1. When we use regularization, the Neural Network takes more epochs to reach 100% training accuracy.
2. However, the model eventually overfits, after some epochs.
3. Similarly, due to somewhat better generalization, testing accuracy is slightly improved.
4. With weight decay, the improvement is not much. Data augmentation (random cropping and horizontal flipping) produces much better improvement in testing accuracy.
5. However, the improvement is much better if we use a different architecture (inception instead of SimpleAlex).

IMPLICATIONS

As per the results explicit regularization helps in generalization of neural networks but we can not say that it drastically improved the generalization .

Changing the internal parameters of a model resulted in more effect like in data augmentation.

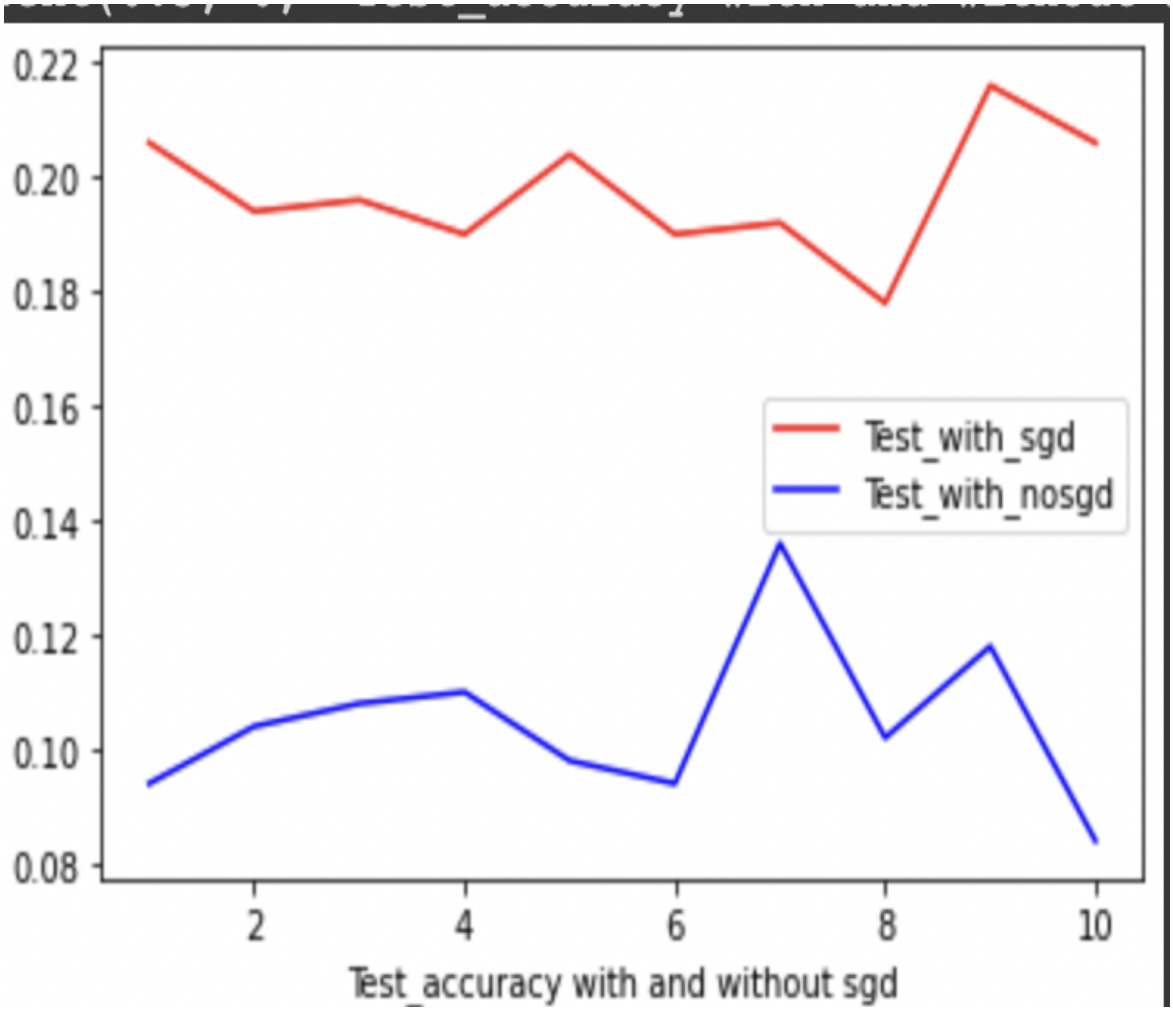
This shows that although regularization helps it is not improving the model's performance.

FINITE SAMPLE EXPRESSIVITY

The paper also explored the power of neural networks . This is because as the neural networks go deep they have more information than the previous layers so for a function to fit we need to have just enough neurons to memorize. If we don't want to compromise on depth we can always increase the width of the neural network. They showed that as the number of parameters p of a network is greater than n, even simple two-layer neural networks can represent any function of the input sample. Where n is the number of inputs.

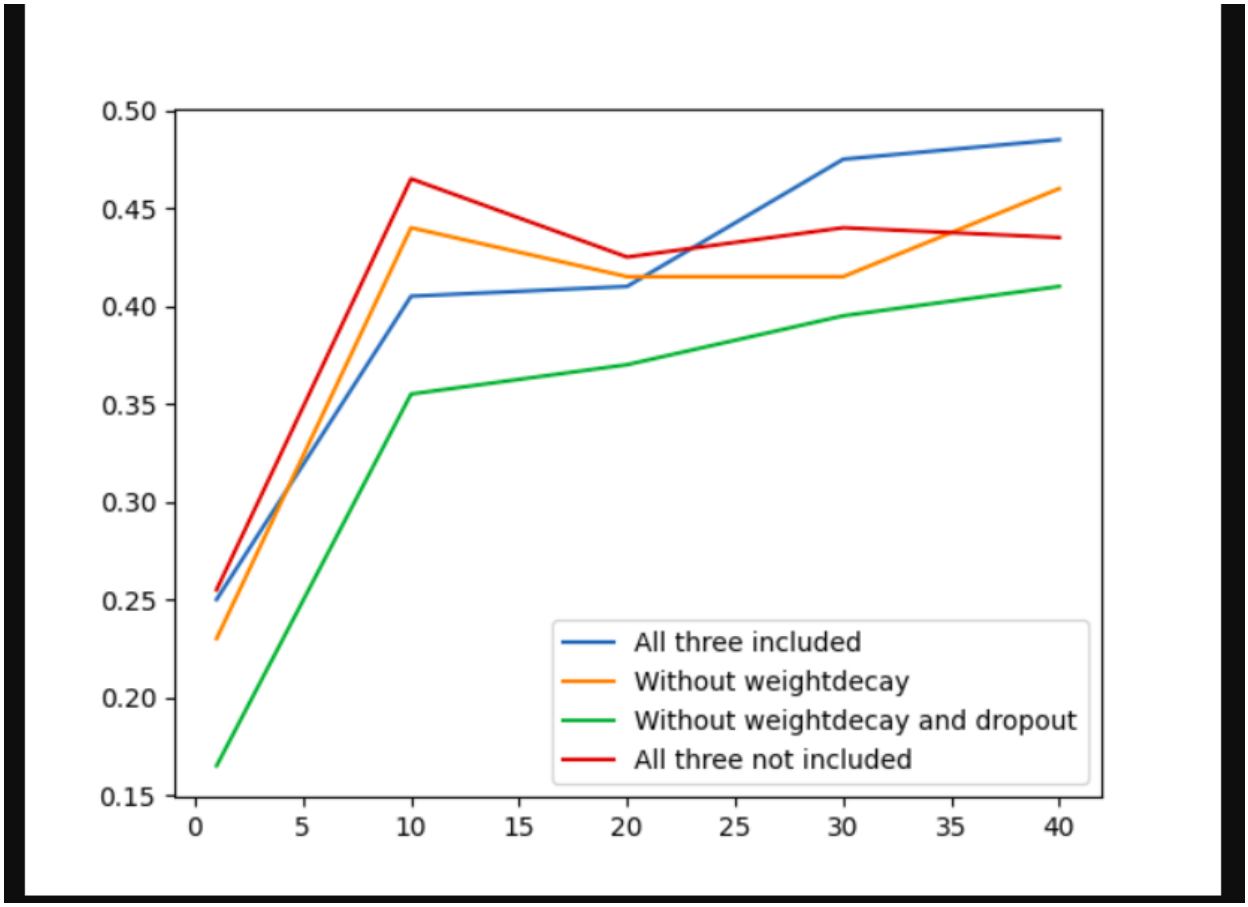
SGD

Stochastic gradient descent is an iterative method for optimizing an objective function. In general, Lagrange multipliers are used to optimize a function $f(x)$. Applying Lagrange multipliers in the sgd will often converge to the solution with minimum norm. This is equivalent to having a l2 norm in the loss function.. This could be one possible explanation of generalization in linear models.



Results:

1. Using SGD (implicit regularization) produces around 10% improvement in the test accuracy. (initially afterwards there was not much effect)
2. However, using SGD only increases the accuracy to 20%, still making it a poor way to generalize.



Observations:

1. From the above graph it is clear that regularization methods do play a role in testing accuracies
2. Although weight decay and dropout improve test accuracy it doesn't improve it by a marginal level
3. But data augmentation would impact the test accuracy by considerable amount

4. This proves that bigger gains can be achieved by simply changing the model architecture.